HUMAN TECHNOLOGY

An Interdisciplinary Journal on Humans in ICT Environments ISSN: 1795-6889

www.humantechnology.jyu.fi Volume 11(2), November 2015, 100–122

OpenSR: AN OPEN-SOURCE STIMULUS–RESPONSE TESTING FRAMEWORK

Carolyn C. Matheus
Computer Science and Mathematics
Marist College
Poughkeepsie, NY, USA

Justin Svegliato
Computer Science and Mathematics
Marist College
Poughkeepsie, NY, USA

Abstract: *Stimulus–response (S–R) tests provide a unique way to acquire information about human perception by capturing automatic responses to stimuli and attentional processes. This paper presents OpenSR, a user-centered S–R testing framework providing a graphical user interface that can be used by researchers to customize, administer, and manage one type of S–R test, the implicit association test. OpenSR provides an extensible open-source Web-based framework that is platform independent and can be implemented on most computers using any operating system. In addition, it provides capabilities for automatically generating and assigning participant identifications, assigning participants to different condition groups, tracking responses, and facilitating collecting and exporting of data. The Web technologies and languages used in creating the OpenSR framework are discussed, namely, HTML5, CSS3, JavaScript, jQuery, Twitter Bootstrap, Python, and Django. OpenSR is available for free download.*

Keywords: *stimulus–response, implicit association, graphical user interface, open source framework, user-centered design.*

DOI: http://dx.doi.org/10.17011/ht/urn.201511113636

INTRODUCTION

Stimulus–response (S–R) tests provide a unique way to obtain information about human perception by capturing implicit (i.e., automatic) responses to stimuli and attentional processes. In computer-based S–R tests, participants are asked to respond to stimuli (i.e., audio or visual) using designated keys on the keyboard, a mouse, or some other input device. Stimuli can be generated and presented in a variety of ways, including through the use of software such as the proprietary MATLAB using PsychoToolbox extensions (Brainard, 1997; Kleiner, Brainard, & Pelli, 2007; Pelli, 1997) or the open-source PsychoPy (Peirce, 2007, 2009). Via these applications, response times can be recorded and responses can be marked as correct or incorrect.

Two major challenges in using S–R tests are the extensive code needed for developing the testing framework, a task often difficult for researchers in disciplines unrelated to computer science and information technology, and the steep cost of many proprietary software packages. In this paper, we discuss the importance of establishing a cooperative relationship between computer scientists and researchers in other disciplines, illustrating how the field of computer science (e.g., software development) can be leveraged for creating applications across interdisciplinary domains. We provide first an overview of S–R tests and then the process of creating and disseminating OpenSR, a user-friendly, open-source S–R framework that provides a graphical user interface (GUI) for configuring and administering an online implicit association test (IAT) that can implicitly assess a wide range of variables. The framework is extensible, which means functionalities can be added to the framework to meet the varied requirements of researchers (Fielding & Taylor, 2002). The primary advantages of OpenSR are its applicability to online data collection by researchers across domains, its GUI for configuring S–R tests, and its open-source accessibility (as opposed to costly proprietary software). OpenSR is available for free download.[1]

