Journal of Behavioral and Experimental Finance 23 (2019) 138–160

Contents lists available at ScienceDirect

Journal of Behavioral and Experimental Finance

journal homepage: www.elsevier.com/locate/jbef

Review article

Web-based experimental economics software: How do they compare to desirable features?

Shu Wing Chan [a,*], Steven Schilizzi [a], Md Sayed Iftekhar [a], Raymond Da Silva Rosa [b]

[a] Agriculture and Resource Economics, UWA School of Agriculture and Environment, M087, The University of Western Australia (UWA), 35 Stirling Hwy, Crawley WA 6009, Australia
[b] Accounting and Finance (UWA Business School), The University of Western Australia (M250), 35 Stirling Highway, Crawley WA 6009, Australia

ARTICLE INFO

Article history:
Received 22 December 2017
Received in revised form 14 March 2019
Accepted 17 April 2019
Available online 9 May 2019

JEL classification:
code
C81
C88

Keywords:
Experimental economics
Web-based
Software
Online experiments
Web-based experiments
Economic experiments

ABSTRACT

Web-based experiments that cut across the lab vs. field distinction are increasingly popular with economists. However, non-standardized software features and services hinder comparability and replication. This study reviews a wide selection of experimental economics software packages and evaluates them against criteria based on the logistics and operational requirements of economic experiments. We find that oTree and SoPHIE rank highest across criteria, but Veconlab and classEx might be suitable for those with a dominant need for a large library of ready-made experiments. We find a portability gap: no presently available software allows portability of experiments across platforms because of technical complexity and the challenging coordination needs of experimental economists. As a result, experiments may be replicated only on the same platform or with the same software, but general replicability is slow and costly. This constrains the development of experimental economics as a replicable science.

Contents

* Corresponding author.
E-mail addresses: chansw@ieee.org (S.W. Chan), steven.schilizzi@uwa.edu.au (S. Schilizzi), mdsayed.iftekhar@uwa.edu.au (M.S. Iftekhar), ray.dasilvarosa@uwa.edu.au (R. Da Silva Rosa).

https://doi.org/10.1016/j.jbef.2019.04.007
2214-6350/

1. Introduction

Experimental economics addresses economic questions in settings where data are "deliberately created for scientific (or other) purposes under controlled conditions" (Friedman and Sunder, 1994). Experimental economics may use technology as simple as pen and paper or cards, e.g., Chamberlin (1948), but web-based experiments have become increasingly popular. They allow researchers to (i) run experiments with more diverse samples as well as specific or rare population groups, (ii) recruit larger subject pools which can give higher statistical power (Reips, 2000), (iii) conduct cross-cultural experiments in real time (Paolacci et al., 2010), (iv) run experiments more quickly (Reips, 2002), and (v) reduce experimenter effects by absenting themselves from the experimental context (Reips, 1996). All these possibilities have the potential to increase external validity. Potential disadvantages of web-based experiments include (i) the possibility of multiple submissions (Reips, 1996), (ii) possible adoption of false identity by participants, and (iii) the possible high rate of incomplete responses (Arechar et al., 2017; Reips, 2002). Whilst these issues are challenging, Reips (2000) provides some potential solutions and discusses the less tractable problem of replicability of web-based experiments outside the technological platforms used to host them.

One practical challenge for experimenters recognized in these studies is identification of the software application best suited for their purpose. Research on the advantages and disadvantages of software packages is lacking notwithstanding the variety of products like oTree (Chen et al., 2015), Seaweed (Chilton et al., 2009), Zocalo (Hibbert, 2005), BoXS (Seithe, 2012), classEx (Giamattei and Lamsbdorff, 2019), and SoPHIE (Hendriks, 2012). In addition to web-based software packages with a graphical user interface (GUI), there are also programming frameworks like Willow (Weel and McCabe, 2009), CORAL (Schaffner, 2014), LabSEE (LabSEE, 2015), and jsPsych (de Leeuw, 2015) which allow more flexibility but require higher programming expertise. In contrast, hosted software platforms allow running of standard experiments for

research and teaching purposes but do not allow writing one's own experiment. The advantage is that they can be used off-the-shelf without any programming required. The variety of offerings make it common to see different groups of researchers implementing the same experiments with different software packages. This makes a comparative study of their respective merits and demerits time-consuming.

Palan (2015) reviews the outcomes of asset market experiments using the following web-based experimental economics software packages: EconPort, Flex-E-Markets, jMarkets, Rotman Interactive Trader and SoPHIE labs with the "Graz–Innsbruck Market System" (GIMS) (a non-web-based software built on the z-Tree platform). Palan (2015) concludes that z-Tree (Fischbacher, 2007) is a viable platform due to its continuous development and widespread use since 1995. Palan (2015) assesses the viability and availability of many custom-built experimental economics software packages. His criteria for software for asset market experiments fall into two categories: criteria specific to asset markets and criteria specific to market mechanisms. Asset market criteria include complete, time-stamped data records, customizability, and extensibility, reliability and lifetime, non-standard hardware and experiment designs and cost. Market mechanism related criteria are a user-friendly interface, choice between single and multi-unit trading, single or multi-period trading with or without wealth carryover, possibility of trade in multiple markets or over the counter, parameter specification, designated trader roles, order validation, order types and algorithm trading. Palan (2015) highlights the inefficiency caused by the parallel development of similar applications as time and cost are not trivial. He proposes a standardized solution software called GIMS which uses a z-Tree program.

Jansen et al. (2014) undertake similar research in comparing software packages for experiments in social ecology (z-Tree, GameWeb, CSID framework, VCWeb, MobLab, boxes, Veconlab, NetLogio HubNet and ConG). They focus on specific-purpose behavioral and experimental economics software packages. The evaluation criteria (Jansen et al., 2014) use are more extensive than in Palan (2015), which include purpose, participant mode such as local or lab, software model like installation required, web-based, local or global, number of standard games that come with the software, capacity to create new games, communication between participants, expertise needed to implement and run a new game, open source, hardware requirement, operating systems, data storage, support group size, documents in terms of pages of documentation, user community and estimated number of papers published that used the platform.

Palan (2015) and Jansen et al. (2014) focus on specific-purpose experiments without a web-based focus. We address the gap in the literature in assessing how well general-purpose web-based economic software meets operational criteria that facilitate comparability and replicability. We begin by identifying a set of ideal desirable features and evaluate publicly available web-based software for general-purpose economic experiments. We classify general-purpose software as those that allow users to run more than one type of experiments.

The rest of the paper is organized as follows. We first review the logistics and operational requirements of web-based economic experiments to assist us in defining desirable features. We then identify desirable features for experimental economics software and discuss how data on such software is collected and classified. This leads us to assessing selected software packages against the list of desirable features. We close by discussing possible options for enhancing the use of web-based economic experiments.

2. Logistics and operational requirements of web-based economic experiments

Croson (2003) and Friedman and Sunder (1994) list key processes required to conduct web-based economic experiments. These include experiment design and testing, subject recruitment and management including identification and maintenance of privacy, communication including instructions, running the experiment and debriefing, payment management, grouping and matching of subjects using different methods, and data management and reporting. These processes determine the attributes of an ideal online platform, as they must all be addressed.

For laboratory and online experiments, there are many commonalities, from design to debriefing. We discuss only the main differences between them, which include the location of subjects, the potential difference in subject responsiveness, user interface compatibility, data storage and payment method.

Laboratory experiments typically run in a dedicated computer laboratory with specialized servers and client software such as z-Tree and z-Leaf (Fischbacher, 2007), installed and run on a local area network. In contrast, online experiments can be, in principle, run anywhere and anytime: in a computer laboratory but also in the field with mobile devices. Location of subjects is a significant difference between online and laboratory, as subjects can be scattered across different locations instead of sitting in the same room. Thus, online experiments can use a diverse set of subjects and have more flexibility in targeting a specific group.

Laboratory experiments may have more control over subjects' interactions compared to online experiments. Any experiment that requires real-time interaction between subjects can be affected by their internet connection as subjects are not on the same network. Network latency is possible for some subjects and may have an impact on the running of the experiment. However, experimenters can minimize the impact by providing a test web page for prospective subjects to test their network performance to the test server.

User interface is another difference between laboratory and online experiments. For laboratory experiments that use desktop software like z-Tree, experimenters are less concerned as the client software should display everything the same across computers. However, for online experiments, it depends on the browsers used by subjects. The way to mitigate this problem is to provide subjects with information on the browsers and the versions that are tested and supported for uses on the web software.

For data storage such as experiment settings and results, web-based software packages tend to use a common relational database management system (RDMS) such as Microsoft SQL Server, MySQL or PostgreSQL. On the other hand, non web-based software packages for laboratory experiments are likely to store data on file rather than in a database. In terms of incentive payments, there is a difference between laboratory and online experiments. For laboratory experiments, cash is commonly used, although other types of rewards are possible. Online experiments are more convenient when fully integrated with an online payment mechanism like PayPal. Subjects can be paid online if they provide relevant details for electronic banking during registration.

The logistical and operational requirements provide the foundation for defining desirable features of a web-based platform for running economic experiments, which we discuss in the next section.

3. Key desirable features of web-based experimental economics software

All experimental economics software packages share some common features. In this section, we examine the key features

important for the utility and success of web-based economic experiment software. We group desired features into ten categories: (i) Software license, support and maintenance, (ii) User interface and usability, (iii) Data management, (iv) Flexibility and portability, (v) Subject management and payoff, (vi) Experiment dashboard, (vii) Application development and extensibility, (viii) Multi-lingual support, (ix) Multi-tenancy support, and (x) ready-made documented examples.[1]

3.1. Software license, maintenance and support

We discuss why open-source is a desirable software feature and how community support can be crucial to the success of open-source software. We also discuss the importance of active development.

3.1.1. Open-source

Open-source software (OSS) often provides a low-cost solution compared to commercial software. Examples of highly regarded and widely used OSS products include Linux, Apache, Drupal, programming languages like Java, PHP, Python, Ruby, and relational databases like MySQL and PostgreSQL. Factors that affect the success of OSS include user base, language translations, responsibility assignment and modularity (Midha and Palvia, 2012). On the other hand, open-source projects could fail due to various reasons, such as usurpation by a competitor, obsolescence, lack of time for development and maintenance by developers, lack of interest, outdated technologies, and low maintainability (Coelho and Valente, 2017).

3.1.2. Active community support

Active community support contributes to the success of OSS by increasing the chance of adoption. Many OSS projects use discussion forums, mailing lists, Facebook pages, and other collaboration means to connect with their community. Mature OSS projects like Linux, Apache, PHP, and Java organize worldwide annual conferences for their users in order to share ideas and knowledge. Community support promotes the software and improves collaboration on solving problems and sharing codes and ideas. Many OSS projects either collaborate with third-party organizations or establish a commercial arm for individuals and companies that are more comfortable with paid support.

3.1.3. Active development

Experimenters are likely to re-run experiments for validation purposes, to extend the research, or to increase the sample size. Software that is maintained and under active development ensures continuity and bugs are addressed. Jansen et al. (2014) report that software repository systems such as GitHub allow retrieval of an old version of the software; however, there is a risk the preserved version of the software might not work on future hardware and operating systems and browsers.