CURRENT PRACTICES USING S–R TESTS

S–R tests are used across multiple disciplines. A widely used paradigm investigates the latency of saccades (i.e., small jerky eye movements) for detecting peripheral stimuli during perceptual discrimination tasks (Deubel, 2008). This paradigm has been used to examine numerous variables, such as the effect of stimulus onset asynchrony (i.e., the time latency between the sequential presentation of stimuli) on saccade latencies and strategies (van Stockum, MacAskill, & Anderson, 2011); the ability for humans to perform voluntary saccade commands when presented with sudden visual stimuli at varying distances (Edelman & Xu, 2009) or to inhibit saccadic responses to visual distractors (Buonocore & McIntosh, 2008); and to investigate whether endogenous visual cues (e.g., an arrow, displayed at the center of the visual field, pointing with high probability to a possible target location)—expressed by voluntarily focusing attention on a specific location without making eye movements—can suppress exogenous cues that automatically capture attention (Koelewijn, Bronkhorst, & Theeuwes, 2009). In such experiments, eye movement is tracked and recorded as participants respond to a series of tasks in which stimuli are presented on a computer screen. For example, while the eyes are focused on a central cue, symbols briefly appear at the cued target location as well as at uncued nontarget locations. Reponses to the stimuli, measured either by external responses (e.g., pressing a key on the keyboard) or via eye-tracking software, are recorded and analyzed. Saccadic movements allow a detailed analysis of objects by bringing them onto the fovea. Located in the center of the macula region of the retina, the fovea allows sharp central vision and visual detail. When the eye is fixated on an object of interest, peripheral information is processed in order to decide where to look next. Thus, a saccade also represents a decision that must be made about the competition between a primary visual fixation and competing peripheral stimuli (Ludwig, Gilchrist, & McSorley, 2005). The impact of the competing stimuli, called the remote distractor effect, may influence how people determine where to look and what to focus on.

One purpose of such research is to examine the influence of top–down factors (e.g., attention, inhibitory strategies, or expectations) on visual responses (Bompas & Sumner, 2009; van Stockum et al., 2011). For example, saccadic movement testing is currently being used as one way to detect whether a person demonstrates signs of attention deficit hyperactivity disorder (ADHD). Some researchers suggest that symptoms of ADHD may be a result of abnormal motor responses to stimuli due to a failure to inhibit behavioral responses or focus on fixed objects (Barkley, 1997; Goto et al., 2010; Quay, 1997). Goto et al. (2010) compared voluntary control of saccades in children with and without ADHD diagnoses. Using an S–R research design framework in which visual targets were presented at a central fixation point (FP) as well as to the right and left of the FP, saccade latency and accuracy—measured through recordings of eye movements—were computed in a series of tasks.

During the tasks, participants were seated upright, were equipped with a chin rest, and faced a display that was 100 cm away from the eyes. In each trial, an FP appeared at the center of the screen and visual targets were presented randomly to the left or right of the FP at random intervals. Participants were instructed to fixate first on the FP and to perform horizontal saccades to the visual target as soon as possible. Some trials involved visually guided saccade tasks (VGSTs), and others involved memory guided saccade tasks (MGSTs). In the VGSTs, participants were instructed to fixate on the FP and to perform horizontal saccades to the visual target, which appeared for 50 ms, to the right or left in random intervals of 3 to 5 s. In the MGSTs, participants were instructed to fixate on the FP while the lateral target flashed in the randomly presented locations. Saccade toward this lateral target at this point was prohibited. When the central FP was switched off after the flashed target disappeared, participants were instructed to make a saccade to the remembered position of the flash. Saccades made to the flash while the FP was visible were judged as anticipatory errors. In a third block of trials, the antisaccade block, participants were instructed to look at a position approximately equidistant from the FP but in the opposite direction. Trials in which the participant first looked at the target and then looked in the opposite direction were judged as direction errors.

Each participant completed at least 25 trials in each block. Trials with confounding factors (e.g., head movement, poor fixation, misinterpretation of the task instructions, or saccade after reappearance of the target in the MGST) were excluded. Horizontal eye movements were measured by electrooculography. The target position and eye-movement signals were captured by a computer using a 1401-plus AD converter and Spike2 software. Results indicated that a higher percentage of spatial errors in VGST and MGST were observed in younger ADHD participants, who demonstrated response inhibitions dysfunction (i.e., an inability to inhibit behavioral responses to stimuli). However, the effect appears to change with age and prefrontal cortex development. Children, defined as less than 15 years of age, appear to have greater difficulty suppressing reflexive saccades. The ability to suppress reflex saccade movements matures in adolescence, leaving young children with ADHD especially vulnerable to environmental stimuli. Thus, preadolescent children with ADHD symptoms may be good candidates for early treatment and behavior inhibition training (Goto et al., 2010; Rueda et al., 2004).