Palan (2015) lists the expected lifetime of software as a key consideration in choosing amongst software packages. Many open-source experimental economics software packages have a short duration of viability. A common cause is that developers complete their thesis or project and move on. This underscores the importance of active ongoing community support and engagement.

[1] This list was prepared based on an extensive consultation of existing literature. However, there are several other desirable features such as availability of treatment variations, continuous-time interactions, and time-out handling, which have not been included in this paper.

3.2. User interface and usability

Usability is an important part of software quality as it plays a critical role in its uptake. The popularity of mobile devices means web-based software packages are expected to work on multiple devices. We discuss the advantages and disadvantages of multiple devices and the importance of a graphical user interface (GUI).

3.2.1. Multiple device support

Web software allows users to be platform- and device-independent. Traditionally, experimenters run laboratory experiments with either Windows or Mac desktop computers.

Responsive web design (RWD) methodology enables Web software to work on different devices with a single design and URL. It rearranges the user interface according to the screen size for providing the same user experience (Gardner, 2011). Bootstrap, a very popular RWD framework, developed by Twitter, is widely used for Web applications. Some alternatives are Bulma, Skelton, Pure, Groundwork, Cardinal, powertocss, Mueller, Bootflat, Material and Endnote. Major advantages of RWD include one single URL and one version of the code to deploy and update. A drawback is that RWD needs to be downloaded and loading times are often slow for mobile devices. The problem is mitigated with combining RWD and Server-Side Components (RESS) which are processed on the server first before sending to the client (Wroblewski, 2011).

3.2.2. Usability

Usability plays an important role in software development as it increases user acceptance and increases engagement. Nielsen (2012) defines usability by five quality components: learnability, efficiency, memorability, errors and satisfaction. Learnability is how easily users can accomplish basic tasks when they use the design for the first time. Efficiency is defined by how quickly users can perform tasks accurately once they have mastered the software. Memorability is how easily users can re-establish proficiency if they stop using the software for a while. Factors that can improve memorability include software that validates user input and provides feedback and recovery for errors. Satisfaction is defined by how pleasant it is to use the design which can determine how likely users will use the software again in the future.

3.3. Data management

A database management system is essential. Ideally, the software used in a particular experiment allows data to be exported in a format usable in other software packages (e.g. Excel, Matlab, or R).

3.3.1. Data stored in a database management system

Database management systems (DBMS) should be used if data persistence is required. In comparison with storing data in a file, DBMS performs better regarding integrity, reliability, performance, and security. For its greater speed, system memory is another resource for storing experimental data, but it can result in data loss due to sudden power outage or operating system problems. Structured Query Language (SQL) is a commonly used programming language to manage data in DBMS, although it is not the only option. Some programming languages provide object relational mapping (ORM) which allows programmers to manage data in the database without writing SQL statements. SQL manages data in the backend (DBMS) while HTML handles the front-end interface.

3.3.2. Data export

Data collected is often required to be exported for further analysis or reporting using other software packages. Ideally, software should allow experimenters to export data in commonly used formats such as CSV, text or XML files. For instance, CSV is a format that can be imported into commonly used statistical and analytics programs such as SPSS, Matlab, and R.

The availability of data export capability has economic implications. If data cannot be exported into a format that can be used by other software packages, experimenters will have to input the data into other software manually. The process can be resource intensive and time-consuming. This makes data exportability in multiple commonly-used formats an important desideratum.

3.4. Flexibility and portability

Research questions can differ across experiments, so it is desirable that experimenters have the flexibility to write their own experiments. Further, the ability to run an experiment on other software allows easier replication of results.

3.4.1. Flexibility to write one's own experiment

Off-the-shelf experimental designs are convenient for learning new software and for teaching, but the ability to design new experiments is typically critical for researchers. Ideally, the language used for writing experiments should be a common one, as this will make it easier to access resources.

3.4.2. Restricted vs. general replication

Replication is important in scientific experiments (Ioannidis et al., 2017) and software should facilitate it. There are two types of replication: restricted and general. The restricted kind allows replication only with the same software. Typically this involves export or import of proprietary format files or the same source code files with instructions that may be used by other experimenters to replicate the experiment with the same software. The instructions include the parameters used and the location of the source code files to be copied, which can vary between software packages.

General replication allows an experiment to be written on one software and used with another software. The smaller the modification required, the better. General replication is also known as portability, and we discuss this term next.

3.4.3. Portability

Portability allows experiments to be not only replicated in a restricted sense – on the same platform or with the same software – but allows general replicability, across platforms and different software, to be easier, quicker, and, often, less costly. Experiments written in a programming language or for a particular software carry the risk that they might not work in the future when it is no longer maintained. To mitigate this risk, the source code for experiments should be portable across different software, like HTML pages, which work across different browsers. For example, Extensible Markup Language (XML) may be used to write experiments with software built to understand an open standard XML. Experimenters would write experiments in XML format using the defined tags to describe each step. A standard experimental economics markup language (EEML) can offer several advantages: an easier-to-learn descriptive language, portability and replication of experiments across different software, and creation of GUI tools for speeding up the development of experiments.

3.5. Subject management and payouts

For conducting experiments, one needs to manage subjects and, typically, pay them.

3.5.1. Subject management

Participants may be recruited in different ways. Experimenters need to communicate with participants and record their details. ORSEE (Greiner, 2015), a web-based participant management system, has been widely used by experimenters. Amazon Mechanical Turk (Amazon Mechanical Turk, 2015) is another option that allows experimenters access to a global labor market. Similar services are HROOT (Bock et al., 2014) and Prolific (Prolific, 2018; Palan and Schitter, 2018). Advantages of these systems include easier management, possible automated process and being paperless.

Experimental economics software should ideally work with software packages or online platforms for external participant management or have built-in participant management functions. However, web-based software packages can use APIs (Application programming interface) for seamless integration with external online platforms, which provide a better user experience.

3.5.2. Group matching

Many experiments require subjects to be allocated to groups. The basis of grouping may vary. For instance, in fixed grouping, a subject is paired with another subject throughout the experiment. In random grouping a subject is paired with a different subject in each round. Ideally, software should allow experimenters to design and write their customized grouping mechanism via the user interface.

3.5.3. Subject payouts

Subject payouts provide incentives to participants. Payment can include a show-up fee, a conversion rate, and any specific payment component for the experiment. Also, a period can be paid or unpaid as some periods might be used for practice or selected randomly for payment.

The software should give experimenters flexibility to manage subject payments. Other than the usual payment type, experimenters should be allowed to customize payment settings according to their needs, e.g. being able to add customized fields for specific experiments. Best practice requires that the setting and payment details for each experiment be recorded in a secure database.

Integration with Amazon Mechanical Turk or PayPal or similar systems can reduce the need for software to directly manage the transactions, as this can be costly. The advantages of using common third-party payment gateways include a convenient distribution of payments.

3.6. Experiment dashboard

An experiment dashboard is a visual representation that allows the experimenter to monitor, adjust, and execute experiments. For example, it can show how each subject is doing during the experiment and highlight the subjects who are falling behind, either using color or some other form of a visual representation.

The dashboard should have default settings with essential parameters displayed on the screen that are relevant to most experiments. At the same time, it should allow experimenters to customize them for their own needs. A well-designed experiment dashboard provides experimenters with useful insight for managing experiments.

3.7. Application development and extensibility

3.7.1. Availability of application programming interface (API)

An application programming interface (API) is a set of functions that allow developers to access the features and data of the software. An API extends functionalities and facilitates integration

with third-party software. Well-designed and well-documented APIs help to ensure developers can extend the software coherently and consistently.

An advantage of using an API is the connection to the core of the software without the need to understand the underlying structure. It also reduces the need to rewrite code for common functions and thereby improves productivity as developers can devote more time and effort to extending software functionalities for specific purposes.

3.7.2. Modular design

Modular design in software development consists in subdividing the system into smaller parts called modules. It allows reusability of modules within the system and even in other systems. Apart from reducing the cost of duplication, modular design can improve the quality of the software as each module can be tested independently before integration. Another advantage is that work can be shared among developers when functions are split into modules and carried out concurrently. However, the quality of modules needs to be monitored and maintained through peer review of the code. Any faulty or low-quality modules can create a risk to the software as a whole.

3.7.3. Automated testing

Automated testing is typically easily repeated and makes the software more reliable. For web software, it is possible to run automated testing with software like Selenium (2015) by recording the steps of actions for replay either through the GUI tool or by scripting with its API. It can also provide a means of regression testing, functional testing, and load testing.

Most programming languages have libraries for creating a web bot. A bot is a software application that runs automated tasks repetitively. There also exists an open-source framework for building bots like Robot Framework (2017). It is useful if the software allows experimenters to simulate participants with web bots because a significant number of potential bugs may be identified before conducting the real experiment.

3.8. Multi-lingual support

Multi-lingual support allows the user interface to be displayed in different languages, which can encourage more users to use the software. Depending on the design and availability of resources, different approaches might be taken to implement multi-lingual support within the software. One method is to use a text file to store the user interface text and their translation of each language separately. The application needs to be programmed for accepting the language text file and displaying the translated text in the file on the user interface. The most common way to implement multi-lingual support is using GNU gettext and the corresponding internationalization routines.

3.9. Multi-tenancy support

A multi-tenancy system allows more than one experimenter to run experiments at the same time. Experimenters within the same organization can use the software run by their IT department, which allows them to focus on conducting experiments rather than dealing with system administration tasks.

Each experimenter creates an account and experiments run by them and the data generated are logically separated from other experimenters within the system. Also, it is possible to allow experiments to be shared within the system with other experimenters. This can reduce the cost of duplication and also facilitate collaboration. Importantly, data can be centrally stored and backed up to protect against data loss. This gives experimenters more time and resources to focus on experimental design and implementation.

3.10. Ready-made documented examples

Experimenters without any prior programming experience can learn from reading and modifying example codes. Their availability can ease the learning curve and encourage buy-in of the software. Documentation improves readability and reduces the cost of maintenance. A basic rule in documenting code is to always use meaningful names for variables, constants, functions, and classes with inline documentation to explain their purpose. Proper documentation facilitates reusability rather than a reinvention of the wheel. Many classic economic experiments have been replicated using different software and programming languages. Ideally, software should provide some ready-made examples with well-written comments and documentation.

In the next section, we discuss our research method and how we collect and analyze the data of Web-based Experimental Economics Software Packages (WEESPs).

4. Research method and data

4.1. Research method

To answer our research question, we searched for available WEESPs, gathering relevant data from the software documentation and websites, evaluating the features of the software packages, and comparing them against those described in the previous section. A review of experimental economics journals yielded limited information other than confirming the current dominance of z-Tree.

To increase the potential search results outside the mainstream experimental economics journals, we used the Web to search for relevant research articles. We performed a search via Google, Google Scholar, Scopus, and journals listed in ProQuest using four keyword clusters: "experimental economics software", "Web experimental economics software", "mobile experimental economics software" and "online experimental economics software". We also checked the referenced links from the returned result pages and papers, for mentions of software not found in our primary searches. We visited each software website to collect all the relevant information. In some instances, we contacted the software author for information about their software.

One of the major challenges was that some software developers had not maintained the software on an on-going basis and the contact information available on the website is no longer valid. Further, while the majority of the authors replied to our request, some authors did not respond.

4.2. Data

4.2.1. Data collection

In total, we found thirty-three experimental economics software packages over the period late 2015 and early 2016 when we performed the search. For each software, we collected software name, URL of the software website, the platform it was developed on, whether it was free or commercial, first release date, latest version number, last release date, any published or working papers, type of experiments supported, experiment conducted in published or working papers, license type, integration with Amazon Mechanical Turk, and availability of documentation and support. The collected data provide the foundation for further filtering and analysis. Appendix A provides descriptive information about the software.

The majority of software packages are developed in three programming languages: Java, PHP, and Python. A few of them use Model-View-Controller (MVC) frameworks like Python Django and PHP Zend along with Bootstrap, a framework that allows

Table 1
Programming languages, Web frameworks and other technologies used by the experimental economics software assessed in this study.

Programming languages	Web frameworks	Other technologies
Java: The AEE Lab	Python Django: **oTree**[a]	Node.js: Jars
BoXS		
ComLabGames	PHP Zend: **SoPHIE**[a]	.Net Framework: **EconPlay**[a]
ConG	SoPHIE lab	
CORAL[a]		CGI-bin: t-Tree
EconPort[a]	Bootstrap: classEx	
jAuctions	**LabSEE**[a]	
JessX	**SoPHIE**[a]	
jMarkets	SoPHIE lab	
Moblab	**oTree**[a]	
Multistage		
Weblab		
Zocalo		
PHP: **classEx**		
FEELE lab[b]		
LabSEE[a]		
Seaweed		
SoPHIE[a]		
SoPHIE lab		
Veconlab[a]		
Python: **oTree**[a]		
PEET		
Willow[a]		
JavaScript: **jsPsych**[a]		
nodeGame		

Note:
[a]Assessed software.
[b]FEELE uses Veconlab.

user interface to be responsive on different types of devices. We list the programming languages, Web frameworks, and other technologies used by the software packages found in Table 1. In Table 2, we indicate whether it is web-based, open source, free or commercial, and specific-purpose or multi-purpose. Appendix B includes a brief description of the programming languages, Web frameworks and other technologies.

4.2.2. Data inclusion

The selected software packages for review include web-based, open-source/free, accessible either via download or its website, and multi-purpose for creating more than one type of experiment, with documentation and contact details.

4.2.3. Data exclusion

Any software packages that are not web-based, non-open source or free, or without any download or online version, are excluded. The software packages that appear in bold in Table 1 are examined in greater details.

5. Reviewed web-based experimental economics software packages (WEESPs)

In this section, we start with the summary assessment of the reviewed software packages based on the desirable features. Next, we describe our findings for each platform followed by comparison between them.

5.1. Summary assessment of the reviewed software packages

In the following tables, we provide a summary assessment of the software packages against the desirable features of Section 3 for both hosted software packages and non-hosted software packages, followed by a detailed discussion of individual software features.

Among the reviewed WEESPs, Veconlab, EconPort, EconPlay, LabSEE and classEx are hosted software packages, and SoPHIE, oTree, Willow, jsPsych and Coral are non-hosted software packages. The advantages of hosted software packages are that they do not require installation of any hardware and software and can be used immediately after registration. However, some potential disadvantages of them are that experimenters do not have any control and might not be able to create their own experiments using the packages. Maintenance hours of hosted services might not suit an individual's requirement and services can be stopped at any time if the hosting party decides not to continue or ask for payment. Therefore, the pros and cons of individual packages should be considered carefully before using a package.

Note that item 10, portability, stands out as the only desirable feature that is absent from all the reviewed software packages.

5.2. Discussion of selected software packages against the desirable features

In this section, we review the selected multi-purpose web-based experimental economics software packages or WEESPs. These can be further divided into software with Graphical User Interface (GUI), like oTree and SoPHIE, or hosted services like Veconlab, EconPort, LabSEE, EconPlay and classEx. There are also development frameworks such as jsPsych, Willow, and CORAL which offer experimenters more flexibility in building the GUI.

We assess the software packages on their features and not on performance or security. We leave out performance and security because factors like hardware, network and security configuration affect performance and security and we do not have the requisite information to take these factors into account.

5.2.1. SoPHIE

SoPHIE is an open source experimental economics and social sciences software developed in PHP using the Zend and Bootstrap frameworks. It can be integrated with any database supported by PHP. Its inventor claims SoPHIE was created to counter the complexity of using z-Tree. It is currently under active development and used by researchers in a number of universities. It is free to download and can be run on different operating systems. Users can also choose the service hosted by its commercial partner, sophielab.com, if greater support is required.

SoPHIE allows experimenters to design and create experiments directly within its Web interface. It provides six elements (termed as "steptypes" in SoPHIE) which include components like form, generic PHP, presentation, questionnaire, quiz, and sync. They can be used for creating any experiment. With a test run feature, it gives experimenters a preview of their designed experiment. Other than the installation of the software, experimenters manage the lifecycle of an experiment on the web interface. With the use of the Bootstrap framework as the user interface, it supports multiple devices by default, important for web-based software nowadays.

Unlike most reviewed software packages, SoPHIE has a built-in participant management system similar to ORSEE. A plugin for Amazon Mechanical Turk (AMT) integration can be purchased under sophielab.com commercial license. Data is stored in the database and can be used with major RDBMSs supported by PHP. With multi-tenancy support, it implements role-based access control for user and group management which allows more than one experimenter to use the system simultaneously.

Experimenters and subjects have different URLs for accessing the system. An admin interface is provided for monitoring the progress of experiments, as well as other functions like export and import of experiments created with SoPHIE for replication purposes. It has multi-lingual support in English, French, German, Italian, Spanish, and Chinese.

For group matching, experimenters start with the number of groups for experiments. In individual experiments without any

Table 2
Other details for the assessed economics software packages.

Software	Web-based	Open-source	Commercial	Hosted service	Multi-purpose	Specific purpose	MTurk Integration
The AEE Lab	✔		✔		✔		
BoXS	✔	✔				✔	
classEx	✔			✔	✔		✔
ConG		✔				✔	
CORAL	✔	✔			✔		
EconPort	✔			✔	✔		
jAuctions		✔				✔	
JessX		✔				✔	
jMarkets		✔				✔	
Moblab	✔		✔		✔		
Multistage		✔			✔		
Weblab	✔	✔			✔		
Zocalo		✔				✔	
FEELE lab	✔	✔		✔	✔		
LabSEE	✔			✔	✔		
Seaweed	✔	✔				✔	
SoPHIE	✔	✔			✔		✔
SoPHIE lab	✔		✔		✔		✔
Veconlab	✔			✔	✔		
oTree	✔	✔	✔		✔		✔
PEET		✔			✔		
Willow	✔	✔			✔		
Jars	✔	✔			✔		
JsPsych	✔	✔			✔		✔
nodeGame	✔	✔		✔	✔		
EconPlay	✔			✔	✔		
t-Tree	✔	✔				✔	

interaction, the number of groups created equals the number of participants. In experiments with participant interaction, the default is called "pre-generated grouping", which puts participants in the same group in each repetition. However, experimenters can choose to use random grouping to have participants group with different participants in each repetition. For MTurk integration and to reduce the drop-out rate, it creates a waiting room for grouping participants after all individual tasks are completed. Experimenters specify group size, individual waiting time until a timeout and the global waiting time until timeout (in seconds). Participants wait for a limited time to get matched with another participant. The third grouping option is to write one's own grouping mechanism by using the group API functionalities.

A major disadvantage of SoPHIE is the lack of ready-made experiments, the Ultimatum Game being the only example. However, SoPHIE's author confirmed that it had been used for simple individual decision tasks, basic economic games and human interaction, and bargaining, in particular, alternating-offer and continuous bargaining. The learning curve can be steep given the availability of only a few examples. Also, it does not have a bot function for automated testing. The current admin interface allows the experiment dashboard to be expanded to provide more insights into the experiments.

5.2.2. oTree

oTree is an open source web-based software using Python with Django and Bootstrap frameworks. According to one of its developers, it was developed because he and his collaborators could not find any software suitable for their requirements (Chen et al., 2015). It is under active development with much information available on its website. The website indicates universities from a number of countries are users, a point confirmed by one of its developers. The software is freely available for download and can be run on different operating systems.

The main strength of oTree is its flexibility for writing one's own experiments, with twenty-six ready-made experiments that can be used for learning and reuse purposes. Experiments are written in Python using any text editor. It is possible to create visual content using HighCharts, a JavaScript-based charting library. Experiments can also be run on another oTree installation by replicating source code files and settings.

oTree provides by default support for multiple devices similar to SoPHIE with the use of the Bootstrap framework. Regarding data persistence, its default option uses SQLite, a file-based relational DBMS (RDBMS), which is good for development purposes. However, common RDMSs like MySQL or PostgreSQL should be used for production.

oTree has an admin interface for experimenters to monitor live progress and perform actions such as moving to the next stage participants who have fallen behind in the experiment. It is written in modular design with an API available for further extension of functionalities. Automated testing using bots is another attractive feature. Further, oTree has multi-lingual support in different languages. It provides many options for group matching, like fixed matching where subjects are grouped sequentially, randomly, by arrival time, or by rank or score after the first round. Also, two more complex options are provided: a fixed number of groups with a divisible number of players and one with a non-divisible number of players.

A major disadvantage for users who prefer using GUI is that most settings for experiment configuration, e.g. for participants, groups and payoff settings, are done via the setting.py file or set up in python files. Also, session configuration via the dashboard is possible if the option is specified in settings.py file in advance. This could be enhanced by providing a Web interface for setting up those parameters.

More importantly, it has no multi-tenancy support which means only one experimenter can use each installation of the software with data isolation or multiple users without data isolation. Single tenancy with multiple users can allow users to see each other's data.

5.2.3. Veconlab

Veconlab is a hosted service for implementing economic experiments written in PHP and MySQL. New experiments have been added since we started the observation on its website, which confirms it is under active development. It allows anyone

to create an account for running online experiments. Instructions on how to set up and run an experiment are fully provided. It has recently reached the one million participant login mark. Visitors can try its online demo "Traveler's Dilemma" before registration.

The main strength of Veconlab is its high number of ready-made experiments, the highest amongst the software we reviewed. They cover different categories, namely auctions, bargaining, games, finance, markets, public goods, and information. All configurations are set up on the Web interface with a detailed explanation for each parameter. Data for experiments are stored in a MySQL database to ensure data persistence. It has very clear documentation on its experiments, which, as discussed earlier, is useful for classroom and research purposes. Its user interface is simple to follow, with guiding instructions on each screen. For group matching, two options are provided, fixed and random, but no other settings.

Veconlab's major limitation is its lack of capability for writing one's own experiment. As a workaround, experimenters can send an email to the maintainer and suggest new experiments, but there is no information on how such suggestions are received. Also, there is a maximum limit of 30 sessions before the data is recycled. For its user interface, it has plenty of room for improvement, as it has an early-day Web page style that could be improved with more effective use of color, images, and font styles. Moreover, without responsive support, its interface does not properly support mobile devices. No multi-lingual support is found on the interface. For subject management, experimenters need to use external software like ORSEE or Amazon Mechanical Turk, similar to most reviewed software packages. Data for experiments can only be displayed on the screen but not exported into other formats, another disadvantage. As a simple workaround, experimenters can copy data manually into Excel, but the lack of a data export interface is a significant shortfall. Its simple admin interface has plenty of room to expand, in particular by including an experiment dashboard.