In social psychology, the IAT is a computer-based S–R test that has been used to examine implicit attitudes about factors such as age, race, self-esteem, and gender (Greenwald, McGhee, & Schwartz, 1998; Greenwald, Poehlman, Uhlmann, & Banaji, 2009), as well as other intergroup biases and behaviors based on weight and sexuality (Lane, Banaji, Nosek, & Greenwald, 2007). In an IAT, participants make judgments about stimuli (e.g., text or images) presented on a computer screen in a series of timed tasks; response times are recorded in milliseconds. The speed and error rate in task completion can be used to measure underlying variables. The strength of automatic association between concepts can be inferred, for instance, by measuring the time it takes to sort words according to their category membership (Greenwald, Nosek, & Banaji, 2003). Thus, the IAT may reveal implicit biases that people unknowingly have or wish to keep hidden.

An IAT is divided into sections called blocks, which consist of a series of trials (e.g., in a block in which the test participant is asked to sort items into categories, the sorting of one item via the response mechanism represents one trial). The responses from each block (including the error rate and speed at which the participant sorted the stimuli) can be used to calculate a *D-score*, which represents the difference between the time it takes to sort words into stereotypically-congruent versus stereotypically-incongruent categories (Greenwald et al., 2003). D-scores can be used as dependent variables in subsequent data analyses. Participants generally find tasks easier to perform when they are congruent with existing thought processes. In other words, people generally perform tasks more efficiently when the tasks are compatible with their existing cognitive associations, whereas people generally slow down and make more mistakes when the tasks are incompatible with their existing cognitive associations (Rudman, Glick, & Phelan, 2008). Hence, the degree of difficulty should be evidenced by response time and error rate in the different tasks. Table 1 provides an overview of the series of blocks and trials typically used to create an IAT.

One application of the IAT sought to detect suicide intentions. A 2009 study, called the *Army Study to Assess Risk and Resilience in Servicemembers* (Tingley, 2013), investigated

Table 1. Sequence and Purpose of IAT Blocks.

Block	Function	Purpose
1	Practice	Introduce participants to the target-concept dimension
2	Practice	Introduce participants to the attribute dimension
3	Practice	Present combinations of congruent pairs of stimuli
4	Test	Present combinations of congruent pairs of stimuli
5	Practice	Introduce participants to the reverse position of the attribute dimension on the screen
6	Practice	Present combinations of incongruent pairs of stimuli
7	Test	Present combinations of incongruent pairs of stimuli

the mental health and suicide risk among United States Army soldiers. In this application of the IAT, the word *Life* appeared in the upper left corner of the computer screen and *Death* appeared in the upper right corner. Stimuli words associated with *Life* or *Death* appeared one at a time on the computer screen, and participants pressed one of two keys to sort words such as *alive* and *survive* with *Life* and words such as *funeral* and *die* with *Death*. The categories were then changed, with *Me* appearing in the upper left corner and *Not Me* appearing in the upper right corner. A new set of stimuli, such as *myself* and *mine* as well as *theirs* and *them*, were sorted similarly to the first block. The categories then appeared in groups: *Life or Me* and *Death* or *Not Me*. This forced participants to press the same key to associate *alive* and *survive* with *Life or Me*. Additionally, *funeral* and *die* were associated with *Death* and *Not Me*. In theory, greater speed with fewer errors[2] indicated a greater association between the participant and living. In the next block, the screen position of *Life* and *Death* were switched and participants sorted words into incongruent categories: *Death or Me* and *Life or Not Me*. Greater speed with fewer errors in this set of trials indicated a greater association between the participant and dying.

This application of the IAT, which deviates from more standard uses, attempts to deal with a growing issue for military personnel. However, the IAT is more often used to measure a wide variety of variables in the discipline of social psychology, such as attitudes about race, age, and gender. In these cases, the variables under study populate the attribute and target-concept dimensions. For example, White and White (2006) used the IAT to examine stereotypes about gender and occupation, proposing that certain occupations are stereotyped as masculine (e.g., engineer) or feminine (e.g., elementary school teacher). In their study, characteristics associated with occupations composed the target-concept dimension of the IAT, and gender—represented by stereotypical male and female proper names—composed the attribute dimension of the IAT.