5.2.4. EconPort

EconPort is another free hosted service that allows experimenters to create experiments online. It is a Web application written in J2EE (Java 2 Platform, Enterprise Edition) architecture connected to a database using JDBC (Java Database Connectivity) and developed by Georgia State University. No limitation of experiments and sessions are mentioned. However, it does not appear to be under active development, and the user community is unknown.

It is suitable for experimenters who want to run one of the five types of ready-made experiments available: normal form games, extensive form games, GARP (consumer utility maximization), marketlink, and one sided auctions. Documentation is provided to help experimenters run their experiments. The configuration is done partly on a Web page and partly using a Java Applet which requires Java WebStart to be installed.

Participants access experiments with an access code sent by the experimenters without need for registration. Its user interface is neat and tidy but still has an early days Web interface. It has multi-tenancy support, which allows multiple experimenters to use it at the same time. For group matching, it allows constant pairing, one-time random pairing, random round-robin pairing with a random player role assignment, random round-robin pairing while maintaining player roles and alternate player roles between matches.

EconPort has the lowest number of desirable features. As a closed source software, the community cannot contribute to the development of the software. Its major disadvantage is that it is not possible to write one's own experiments. A proper data export function should be implemented as it currently displays data on the screen. Experimenters need to copy and paste data from the screen and store them into a file.

For subject management, experimenters would have to use software like ORSEE or Amazon Mechanical Turk. No information on integration with Amazon Mechanical Turk can be found. With its limited number of ready-made experiments and inability to create one's own experiments, EconPort is not very useful if experimenters are not after the experiments provided. Also, it does not have multiple devices or multi-lingual support. The requirement of Java Web Start can create an issue, as not all browsers support Java by default nowadays.

5.2.5. EconPlay

EconPlay is another hosted service that allows experimenters to run experiments online. The platform is developed using .Net technology with data stored in a Microsoft SQL server database. Its author claims that it is under active development and has a number of regular users. It is freely available but requires registration.

EconPlay has a modern user interface compared with Veconlab and EconPort and is very easy to follow. There are eleven ready-made experiments, which include games, public goods, bargaining, auctions, and markets. Similar to Veconlab, all configurations are set up on the Web interface, together with detailed instructions. However, unlike Veconlab, there is an export function that allows experimenters to download experiment data into a CSV file.

EconPlay provides an admin interface for only a few administrative tasks. It has multi-lingual support in English and French for its ready-made experiments. With multi-tenancy support, it allows multiple experimenters to use the system at the same time, and the data is logically separated. Group matching is based on the number of players selected (from two to forty), and the number of players per group is worked out automatically. However, there are some inconsistencies, as some odd numbers of players only allow one group consisting of all players, while some assign a different number of players to different groups. There are no other options for pairing subjects.

A major limitation is a lack of capability to write one's own experiments. Also, as a non-open-source product, its development work cannot be shared by its community members. It has only one developer, so there is a risk of discontinuity. The user interface is modern, but the lack of responsive capability prevents EconPlay from reliably supporting multiple devices. For experimental data persistence, only up to 15 sessions can be stored before the data is recycled. Like all the reviewed software packages, it has an admin interface that can be expanded into an experiment dashboard. For subject management, experimenters need to organize this with third-party software like ORSEE or Amazon Mechanical Turk. No direct integration with Amazon Mechanical Turk is available.

5.2.6. LabSEE

LabSEE is another hosted service developed in PHP. It is freely available but requires registration. Its author started to create labSEE after being dissatisfied with z-Tree while completing his PhD at the University of Warsaw. It is under active development. The major users are the Faculty of Economics at Warsaw University and the Department of Psychology at Kozminski University in Poland. Support is mainly via email, but the author hopes to establish social media accounts as well as a discussion forum.

LabSEE's significant advantage is that experimenters can write their own experiments, and this sets it apart from the other hosted services reviewed. There is no limitation on the number of experiments or sessions. For writing experiments, JavaScript is the programming language, as the developer believes that it

is easy to learn. It supports multiple devices using the Bootstrap framework, much like SoPHIE and oTree. It is capable of creating any type of experiment and offers multi-lingual support using Unicode.

LabSEE allows the sharing of experiment code among users, which is an attractive feature. At the time of writing, there are seventeen codes to choose from, with fifteen in Polish and two in English. For integration, it has a beta Learning Interoperability Interface (LTI) extension that is available for connecting to software like Moodle. For group matching, subjects are assigned into groups sequentially, listing group names one after another. Alternatively, group matching can be implemented by setting the number of groups with an optional group name prefix. As a programming framework, experimenters can extend the default group matching function if they have a specific requirement that the default group matching method does not satisfy.

Like most of the reviewed software packages, LabSEE does not have subject management capability and needs to be managed with third-party software like ORSEE, HROOT or Amazon Mechanical Turk. No built-in integration with Amazon Mechanical Turk is implemented. The experimental dashboard needs to be expanded onto the current admin interface. Although the concept of shared experiments is attractive, its current downside is that they are mainly in Polish, which is likely to limit the uptake of the service outside Poland. The documentation is still being developed, and some functions, like data and message returns, need to be improved to reach present norms of quality. Also, the documentation on the website mixes English and Polish. The author acknowledges there is still work to be done.

5.2.7. classEx

classEx is a hosted service developed in PHP and MySQL and available for free. It was developed to allow lecturers carrying out experiments in the classroom, but it could be used for research purpose as well. The software is under active development at the University of Passau, Germany.

It has a mobile-first approach which runs on any mobile and non-mobile devices. It has ready-made standard experiments that allow the setup of multiple treatments, incentives, roles, groups, rounds and stages. Like LabSEE, classEx allows experimenters to write their own experiments, which can be useful for researchers as well.

Like LabSEE, sharing and comparing experiments by different experimenters is possible for knowledge sharing. Utilizing an AJAX protocol, it avoids reloading the page too often and give the user instant feedback. The AJAX feature can be turned off for direct interaction in the classroom which provides the instructor with an option when it is not required. classEx uses standard CSS3 technology to design its layout that works on both mobile and non-mobile devices. The number of ready-made games is growing as the wiki page shows only forty but fifty-six games can be found on the platform.

It has multi-lingual support and currently supports English, German and Spanish. The user interface allows the experimenter to switch between lecturer mode, overview mode and editing mode.

All experiment setup and experiment writing are being done via the user interface. Experimenters can export and download results in Excel spreadsheet xlsx file format.

On the interface of adding a parameter, text input boxes below a label do not have any information why the user is required to enter more than one which is something that can be improved. Also, after registration, the notification email does not contain any information about the initial password. There is a link on the login page which experimenters can request for the password. However, it is not obvious to new users as many platforms normally generate and send a default password. Overall, classEx is a very useful platform for classroom teaching.

5.2.8. jsPsych

jsPsych is an open source JavaScript-based development framework tailored to online behavioral experiments that have a defined structure. Its inventor chose JavaScript because it is the only native language that comes with any browser. jsPsych is under active development, with detailed documentation on its API and plugins. According to its inventor, jsPsych has been downloaded and used for many different types of experiments. Support is mainly provided through its Google Group forum.

As a development framework, JsPsych can be used to create any experiment, limited only by skill and access to resources. Its inventor claims the types of experiments it is capable of running include, but are not limited to, learning, perception, decision making, statistical learning, and memory, plus studies on psycholinguistics. It has a modular design with twenty-three plugins ranging across audio, HTML, survey, animation, and fetching user keyboard input to giving out instructions to subjects. It is possible to create one's own custom plugins which can extend its functionalities. To write experiments, experimenters can use any text editor and start with lines that import the jsPsych library. Experimenters can replicate experiments by sharing the source code with those who have jsPsych installed. Its main advantage is its flexibility, which gives experimenters substantial control over which experiments they can create.

A significant disadvantage is that it does not have any ready-made experiments available. A sample "Hello World" experiment is used to demonstrate the basic use of the library. By default, it stores experimental data in memory instead of storing in a database. This is a downside for data persistence, although it provides an example of how to use a server-side programming language, PHP, to use for data persistence. No built-in subject management is available, so experimenters need to use ORSEE or its plugin with Amazon Mechanical Turk, like most of the software reviewed. For more comprehensive integration with Amazon Mechanical Turk, a possible option is to use jsPsych together with PsiTurk (2014). That no graphical user interface is made available can be seen as providing some flexibility, but this requires experimenters to create it on their own. Its capability on multiple devices support is dependent on skills and expertise. Also, there is no multi-tenancy support, so it is not built for multiple experimenters using it on one installation. Moreover, neither admin interface nor experimental dashboard is provided, so experimenters need to create one if required.

In summary, the lack of many desirable features makes jsPsych an expensive option in time and effort required from experimenters. It is more suitable for experimenters who like to have more control and have the necessary skills and access to resources.

5.2.9. Willow

Willow is a web development framework for experimental economics developed in Python by the Department of Economics at George Mason University. It is free to download and use. Based on its Github and Sourceforge websites, the source code download has not been updated for seven years and appears not to be under active development. No information can be found on its documentation and discussion forum on whether it has an active community.

The framework allows a high degree of flexibility and can be used to create any experiment, subject to skill and access to the technical resources found in Python and Web front-end technologies like HTML and CSS. It allows experimenters to write their own experiments in Python with the API functions that are specifically tailored to creating economic experiments. Data is stored in CSV files which can be used by the common data analysis software. By sharing source codes with Willow, experimenters can replicate experiments. It provides a technical manual

which is easy to follow and would suit experimenters who have a background in Python programming or are interested in learning Python and web front-end technologies.

Its major limitation is that it is not under active development. Any bug fix or further enhancement is missing. Experimenters need to build the interface on their own which requires skill or access to resources, a common problem when working with development frameworks. Support for multiple devices is dependent on access to skills and resources as well. For subject management, experimenters need to use a third-party software, and nothing is mentioned in its documentation about integration with Amazon Mechanical Turk. No admin interface or experiment dashboard can be found, so that they would have to be implemented by experimenters from scratch.

Without multi-tenancy support, Willow is built for use by only one experimenter and for one installation at a time. No group matching function is found, but experimenters can create one of their own. Another disadvantage is that data is not stored in a database, although it can be imported into one with the CSV file it generates. Also, Willow provides only one simple example experiment, which does not give the option to reuse the existing code. It is suitable for experimenters who like to write experiments in Python using a framework instead of a complete system like oTree. However, Willow comes with the risk that the software may not be maintained in the future.

5.2.10. CORAL

CORAL is a development framework written in Java and developed at the School of Economics and Finance of the Queensland University of Technology. It was still in its early development stage when we assessed it. It is free to download and use. It was last updated in 2015 and is possibly no longer under active development. We could not find any information about its user community.