The parameters for creating an IAT are similar regardless of what variables are being studied. Researchers simply populate the attribute and target-concept dimensions of the IAT using stimuli associated with the variables under study.

In summary, the IAT is a computer-based word-sorting test that measures the strength of automatic associations among concepts. The epistemological grounding of the IAT is that people will perform tasks more efficiently (i.e., more quickly and with fewer errors) when the associations are compatible with their existing thought processes. Thus, biases can be implied based on the difference in time and quantity of errors in completing stereotypically congruent versus incongruent tasks. For example, people who have a bias toward living will more quickly identify with concepts associated with being alive (e.g., breathing, survive), whereas people who may have suicidal tendencies (i.e., a bias toward death) will more quickly identify with concepts associated with being dead (e.g., funeral, die; see Tingley, 2013).

USER-CENTERED DESIGN

From the perspective of researchers, the primary problems with designing S–R studies that utilize an IAT are the costly proprietary software and extensive programming experience necessary to create and administer an IAT. That is, current options for creating an IAT do not adequately take into account the needs of the user (i.e., the researcher designing and administering the test). This is a common problem, as pointed out by Nielsen (2005, p. 2): “The vast majority of user interface design decisions are made based on the designer’s personal taste.

If we are lucky, the designer may venture a guess at users' needs, but that is as far as it goes." OpenSR provides one way to bridge that gap and enhance current practices through a framework that provides researchers the ability to configure, administer, and analyze an IAT.

The primary goal of this project was to provide researchers with an easy way to create an IAT by providing a framework that specifically meets the design needs of researchers across a broad range of disciplines. Thus, the goal was to provide a tool that is both easy to use and useful. When systems are perceived as being difficult to use, users may avoid them. On the other hand, people are more likely to use systems that they perceive as easy to use. Several models have been proposed to explain decision-making processes for accepting and using new technology, such as the technology acceptance model (TAM; Davis, 1989), the information system (IS) success model (DeLone & McLean, 1992, 2003), and the unified theory of acceptance and use of technology (Venkatesh, Morris, Davis, & Davis, 2003).

The TAM (Davis, 1989), developed to help explain how users accept new technology, examines actual or intended use of a system. More specifically, the TAM suggests that people are more likely to use a new system if they perceive it as being easy to use (A. B. Matheus, Matheus, & Neely, 2014). A revised version of the TAM—the TAM3 (Pearlson & Saunders, 2013)—also evaluates factors that influence how people perceive the ease of use and usefulness of a technology (e.g., individual differences, system characteristics, and social influence), which may influence intentions to use a system. Similarly, the IS success model (DeLone & McLean, 1992, 2003) examines factors related to user satisfaction and intentions to use a system. These researchers identified system quality, information quality, use, user satisfaction, individual impact, and organizational impact as significant factors.

The unified theory of acceptance and use of technology (UTAUT) is another model that examines intentions to use a system (Venkatesh et al., 2003). The model was tested in a study in which users completed a satisfaction survey regarding their experience with a new, mandated software system (Venkatesh et al., 2003). Four constructs related to system usage were identified in the study: performance expectancy (i.e., perceived usefulness), effort expectancy (i.e., perceived ease of use), attitude (i.e., user's affect), and social influence (i.e., whether or not the user was influenced by other people). A related study illustrates the application of these models. Using structural equation modeling, A. B. Matheus et al. (2014) used elements from the TAM, the IS success model, and the UTAUT as a foundation for examining the impact of four constructs—perceived information quality, perceived design quality, perceived usefulness, and perceived ease of use—on performance in a Web-based task. The goal was to determine what factors influence the successful use of the World Wide Web, which represented the use of a system. The results indicated a number of findings related to system usage, including (a) when users perceive that a Web site has good design quality (e.g., is well organized with links that are easy to find and use), they are more likely to believe the information on the Web site is valuable and accurate as well as easy to use when finding a specific Web site; (b) users who find Web sites easy to use (e.g., displays and labels are easy to read) perceive that Web site to be more useful for completing tasks; (c) users who perceive a Web site to be useful (e.g., able to adequately and effectively meet task needs) are more successful overall in completing tasks; and (d) users who perceive information on a Web site to be of high quality are more likely to believe that the Web site can effectively meet their needs in completing a task.