A major advantage of CORAL is that experimenters can use a CSV file to define the stages that represent the overall flow of an experiment. It includes properties like template filename for a stage, loop, repeat, condition, validation or not, and wait for input screen in chronological order of an experiment. It is very straightforward to understand. Experimenters define the overall setting of an experiment in a text file which specifies the stages CSV filename, the number of simultaneous clients, the database type (memory or database using a JDBC connection) and screen size. For the user interface, experimenters need to build the template files using a template engine, Apache Velocity. Its documentation mentions any common language can be used for its template engine due to Java's scripting module that includes z-Tree treatments. The logic of an experiment, like payouts and feedback to subjects, needs to be written in JavaScript and stored in separate files for modular design. For experiment testing, it comes with a robot.js file which can be used for automatic testing.

One major problem we found is its missing documentation for getting started on its website. However, a simple public good game example shows the basics of how to use the framework, which mitigates the problem. Missing documentation is a major issue for experimenters who wish to use this framework, as it requires reading its source code for a proper understanding. Like other development frameworks, it lacks a lot of desirable features and requires time and effort to build them from scratch. Also, as stated, it appears not to be under active development. We could not find an easy way to communicate with the author or with other users.

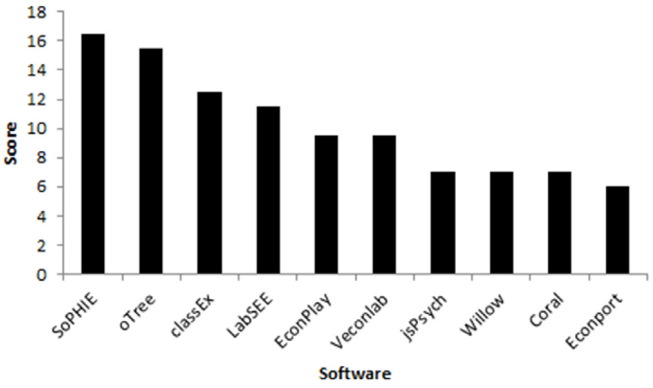

Fig. 1. Comparison of desired features among reviewed software packages.

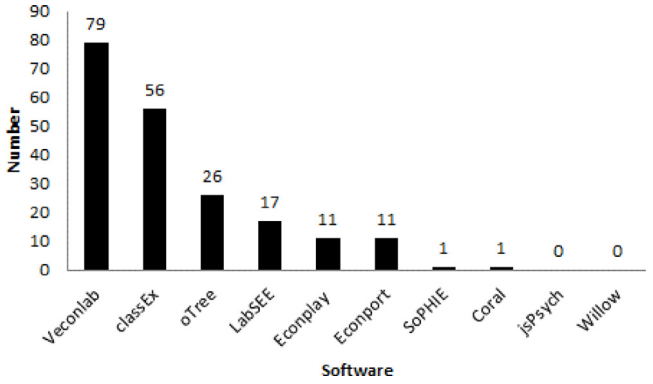

Fig. 2. No. of ready-made experiments of reviewed software packages.

5.3. Comparison of selected multi-purpose software packages against desired features

Using the data in Table 3 for non-hosted packages and Table 4 for hosted software packages, we calculated performance scores (Fig. 1). In the scoring method, we put 1 for every 'yes ✓' and 0.5 for 'partly yes ~'. The ones with the higher scores have a greater number of the desirable features discussed in Section 3. An implicit assumption is that all the desirable features are weighted equally. This is to avoid introducing our own preferences, leaving readers free to weight them as they wish.

Fig. 2 lists the number of ready-made experiments made available on each platform. classEx and Veconlab have the most ready-made experiments. The main difference between them is that experiments in classEx, but not in Veconlab, can be modified for creating new ones.

This review suggests that for experimenters who are looking for newer and actively developed software with GUI for writing their own experiments, oTree and SoPHIE should be considered. One major advantage of oTree over SoPHIE is the number of ready-made experiments that can be used and modified straightaway. They are both capable of creating many types of experiments and use the Bootstrap framework that allows experiments to be run on multiple devices. The choice between them might come down to one's preference for using Python or PHP as the programming language.

If experimenters are looking for software with plenty of ready-made experiments, Veconlab and classEx are the standout choices. They have the largest number of experiments that can

Table 3
Selected multi-purpose non-hosted software packages against desirable features.

Software Features	SoPHIE	oTree	Willow	jsPsych	Coral
1. Open-source	✓	✓	✓	✓	✓
2. Active community	✓	✓	✗	✓	unknown
3. Active development	✓	✓	✗	✓	✗
4. Multiple devices support	✓	✓	✗ – not by default	✗– not by default	✗ – not by default
5. Graphical user interface	✓	✓	✗	✗	✗
6. Data Stored in database	✓	✓	✗	✗ – not by default	✓
7. Data export	✓	✓	✓	No	unknown
8. Flexibility to write your experiment	✓	✓	✓	✓	✓
9. Replication	✓	✓	✓	✓	✓
10. Portability	✗	✗	✗	✗	✗
11. Subjects management	✓	✗	✗	✗	✗
12. Subject Payoff	✓	✓	✓	✗ - build your own	✗ – build your own
13. Experiment Dashboard	~ partly	~ partly	✗	✗	✗
14. API	✓	✓	✓	✓	✓
15. Modular design	✓	✓	✓	✓	✓
16. Bot testing	✗	✓	✗	✗	✓
17. Multi-lingual support	✓ – unicode	✓ – unicode	✗ - build your own	✗ -build your own	✗ - build your own
18. Multi-tenancy support	✓	✗	✗	✗	✗
19. Group Matching	✓	✓	✗	✗	✗
Total with ✓ and ~ partly	16.5	15.5	7	7	7

Table 4
Selected multi-purpose hosted software packages against desirable features.

Software Features	Veconlab	EconPort	EconPlay	LabSEE	classEx
1. open-source	✗	✗	✗	✗	✗
2. Active community	✓	unknown	✓	unknown	✓
3. Active development	✓	✗	✓	✓	✓
4. Multiple devices support	✗	✗	✗	✓	✓
5. Graphical user interface	✓	✓	✓	✓	✓
6. Data Stored in database	✓	✓	✓	✓	✓
7. Data export	✓	✓	✓	✓	✓
8. Flexibility to write your experiment	✗	✗	✗	✓	✓
9. Replication	✗	✗	✗	✓	✓
10. Portability	✗	✗	✗	✗	✗
11. Subjects management	✓	✗	✗	✗	✗
12. Subject Payoff	✓	✓	✓	✓	✓
13. Experiment Dashboard	~ -partly	✗	~ partly	~ partly	~ partly
14. API	✗	✗	✗	✗	✗
15. Modular design	unknown	unknown	unknown	unknown	unknown
16. Bot testing	✗	✗	✗	✗	✗
17. Multi-lingual support	none	none	✓ – 2 languages	✓ – unicode	✓ – 3 languages
18. Multi-tenancy support	✓	✓	✓	✓	✓
19. Group Matching	✓	✓	✓	✓	✓
Total with ✓ and ~ partly	9.5	6	9.5	11.5	12.5

be run immediately, with new types of experiments being added by the maintainers as well. They both are excellent choices for classroom teaching as well. The major downside for Veconlab is that the user interface is not mobile friendly while classEx can be a good choice if the use of mobile devices is a requirement. For researchers who are looking to write their own experiments on hosted services, Veconlab, Econlab, and Econplay would not be suitable. LabSEE and classEx are the only hosted services that allow writing one's own experiments.

For more flexibility, experimenters can consider a development framework like jsPsych which uses JavaScript to program experiments, and the interface can be built to the exact specifications that experimenters want. However, it also means greater cost and time, depending on skill and access to technical resources. jsPsych is under active development, and we suggest to use a server-side language together for data persistence. Willow has a detailed tutorial on how to use it but has not been updated for some time. CORAL is a new player in the domain and needs more time to mature.

All the reviewed software packages have different strengths and weaknesses. Portability is a major issue as no experiment created on one platform can be run on another. Most software packages have some administration interfaces that can be part of an experiment dashboard with more room for improvement. Automated testing is another area that needs to be focused on, as only two software packages have this feature built-in, oTree and CORAL.

6. Conclusion

Web technologies have generated positive impacts on many aspects of today's society. With their increasing influence, Web experiments are likely to play a more important role. They offer many advantages, like easier access to larger and more diversified sample populations, cost savings in administration, and attraction of participants with no restriction on location. Challenges like subject identity, multiple submissions and the like might make some experimenters hesitate about their viability. However, these issues can be overcome today with technological improvements, and the benefits will very likely outweigh the potential risks.

Web-based experimental economics software packages, or WEESPs, are created for both specific and multiple purpose experiments and have been developed in different programming languages. Reasons for their existence include dissatisfaction with existing software solutions, requirements not satisfied by existing web-based software packages, and preference for a particular programming language. WEESPs range from development frameworks that provide the most flexibility but demand the highest skill, to hosted services with lots of ready-made experiments and no programming skills required. No matter which one the experimenter chooses, an experiment running on one software cannot be run on another. Although Github or a backup allows old versions of software to be preserved, they may not work on future hardware, operating systems or browsers. We believe a markup language using XML to describe an experiment setting can be used to solve the portability issue. The process to develop one can be complex and requires enormous coordination effort within the Experimental Economics research community. However, markup language projects are not rare and have been taking place in many areas for solving portability challenges.

Web experiments offer experimenters opportunities that cannot be easily achieved in standard laboratory experiments. However, experimenters do not have to give up laboratory or field experiments, as WEESPs can run them as well. Some WEESPs can run even without a web server and Internet connection. More importantly, with increased portability, web-based experiments will allow ease of replication which, following on the tracks of physics and biology, will help experimental economics strengthen its scientific credentials (Camerer et al., 2016; Gertler et al., 2018). With the increasing importance of the Web and ample room for improvements, experimenters should consider adding web-based experiments to their toolkit.

Acknowledgments

The authors would like to thank the following researchers for providing information and assistance in learning more about their software:-

- Chris Wickens <chris@otree.org> for oTree
- Achim Hendriks <achim.hendriks@uni-osnabrueck.de> for SoPHIE
- Boun My Kene <bounmy@unistra.fr> for EconPlay
- Josh de Leeuw <jodeleeu@indiana.edu> for jsPsych
- Robert Borowski <info@labsee.com> for LabSEE

The authors are grateful to the Editor and the reviewers for providing detailed feedback on the paper. M S Iftekhar acknowledges funding support from the Australian Research Council's Discovery Early Career Researcher Awards grant (ARC DECRA grant number DE180101503).

Appendix A. Software packages information

The software packages information can be found in the following two tables. Table A.1 lists the source of the software packages with their URL, platform, online and free to use or not and license type. Table A.2 lists the detailed information of the software packages with Published/Working paper(s) found, type of experiments support (based on the website information), type of experiments conducted (from published/working papers found), a broad class of experiments, MTurk integration ready and documentation/support.

Appendix B. Programming languages, web frameworks and other technologies used in web-based experimental economics software

To develop web applications, developers can choose to use different programming languages, Web frameworks and other technologies. Programming language can be used to create dynamic content combined with HTML (Hyper Text Markup Language). Dynamic content can be account and subject management, treatment management, experiment creation and testing, record keeping, input validation, payoff calculation and other required functions. However, two developers can build the same web application with the same programming language in a very different structure like how they manage libraries/functions for database connection, session management, and dynamic content presentation and other common functions. Other developers who need to take on the project would have to spend time going through the code and understand how things are interconnected.

On the other hand, a Web framework is designed to make the development of web applications for a programming language in a standard way. It provides libraries for common Web application functions such as database connection, user and session management, templating engine for dynamic content presentation and other common functions. Web framework promotes code reusability and helps others to take on a project easier as the structure of application developed with a framework have a defined structure.

Table A.1
Source of software packages.

Software	URL	Platform	Online	Free	License type
AEE Lab	http://www.aton.com.au/index.html	Java (no download)	Yes	No	Commercial
BoXS	https://www.bonneconlab.uni-bonn.de/boxs	Java	Yes	Yes	GNU General Public
Cal Economics	http://www.res.otaru-uc.ac.jp/~uzawa/cal-economics/cal-ee.html The link is no longer available as the administrator disabled its access. See the information from the URL below. https://kaken.nii.ac.jp/grant/KAKENHI-PROJECT-10630001/	Windows software	No	Yes	Free to use
classEx	https://classEx.de	PHP	Yes	Yes	Free to use
Classroom Expernomics	https://w3.marietta.edu/~delemeeg/expernom.html	Possibly written in Perl CGI-bin based on the search function (last updated 2008)	Yes	Yes	None
ComLabGames	http://www.comlabgames.com/	Java	Yes	Yes	Free to download but no specific license type is mentioned
ConG	https://leeps.ucsc.edu/cong	Java	Yes	Yes	Under a version of the Simplified BSD License ("2-Clause BSD License" or "FreeBSD License") with the now-customary citation requirement if the software is used for academic publications
CORAL	https://code.google.com/p/coral-econ/	Java	Yes	Yes	Apache License 2.0
EconPlay	http://www.econplay.fr/	.Net (no download - cloud service)	Yes	Yes	Free to use
EconPort	http://www.econport.org/econport/request?page=web_home	Java Tomcat (no download - cloud service)	Yes	Yes	Free to use
EXCEN	http://expecon.gsu.edu/Software.html	Use EconPort (A9)	Yes	Yes	Free to use
FEELE	http://projects.exeter.ac.uk/feele/	PHP	Yes	Yes	Free to use
jars	https://github.com/s-plum/jars	Node.js application	Yes	Yes	Free to download but no specific license type is mentioned
jAuctions	http://people.hss.caltech.edu/~jkg/jAuctions.html No current link can be found.	Java (no download)	Yes	Yes	Nothing mentioned as no download is available
JessX - Java Experimental Simulated Stock Exchange	http://jessx.ec-lille.fr/index.php?page=overview	Java	Yes	Yes	GNU/GPL license
jMarkets	http://ssel.caltech.edu/software.html	Java	No	Yes	GNU General Public License
jsPsych	http://www.jspsych.org/	JavaScript	Yes	Yes	MIT
LabSEE	http://labsee.com	PHP (no download - hosted service)	Yes	Yes	Free registration
Marketscape	http://eeps6.caltech.edu/market-081029/	Downloadable CD. The demo site might indicate it is written in SSI/Perl	Yes	Yes	GNU General Public
MobLab	http://www.moblab.com/	JSP (cloud service for teaching and research)	Yes	No	Commercial
Multistage	http://ssel.caltech.edu/software.html	Java	No	Yes	Free to use
oTree	http://www.otree.org/	Python using Bootstrap framework	Yes	Yes	MIT open source license
Qualtrix	http://www.qualtrics.com/	PHP	Yes	Yes	Free to register
Regate	http://www.gate.cnrs.fr/perso/zeiliger/regate/regate.htm	Windows software running on Win XP only	Yes	Yes	Free to download but no specific license type is mentioned
Regate NG	http://www.gate.cnrs.fr/spip.php?article403	Python	Yes	Yes	MIT and GNU LGPL
Seaweed	http://sourceforge.net/projects/c-weed	PHP using the web framework CodeIgniter-1.7.2	Yes	Yes	MIT License
SoPHIE	http://www.sophie.uni-osnabrueck.de/	PHP using Zend and Bootstrap frameworks	Yes	Yes	BSD style license (based upon the "New BSD License") extended by the citeware clause
SoPHIE Labs (Commercial partner of SoPHIE)	http://www.sophielabs.com/	PHP (no download - cloud service)	Yes	No	Commercial
t-Tree	http://www.sweethall.net/auct-prog.cgi/ui	CGI-bin	Yes	Yes	To use the software for experiments, a user needs to sign a contract.

(*continued on next page*)

Table A.1 (*continued*).

Software	URL	Platform	Online	Free	License type
UAA Python Experimental Economics Laboratory - PEET	http://econlab.uaa.alaska.edu/Welcome.html	Python	No	Yes	GNU General Public License
Veconlab	http://veconlab.econ.virginia.edu/	PHP (no download - cloud service)	Yes	Yes	Free to use
WebLab	https://github.com/tomrutter/WebLab	Java (last updated five years ago)	Yes	Yes	Free to download but no specific license type is mentioned
Willow: Experiments in Python	http://econwillow.sourceforge.net/	Python	Yes	Yes	Free to download but no specific license type is mentioned
Zocalo	http://zocalo.sourceforge.net/	Java	No	Yes	Free to use

Apart from programming languages and Web frameworks, there are other technologies do not fit into the categorization of programming language and Web framework.

The following section provides a summary of each one of them that are used by the list of web-based experimental economics software that is found.

B.1. Programming languages

B.1.1. Java

Java is a programming language invented at Sun Microsystems and released in 1995. It has been taught by many Computer Science schools worldwide. The slogan of Java is "Write once, run anywhere" which illustrates benefits of the Java language. Java can be developed and run on any device as long as a Java virtual machine (JVM) is available on the device without any changes on the code. It has been used for network and Web development. The related Java development technologies are Java Applet for running application in the browser, J2EE Web framework and JDBC for database connection (Java, 2015).

B.1.2. PHP

PHP (recursive acronym for PHP: Hypertext Pre-processor) is a widely used and popular open source scripting language specifically created for Web development even though it can be used for general purpose. Its syntax is similar to Perl, Java and can be embedded within HTML. It has been widely used for websites and Web applications development and can be run on a number of operating systems like Windows, Mac and Linux. It provides a lot of functions like database connection, session management, file upload and many other libraries that make Web development a lot easier (reference here). Some well-known open-source products that are written in PHP are Content Management System like Drupal, Joomla and WordPress and Learning Management System like Moodle. The social media application, Facebook, is also written in PHP (PHP, 2015).

B.1.3. Python

Python is a widely used general purpose, high-level programming language. Its interpreters can be run on many platforms like Windows, Mac and Linux. It supports object-oriented, functional and procedural programming. It is used in scientific programming, big data analysis, Web programming and many other areas through its extensive libraries. Django is a very popular Web application framework written in Python which follows the model-view-controller (MVC) architectural pattern. Some of the well-known websites and Web applications that are written in Python are YouTube, Dropbox, Instagram and Pinterest. Google also uses Python for many applications including one of its Cloud services, PaaS (Platform as a Service), Google App Engine (Python, 2015).

B.1.4. JavaScript

JavaScript is a high level, object-oriented language that is built into every Web browsers. It is the only native language that comes with browsers without installation. It can be used to create interaction with users on the client side without the need to talk to the Web server. It is commonly used for client-side input validation, quizzes, poll etc. However, it can also be used as a server-side language like node.js and also for AJAX effect which to connect to the external data source to refresh the data without reloading the full page. jQuery and a number of similar JavaScript libraries are commonly used for JavaScript development (jQuery, 2015).

B.2. Web frameworks

B.2.1. Model-view-controller (MVC) framework

MVC framework is an architectural design pattern in software development. An application is split into three components which are model, view and controller. The model represents the data and the business logic of the application. The view is the presentation layer which is what users can see on the screen. The controller acts as the middleman between the model and the view and connects the appropriate model and view based on the user request. MVC is very popular in Web application development. It can be found on many Web programming languages like Java, PHP, Python and .Net framework (Leff and Rayfield, 2001).

B.2.2. Django framework

Django framework is one of the MVC frameworks written in Python. It eases the effort of developing a database-driven website with the ability to generate administrative interface without the need for coding. It is a high-level open-source Web framework that encourages rapid development and clean design. Some well-known websites that use Django framework are Instagram, Pinterest and Mozilla.

B.2.3. Zend framework

Zend framework is an open-source, object-oriented Web application MVC framework implemented in PHP 5. The principal sponsor of the Zend framework project is Zend Technologies, the company behind the development of PHP. Other significant contributors to the projects include Microsoft and Google. Its components can be used as a standalone library but they form a powerful and extensible Web application framework when combined. It offers MVC implementation and an abstract database class that eases connection to databases. Some other components are forms for HTML5 form rendering, validation and filtering and authentication for authentication and authorization with many common credential stores.

Table A.2
Detailed information on software packages.

Software	Published/Working paper(s) found	Type of experiments support (based on the website information)	Type of experiments conducted (from published/working papers found)	A broad class of experiments	MTurk integration ready	Documentation/Support
AEE Lab	Yes	Teaching Suite Experiments - Basic demonstration market types, Posted Offer, Double auction, over the counter market, MRET experiment (Public or Private Penalty and resale value), Ergon Energy MRET experiment (instant create, delayed create and enhanced DHW), AGL/Alinta merger experiment, MISO electricity, CSIRO SE Landscape experiment	Numerous papers can be found at http://www.aton.com.au/papers.html	Yes	No	The AEELab exists to support consultants, researchers and academics to conduct and use experimental economics research. They manage experimental economics lab facilities at a number of university campuses and can provide easy access to labs and student participants for persons wishing to run experiments. They also develop experiments to address specified research questions, and are a research facility in their own right, conducting experiments in respect to their research agenda. In particular, the AEE Lab can provide Lab management and control software and systems. Hosting of experiments in other labs using their lab-based experiment management systems. Development of experiments, drawing upon their software library of robust and running experiments. Delivery of experiments via their existing labs.
BoXS	Yes	A programming framework which should allow different types of experiments to run subject to the programming skill of experimenters	Public goods — how bundling public with private goods affects individuals' valuations for both goods (Frackenpohl & Pönitzsch 2013)	Yes	No	Online documentation, example programs, FAQ, mailing list, video tutorials
Cal Economics	No information is found.	CAL for Economics	No information is found.	No	No	Online documentation is found at http://www.res.otaru-uc.ac.jp/~uzawa/cal-economics/cal-ee.html
classEx	Yes	Games supported can be found at https://classEx.de/games	Teaching microeconomic principles with smartphones — lessons from classroom experiments with classEx Giamattei, M., and Llavador, H. (2017). Working paper. How Fragile Is Conditional Cooperation? A Field Experiment with Smartphones during the 2014 Soccer World Cup Graf Lambsdorff, J., Giamattei, M., Werner, K.(2017). Journal of Behavioral Decision Making 30(2):492.501. Emotion vs. Cognition — Experimental Evidence on Cooperation from the 2014 Soccer World Cup. Graf Lambsdorff, J., Giamattei, M., Werner, K., Schubert M.(2016). Working paper.	Yes	Unknown	Online documentation is found at https://classEx.de/documentation Google Forum is found at https://groups.google.com/forum/#!forum/classex

(*continued on next page*)

Table A.2 (*continued*).