Taken together, people are more likely to use a system they perceive as being easy to use and useful (e.g., it adequately meets their needs). Many people have come to expect an interactive, user-friendly experience with technology, with programs that can be easily configured. A key component of usability is a human-centered design interface and easy to use GUI.

THE OpenSR FRAMEWORK

The framework of a computer-based S–R instrument provides a platform for expanding the nature of the variables that can be examined, and Web-based applications provide an avenue for expanding the demographic of potential participants. However, some programs lack usability and accessibility for researchers who are not skilled programmers or who simply cannot afford expensive licensing fees. Other programs are open-source (e.g., free); however, many lack a GUI and are only configurable via a command-line program. Table 2 provides a comparison of available software packages that support S–R reaction-time experiments.

Table 2. Software for S–R Experiments.

Software	License	Operating System	GUI	Browser Based	Stimuli Options	Input/Output Options
Affect 4.0	Open	W	✓	X	I, A	K, Mo
DirectRT	Proprietary	W	✓	X	T, I, A, V	K, Mo, Mi, J
E-Prime	Proprietary	W	✓	X	I, A, V	K, Mo, J
Expyriment	Open	W, M, L	X	X	T, I, A, V	K, Mo, RB
Inquisit	Proprietary	W, M	✓	✓	T, I, A, V	K, Mo, Mi, TS, RB
OpenSesame	Open	W, L	✓	X	T, I, A	K, Mo
OpenSR	Open	W, M, L	✓	✓	T, I, V	K, Mo
Paradigm	Proprietary	W, iOS	✓	X	T, I, A, V	K, Mo, Mi, J, RB
Presentation	Proprietary	W	✓	X	I, A, V, VG	K, Mo, J
PsychoPy	Open	W, M, L	✓	X	T, I, A, V, RD	K, Mo, Mi, RB, J
PsychToolbox	Open*	W, M, L	X	X	T, RD, S, VG	K, Mo
Psykinematix	Proprietary	M	✓	X	T, I, A, V, S, RD	K, Mo, Mi, J
SuperLab	Proprietary	W, M	✓	X	T, I, A, V	K, Mo, Mi, TS
Vision Egg	Open	W, M, L	X	X	T, I, V, RD	K, Mo, J, RD

Note. GUI = Graphical User Interface. **Operating Systems:** W = Microsoft Windows, M = Macintosh, L = Linux. **Stimuli Options:** A = Audio, I = Images, RD = Random Dots, S = Shapes, T = Text, V = Video, and VG = 3D Visual Graphics. **Input/Output Options:** J = Joystick, K = Keyboard, Mi = Microphone, Mo= Mouse, RB = Response Boxes, RD = Random Dots, and TS = Touch Screen.
*Requires MATLAB.

OpenSR provides several advantages for creating and administering an IAT over other software. First, it is open-source and, thus, a good option for researchers who lack funding to purchase costly licenses. Second, OpenSR can run on all major computer software platforms (i.e., Windows, Macintosh, and Linux). Third, although OpenSR is not the only framework that provides a user-friendly GUI, it is one of few IAT frameworks that is browser-based and does not require researchers to download and install plug-ins. Another browser-based option, Inquisit, is not open-source, charging fees to license. OpenSR incorporates the primary means of input for a standard IAT (i.e., keyboard and/or mouse). If researchers desire additional features beyond those provided by the current version of OpenSR, then the open code base allows them the ability to work with programmers to extend the functionality.

One of the biggest challenges of IAT S–R tests is developing the extensive code needed to customize and administer the test, which can be difficult without programming experience (Peirce, 2007, 2009). Thus, a user-friendly GUI is of great benefit for researchers from different disciplines who may not have in-depth programming knowledge in that they can easily customize and administer S–R tests (C. C. Matheus & Svegliato, 2013). GUI features, which vary by software, provide user-friendly capabilities for creating an S–R test and specifying the parameters of the test. For example, some proprietary software, such as Paradigm and E-Prime, include a GUI with mouse click-drag-and-drop functionality for specifying testing features, such as adding and customizing instruction sets and trial blocks (e.g., configuring stimuli types and trial identifiers, setting the number of blocks and the number of trials within each block, and specifying inter-trial intervals).

Similar to OpenSR, some S–R software, such as Presentation, Psykinematix, and OpenSesame, have GUIs with simple test-design capabilities through which test creators can navigate the menus of options to design and customize tests and procedures and to select options for exporting data. Test specifications are customized by using a mouse to click through forms to enter preferences for instruction sets, stimuli, and blocks and trials. This type of interface is customary for many systems and familiar to researchers. Using standard interface objects (e.g., list boxes, drop-down boxes, and pull-down menus) provides navigation consistency and is intended to reduce the learning time required by users of a new system (Dennis, Wixom, & Tegarden, 2012).

The current paper addresses the aforementioned challenges by presenting OpenSR, an S–R testing framework with a user-friendly GUI that can be used for the creation and administration of an IAT, including the ability to easily customize the testing parameters. Features of OpenSR include (a) a Web-based framework, enabling researchers to administer an IAT via the Internet to participants in any geographical location who have Internet access; (b) an open-source user-friendly framework that is platform independent and can be implemented on most devices (e.g., personal computers and tablets) using various operating systems (e.g., Windows, OSX, Linux); (c) scripting capabilities for automatically generating and assigning participant identifications and tracking, for assigning test participants to different condition groups, and for automatically forwarding test participants to external Web sites, if needed; (d) procedures for collecting and exporting data; and (e) procedures for customizing the program for use in a variety of IAT studies. The remainder of this paper describes the background, development, rationale, features, and applications of OpenSR, as well as the potential to extend to applications beyond the IAT. This information is explained by looking at three distinct facets of configuring, presenting, and analyzing an IAT: the test configuration phase (which includes all aspects of the researcher's test design and implementation prior to the test presentation phase), the test presentation phase (which is completed by the test participants), and the analysis phase (which encompasses all aspects of data storage and the researcher's extraction and viewing of data for analysis).

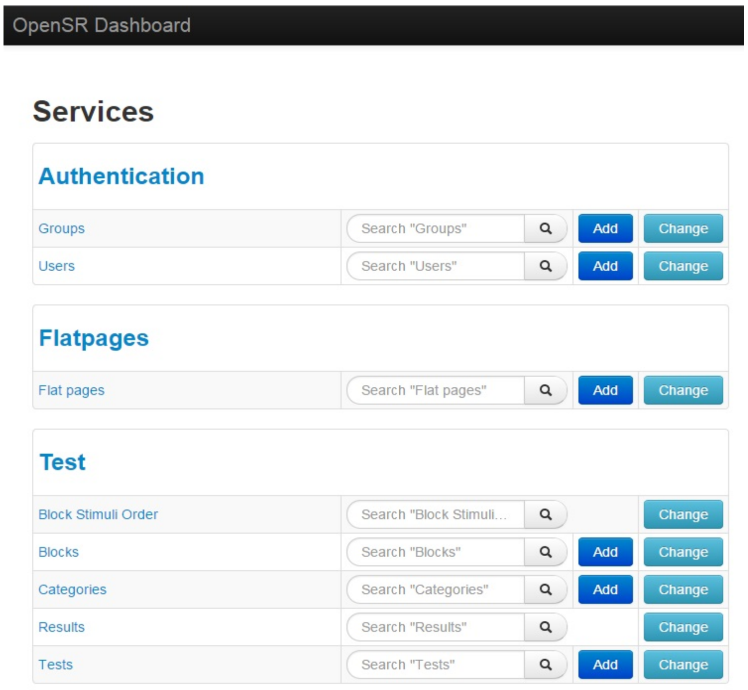
Test Configuration Phase: The OpenSR Dashboard

Every IAT comprises multiple steps and decisions related to the experiment's research goal(s), design, and implementation. Most of these steps and decisions take place within the test configuration phase of the OpenSR dashboard. The researcher's specifications within this phase will have implications for both the test presentation phase and analysis phase, described below, but are intended to facilitate both. OpenSR's test configuration phase includes a variety of tools to assist the researcher in test design, all found within the administrative dashboard.