Software	Published/Working paper(s) found	Type of experiments support (based on the website information)	Type of experiments conducted (from published/working papers found)	A broad class of experiments	MTurk integration ready	Documenta-tion/Support
Classroom Expernomics	No information is found.	No longer available — the website is maintained as an archive.	No information is found.	No	No	Nothing is available as the website is for archive purpose now.
ComLabGames	No information is found.	This module is for designing games to conduct experiments with human subjects over the Internet for learning about strategic behavior through personal experience and data analysis. It is suitable for use in high school and college courses, research laboratories for experimental economists and psychologists, as well as within strategic consulting groups. e.g. auction games, trading games, free-form games	No information is found.	No	No	Online documentation is found at http://www.comlabgames.com/free0.4/index.html
ConG	Yes	Prisoner's Dilemma, Hawk Dove, Public Goods Game (Voluntary Contribution Mechanism), Public Goods Game, Hotelling Spatial Competition Game and programming Skills Required to Extend ConG.	No information is found.	Yes	No	Demonstration, deployment and configuration documentation on the home page.
CORAL	Yes	Coral is a lightweight framework that aims to facilitate the design of economic experiments while being as flexible as possible. It means it can be used for creating different types of experiments subject to the programming skill of experimenters	No information is found.	Yes	No	Online documentation is found at https://code.google.com/p/coral-econ/wiki/GettingStarted
EconPlay	Yes	Games : (2 × 2 Games, Guessing Game), Public Good : (Voluntary Contribution), Bargaining : (Ultimatum Games, Trust Game, Gift Exchange Game), Auctions : (Private Value Auctions, Common Value Auctions), Market: (Double Auctions, Cournot oligopoly, Bertrand oligopoly)	- Nicolas Jacquemet, Stéphane Luchini, Antoine Malézieux, Jason Shogren, (Forthcoming), Is tax evasion a personality trait? An empirical evaluation of psychological determinants of << tax morale >>. Revue Économique - A. Baujard, F. Gavrel, H. Igersheim, J.-F. Laslier and I. Lebon << Individual Behavior under Evaluative Voting. A comparison between laboratory and In Situ experiments >>, in Blais, A., Laslier, J.-F. and Van der Straeten, K. (eds), Voting experiments, Heidelberg: Springer, pp. 257-270, 2016 - Jocelyn Groff and Anne Rozan, (2015) "Could two be worse than one? Individuals' investments in multiple public bads", Economics Bulletin, Volume 35, Issue 4, p.2166-2173	Yes — cannot create your own experiments	No	Online tutorial and FAQ are found at http://www.econplay.fr/aide.aspx

(*continued on next page*)

Table A.2 (*continued*).

Software	Published/Working paper(s) found	Type of experiments support (based on the website information)	Type of experiments conducted (from published/working papers found)	A broad class of experiments	MTurk integration ready	Documentation/Support
			- Romain Baeriswyl, Camille Cornand (2014), "Reducing overreaction to central banks disclosure: theory and experiment", Journal of the European Economic Association, 12(4): 1087-1126. - M. Lefebvre & A. Stenger (2016) Long-lasting effects of temporary incentives in good public games, BETA working paper n°2016-25. - d'Albis, H., Attanasi, G. and E. Thibault (2017), Ambiguous Survival Probabilities and Demand for Annuities: An Experimental Test through Charitable Giving, BETA WP, University of Strasbourg. - Lambert E.-A., Tisserand J.-C. (2016), Does the obligation to bargain make you fit the mold, WP du BETA 2016-37. - Garcia, S., Jacob, J., Lambert, E-A (2017), "Comparison of liability sharing rules for environmental damage: An experiment with different levels of solvency", BETA Working Paper n°2017-12. - Attanasi, G., Cox J. and V. Sadiraj, "Festival Games: Inebriated and Sober Altruists" - Jacob, J., Brunette, L., Eeckhoudt, L. (2017), "Prevent or Cure? Trading in the face of left-skewed binary lotteries?". - Brunette, M., Jacob, J. (2017), "Risk Aversion, Prudence and Temperance in gains and losses: Are we all schizophrenics?"			
EconPort		Normal Form Game, Extensive Form Game, GARP (consumer utility maximization), MarketLink (commodity double auction, asset double auction, posted offer, posted bid), One-Sided Auctions (Dutch, English, first- and second-price auctions)		Yes	No	
EXCEN		Use the EconPort website.	No information is found.	Yes	No	Use the EconPort website.

(*continued on next page*)

Table A.2 (*continued*).

Software	Published/Working paper(s) found	Type of experiments support (based on the website information)	Type of experiments conducted (from published/working papers found)	A broad class of experiments	MTurk integration ready	Documentation/Support
FEELE		American Call Option, Bertrand Competition, Currency Attack, Diamond Dyvbig Experiment, Hold-Up Problem, Insurance Market With Asymmetric Information, Kiyotaki Wright Hazlett Experiment, Lemon game, Monty Hall Paradox, Network Externalities, Price Discrimination, Team Draft, Warren Buffett.	When and what degree does communication help in oligopoly markets (Fonseca and Normann 2012).	Yes	No	Online documentation is found at http://projects.exeter.ac.uk/feele/LecturerStart.shtml
jars		No documentation is found.	No information is found.	No documentation	No	No information is found.
jAuctions		Auction programs that can be used for a wide variety of auction experiments.	No information is found.	Yes	No	No information is found.
JessX - Java Experimental Simulated Stock Exchange		Create a program allowing for the simulation of a financial market with realistic features like training, dot com bubble, several kinds of assets and divergence, stressful. - http://jessx.ec-lille.fr/index.php?page=experiment	No information is found.	Yes	No	A tutorial is found at http://jessx.ec-lille.fr/index.php?page=tutorial; teaching material is a broken link;
jMarkets	Yes	jMarkets is a Java-based web application which implements continuous double-sided electronic markets and supports multiple concurrent markets.	To explore how uninformed traders read information from transaction prices and order flow in financial markets with insiders (Bruguier et al. 2010).The reliability of online preference revelation using a series of controlled laboratory experiments (Chen et al. 2013).	No	No	Online documentation is found at http://jmarkets.ssel.caltech.edu:8000/jmarkets/wiki/InstallGuide and FAQ.
jsPsych	Yes	jsPsych is a JavaScript library for creating and running behavioral experiments in a web browser. The library provides a flexible framework for building a wide range of laboratory-like experiments that can be run online.	https://www.researchgate.net/publication/261221973_jsPsych_A_JavaScript_library_for_creating_behavioral_experiments_in_a_Web_browser	Yes	Yes - with The jsPsych.turk module	Online documentation can be found on the website and Google group is found at https://groups.google.com/forum/#!forum/jspsych
LabSEE	No information is found.	Nothing is mentioned.	The author could not supply the information.		No	No information is found.
Marketscape	No information is found	Experiments can include single or multiple markets, multiple goods and currencies, multiple types of subjects, varying economic conditions or incentives.	No information is found.	No — double auction trading	No	Online demo is found at http://eeps.caltech.edu/market-demo/, and online documentation is found at http://marketscape.caltech.edu/wiki/IndexofTutorials

(*continued on next page*)

B.2.4. Bootstrap framework

Bootstrap (2015) is a free and open source mobile first front-end framework. It is originally developed by Twitter and has since been very popular and widely used in many Web applications and websites. It has HTML and CSS based design templates for user interface components like form, button, navigation, typography and others. Web applications that are developed in the Bootstrap framework have the responsive capability that automatically adjusts the appearance according to the screen size. This makes it a very popular choice because Web applications

Table A.2 (*continued*).

Software	Published/Working paper(s) found	Type of experiments support (based on the website information)	Type of experiments conducted (from published/working papers found)	A broad class of experiments	MTurk integration ready	Documentation/Support
MobLab	No information is found	Auction Theory (11), Behavioral Economics (14), Finance (5),Game Theory (29), Industrial Organization (18), Intermediate Macroecon (6), Macroeconomics (9), Managerial Economics (27), Microeconomics (25), Negotiations (6), Political Economy (7), Principles of Economics (11).	No information is found	Yes	No	Online documentation is found at https://www.moblab.com/games/support/getting-started/
Multistage	Yes	It is designed initially to deal with a broad class of games subject to the experimenter's programming skill.	No information is found.	Yes	No	Online documentation is found at http://multistage.ssel.caltech.edu:8000/multistage
oTree	Yes	Public goods, trust game, beauty contest, survey, Prisoner's Dilemma, Ultimatum, Battle of the Sexes, Vickrey auction, Cournot competition,principal agent, dictator game, matching pennies, traveler's dilemma, bargaining game, common value auction, Stackelberg competition, Bertrand competition, stag hunt, real-effort transcription task, lemon market game.	Holzmeister, Felix, Pfurtscheller, Armin, 2016. oTree: The "bomb" risk elicitation task. Journal of Behavioral and Experimental Finance. 10, 105-108. Holzmeister, Felix. (2017). oTree: Ready-made apps for risk preference elicitation methods. Journal of Behavioral and Experimental Finance. 16, 33-38.	Yes	Yes	Documentation, FAQ, blog, demo and slides are available on the home page, and they are very comprehensive as well.
Qualtrix	No information is found.	Mainly use for creating surveys.	No information is found.	Yes - surveys only	No	Online training is found at http://www.qualtrics.com/university/researchsuite/ and also paid support.
Regate	Yes	The Regate software allows for implementing experimental economics experiments: bargaining, public goods, market, auctions. A simple script programming language has to be used to implement the experiments, while communication between the computers is based on Internet protocols (TCP/IP).	Coordination game (Jacquemet and Zylbersztejn 2013). Bubbles with overlapping generations (Deck 2014). Gender difference and competitive behavior (Gupta 2005).	Yes	No	Online documentation is found at http://www.gate.cnrs.fr/perso/zeiliger/regate/RegateManuel.htm
Regate NG	No information is found.	Documentation in French and Google translate does not work on the page.	No information is found.	Yes	No	No information is found.
Seaweed	Yes	Seaweed can create simple two-player economic games with symmetric actions and one decision per screen. Moreover, your game is ready to be played by randomly paired people on the Internet.	The author/developer evaluated the design interface by asking five economists to fix bugs in and augment a pre-existing game. The game engine was evaluated by posting a game of "Rock Paper Scissors" on MTurk	No - two-player economic games	No	No information is found.

(*continued on next page*)

and websites nowadays are expected to work on devices like PC desktop/laptop, Mac desktop/laptop and mobile devices like smartphones and tablets. Bootstrap makes the task a lot easier. Many Web applications use Bootstrap for their mobile themes and this includes Drupal, Joomla, WordPress, Moodle and many other applications.

Table A.2 (*continued*).