A significant contribution of OpenSR is its user-friendly interface for creating S–R tests that provides researchers a functional tool for creating and administering an IAT, including capabilities for creating and configuring the presentation of the test (i.e., collecting input via forms to render the final presentation of the IAT) and collecting, storing, and retrieving data from the test. Researchers wishing to use OpenSR are provided a simple dashboard offering three primary services: (a) authentication, (b) flat page, and (c) test. Each service comprises several modules, and each module offers a set of associated functionalities (see Figure 1).

Authentication Service

The authentication service allows researchers to specify authorization (i.e., the list of researchers who can access the administrative dashboard) and entitlements (i.e., what actions



OpenSR Dashboard

Services

Authentication

Groups	Search "Groups"	Add	Change
Users	Search "Users"	Add	Change

Flatpages

Flat pages	Search "Flat pages"	Add	Change

Test

Block Stimuli Order	Search "Block Stimuli…		Change
Blocks	Search "Blocks"	Add	Change
Categories	Search "Categories"	Add	Change
Results	Search "Results"		Change
Tests	Search "Tests"	Add	Change

Figure 1. The OpenSR administrative dashboard.

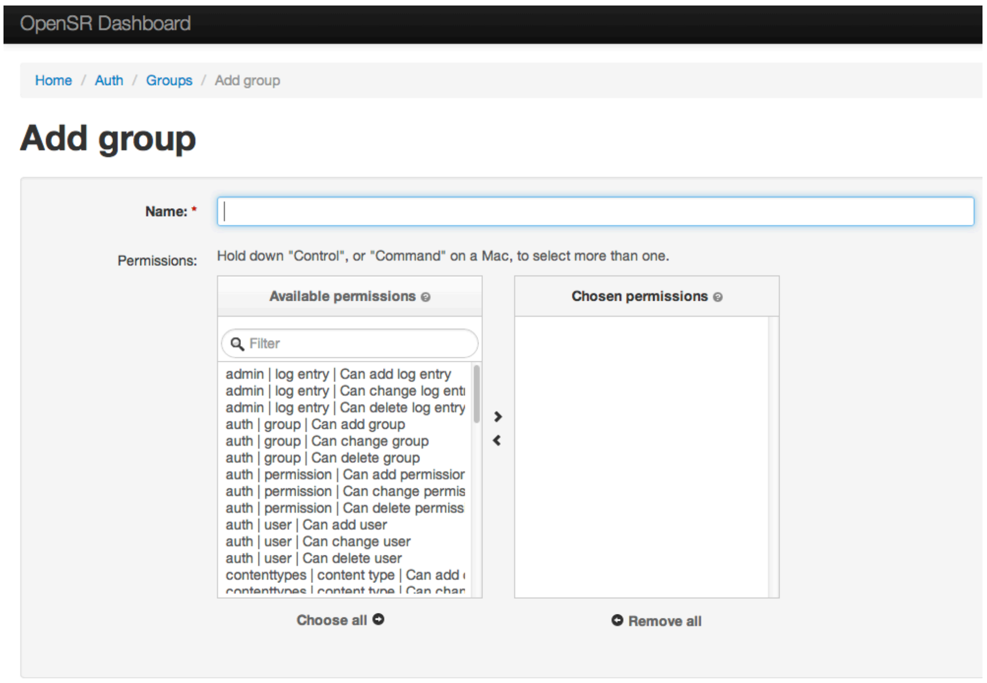
those researchers can take within the administrative dashboard). The functionality of this service is contained within two modules: user management and group management. The user management module provides researchers with the ability to add a test manager by specifying a username and password. The group management module allows researchers to add a test manager to a specific group with predefined permissions. For instance, a group of researchers (i.e., test managers in this case) might be given the ability to administer a test and view test results but not modify the test itself (see Figure 2).