Software	Published/Working paper(s) found	Type of experiments support (based on the website information)	Type of experiments conducted (from published/working papers found)	A broad class of experiments	MTurk integration ready	Documentation/Support
SoPHIE	A working paper reference is found on the website but cannot find the paper.	Human Interaction Experiments with ultimatum as an example.	The author could not supply the information.	Yes	Yes	A number of technical documents are available at http://www.sophie.uni-osnabrueck.de/docs/; Google Group - Community Support - https://groups.google.com/forum/#!forum/sophie-community; The documentation and group support are ongoing.
SoPHIE Labs (Commercial partner of SoPHIE)	Same as SoPHIE.	Same as SoPHIE.	Same as SoPHIE.	Yes	Yes	Commercial support for SoPHIE - 1 ticket for 60EUR, 5 tickets for 275EUR, 10 tickets for 475EUR and 25 tickets for 950EUR. Support tickets cover extensive support inquiries like installation assistance, code checking, remote maintenance, etc. They guarantee the first response to an inquiry within 24 h. Our Support tickets do not include customization and development inquiries. A support ticket is valid for one year.
t-Tree	Yes — existed on the website but the link is broken now.	t-Tree (Tokyo Toolbox for Readymade Economic Experiments) is software for experimental economics that focuses on auctions and market design.	The author could not supply the information.	No - auctions and market design	No	Online documentation but the link is broken http://www.kazumori.net/tTree.pdf
UAA Python Experimental Economics Laboratory — PEET	No information is found.	Island experiment	No information is found.	Yes	No	Manual is found at http://econlab.uaa.alaska.edu/software/UAA-PEET%20basic%20instructions.txt - last updated: 30 Nov 2009

(*continued on next page*)

B.3. Other technologies

B.3.1. .Net framework

.Net (dot net) framework is developed by Microsoft. It is a collection of Web services. Applications developed in .Net are primarily run on the Microsoft Windows platform. Most commonly used programming languages for .Net development are C# and Visual Basic (VB) from Microsoft even though it is possible to develop in other programming languages on another platform with .Net framework installed.

B.3.2. Node.js

Node.js is an open source, a cross-platform runtime environment for creating server-side Web applications. Node.js applications are developed in JavaScript and can run on the node.js runtime across a number of platforms. It uses an event-driven and non-blocking I/O model which gives the advantages of being lightweight and efficient. Many large companies are reported as a user of Node.js on its website such as Microsoft, LinkedIn, Yahoo, Walmart and SAP.

B.3.3. Common gateway interface (CGI)

Common Gateway Interface (CGI) is a standard protocol for web servers to execute external programs/scripts running on a server that generates dynamic web pages. CGI scripts can be written in many programming languages like Perl, Python and PHP. It is an old technology that is used to generate dynamic Web content although it is still in use today. However, it is now slowly replaced by new technologies.

Table A.2 (*continued*).

Software	Published/Working paper(s) found	Type of experiments support (based on the website information)	Type of experiments conducted (from published/working papers found)	A broad class of experiments	MTurk integration ready	Documentation/Support
Veconlab	Yes	Auction: (Takeover Game, Common Value and Private Value Auctions, Multi-Round Auctions with Package Bidding Options, Emissions Permits), Bargaining: (Ultimatum, Principal/Agent, Reciprocity, and Trust Games), Decisions: (Bayes' Rule, Lottery Choice, Investment Game, Value Elicitation, Probability Matching, Search), Finance/Macro: (Asset Market, Macro Markets, Prediction Markets, Gains from Trade, Bank Runs), Games: (Attacker/Defender, Centipede, Coordination, Guessing, Matrix Games, Traveler's Dilemma, Two-Stage Extensive Form Game), Information: (Information Cascades, Lemons Market, Signaling/Poker Game, Statistical Discrimination),Markets: (Bertrand, Call Market, Cournot/Monopoly, Double Auction, Posted Offer, Supply Chain, Vertical Monopoly),	Conditional cooperation in a public goods game (Chaudhuri and Paichayontvijit 2006). Circadian effects on strategic reasoning - the time of day that decisions are made in the Beauty Contest(Dickinson and McElroy 2012)	Yes	No	Instructor's guide with the online demo is found at http://veconlab.econ.virginia.edu/guide.php and description for each available experiment/game.
		Public: (Common Pool Resource, Congestion/Entry, Public Goods, Rent Seeking, Volunteer's Dilemma, Voting, Water Externalities), Micro Principles: (Ten Experiments Configured for Introductory Classes: Trade, Supply, Demand, Costs, Monopoly, Market Failures, Simple Games), Principles: (Input Demand and Real Wages, Input Supply, Circular Flow, Gains from Trade, Inflation, Assets and Present Value), Surveys: (Questionnaire , Quiz Program)				
WebLab	No information is found.	No documentation is found.	No information is found.	No documentation	No	No information is found.
Willow: Experiments in Python	No information is found.	Nothing mentioned but given that it is a programming framework, it can be used to program any experiments assuming the experimenter has the skill.	No information is found.	Yes	No	Forum is found at http://sourceforge.net/p/econwillow/discussion/1094149/ - last activity: Aug 2014; Manual is found at http://econwillow.sourceforge.net/manual.html - last updated: 12 Nov 2010
Zocalo	Yes - http://zocalo.sourceforge.net/papers.html (a link with broken link documents)	A toolkit for building prediction markets, markets in securities that pay out depending on outcomes of future events.	Bubbles in asset markets with overlapping generations (Deck et al. 2014).	No - Prediction markets only	No	Online documentation is found at http://zocalo.sourceforge.net/javadoc/

Appendix C

The following table lists the software packages that are not required for citation and their URL.

Software	URL
Apache	https://httpd.apache.org
Drupal	https://www.linux.org
Java	https://www.java.com
PHP	http://www.php.net/
Python	https://www.python.org/
Ruby	https://www.ruby-lang.org/en/
MySQL	https://www.mysql.com/
PostgreSQL	https://www.postgresql.org/

References

Amazon Mechanical Turk, 2015. Overview – Amazon Mechanical Turk. https://requester.mturk.com/. (accessed 15.08.01).

Arechar, A.A., Gächter, S., Molleman, L., 2017. Conducting interactive experiments online. Exp. Econom. 1–33.

Bock, O., Baetge, I., Nicklisch, A., 2014. Hroot: Hamburg registration and organization online tool. Eur. Econ. Rev. 71, 117–120.

Bootstrap, 2015. Bootstrap – The most popular HTML, CSS, and JS library in the world. https://getbootstrap.com/. (accessed 15.08.01).

Camerer, C.F., et al., 2016. Evaluating replicability of laboratory experiments in economics. Science http://dx.doi.org/10.1126/science.aaf0918.

Chamberlin, E.H., 1948. An experimental imperfect market. J. Political Econ. 56 (2), 95–108.

Chen, D.L., Schonger, M., Wickens, C., 2015. oTree – An Open-Source Platform for Laboratory, Online, and Field Experiments. Working Paper, pp.1–16. http://www.otree.org/oTree.pdf.

Chilton, L.B., et al., 2009. Seaweed: a web application for designing economic games. In: HComp09 Proceedings of the ACM SIGKDD Workshop on Human Computation, pp. 34–35. http://portal.acm.org/citation.cfm?id=1600150.1600162.

Coelho, J., Valente, M.T., 2017. Why modern open source projects fail. In: Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering. ACM, pp. 186–196.

Croson, R., 2003. Why and How To Experiment: Methodologies from Experimental Economics. University of Illinois Law Review. https://illinoislawreview.org/wp-content/ilr-content/articles/2002/4/Croson.pdf.

Fischbacher, U., 2007. Z-tree: Zurich toolbox for ready-made economic experiments. Exp. Econom. 10 (2), 171–178.

Friedman, D., Sunder, S., 1994. Experimental Methods: A Primer for Economists. Cambridge University Press, New York.

Gardner, B.S., 2011. Responsive web design: Enriching the user experience. Sigma J.: Inside Digit. Ecosyst. 11 (1), 13–19.

Gertler, P., Galiani, S., Romero, M., 2018. How to make replication the norm. Nature 554, 417–419.

Giamattei, M., Lamsbdorff, J. Graf, 2019. classEx - An online tool for lab-in-the-field experiments with smartphones. J. Behav. Exp. Finance 29, 223–231. http://dx.doi.org/10.1016/j.jbef.2019.04.008.

Greiner, B., 2015. Subject pool recruitment procedures: organizing experiments with ORSEE. J. Econom. Sci. Assoc. 1 (1), 114–125.

Hendriks, A., 2012. SoPHIE - Software Platform for Human Interaction Experiments. University of Osnabrueck, Working Paper, 2012.

Hibbert, C., 2005. Zocalo: An Open-Source Platform for Deploying Prediction Markets. CommerceNet Labs, 21.

Ioannidis, J., Stanley, TD., Doucouliagos, H., 2017. 'The power of bias in economics research'. Econ. J. 127 (605), 236–265.

Jansen, M.A., Lee, A., Waring, T.M., 2014. Experimental platforms for behavioral experiments on social-ecological systems. Ecology Soc. 19 (4), 20.

jQuery, 2015. jQuery. https://jquery.com/. (accessed 15.08.01).

LabSEE, 2015. LabSEE: Experiments Surveys Test online. http://labsee.com. (accessed 15.08.01).

de Leeuw, J.R., 2015. Jspsych: A javascript library for creating behavioral experiments in a web browser. Behav. Res. Methods 47 (1), 1–12.

Leff, A., Rayfield, J.T., 2001. Web-application development using the model/view/controller design pattern. In: IEEE Enterprise Distributed Object Computing Conference. pp. 118–127.

Midha, V., Palvia, P., 2012. Factors affecting the success of open source software. J. Syst. Softw. 85 (4), 895–905.

Nielsen, J., 2012. Usability 101: Introduction to Usability. https://www.nngroup.com/articles/usability-101-introduction-to-usability/. (accessed 18.06.01).

Palan, S., 2015. GIMS—SOftware for asset market experiments. J. Behav. Exp. Finance 5, 1–14.

Palan, S., Schitter, C., 2018. Prolific.ac – a subject pool for online experiments. J. Behav. Exp. Finance 17, 22–27.

Paolacci, G., Chandler, J., Ipeirotis, P.G., 2010. Running experiments on amazon mechanical turk. Judgment Decis. Mak. 5 (5), 411–419.

Prolific.ac, 2018. Prolific: Bringing people together to power the world's research. http://www.prolific.ac/. (accessed 18.07.01).

psiTurk.org, 2014. pisTurl: crowdsource your research. https://psiturk.org/. (accessed 15.10.03).

Reips, U.D., 1996. Experimenting in the world wide web. In: Proceedings of the 26th Society for Computers in Psychology Conference (SCiP–96). Chicago, USA.

Reips, U.D., 2000. The web experiment method: Advantages, disadvantages, and solutions. In: Proceeding in Psychological Experiments on the Internet. pp. 89–117.

Reips, U.D., 2002. Standards for internet-based experimenting. Exp. Psychol. 49 (4), 243–256.

Robot Framework, 2017. Robot Framework. http://robotframework.org (accessed 17.08.12).

Schaffner, M., 2014. Programming for Experimental Economics: Introducing CORAL - a lightweight framework for experimental economic experiments. https://ideas.repec.org/p/qut/qubewp/wp016.html (accessed 15.10.06).

Seithe, M., 2012. Introducing the Bonn Experiment System (BoXS) (No. 01/2012). Bonn Econ Discussion Papers.

Selenium, 2015. Selenium – Web Browser Automation. http://www.seleniumhq.org/. (accessed 15.08.01).

Weel, J., McCabe, K., 2009. Willow: Experiments in Python. George Mason University. http://econwillow.sourceforge.net (accessed 15.10.06).

Wroblewski, L., 2011, RESS: Responsive Design + Server Side Components. http://www.lukew.com/ff/entry.asp?1392 (accessed 16.03.25).