Flat Page

The flat page service provides researchers with the ability to manage Web pages that appear on a test (i.e., one component of the OpenSR interface that test participants can directly see and interact with). Researchers can create and manage the introduction, informed consent, experimental and control groups, and confirmation pages via this module. These pages can then be associated with a test, as is discussed in the next section. By using a WYSIWYG (What You See Is What You Get) editor, researchers can style and design Web pages to their preferences and goals. In order to create a flat page, the researcher must specify a URL (i.e., the page's unique Web address), a title, and the content.

Test

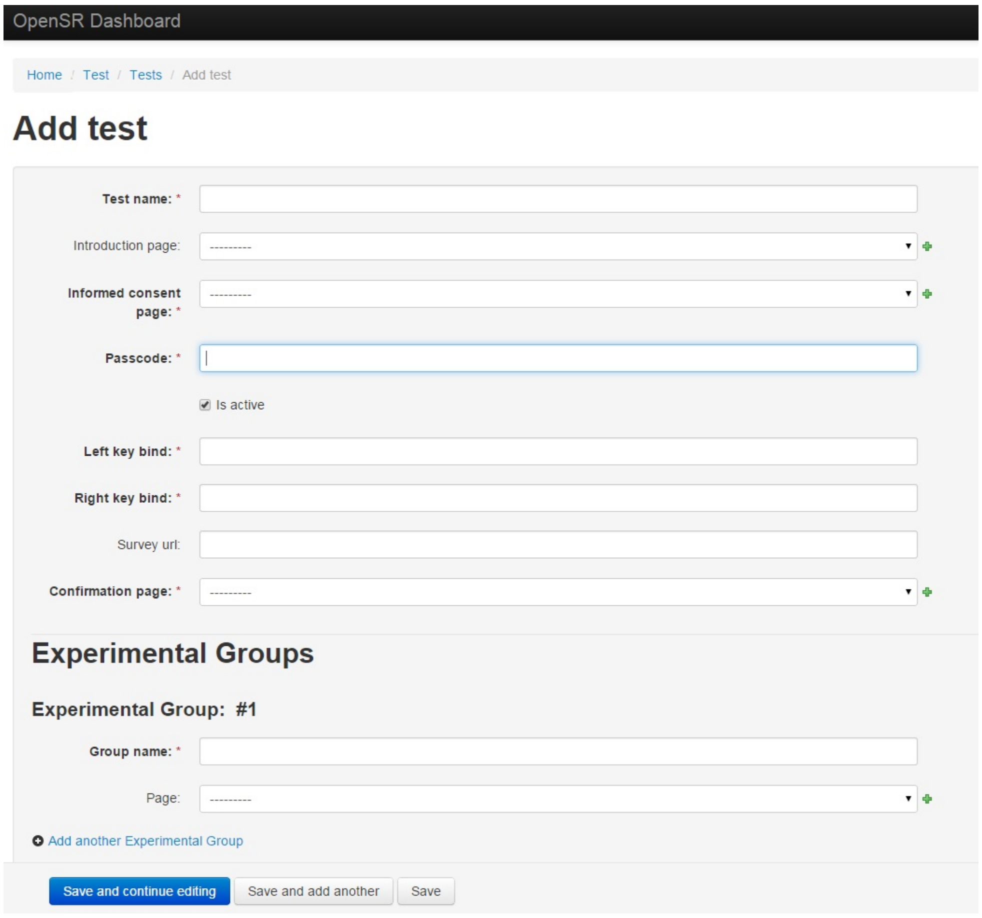
In the test service, researchers can create and manage all aspects of a test (see Figure 3). This functionality is contained within four modules, shown in Figure 1: Test, block, category, and block stimuli order. The test module allows researchers the ability to create, delete, or modify



Hold down "Control", or "Command" on a Mac, to select more than one.

admin | log entry | Can add log entry
admin | log entry | Can change log ent
admin | log entry | Can delete log entry
auth | group | Can add group
auth | group | Can change group
auth | group | Can delete group
auth | permission | Can add permissior
auth | permission | Can change permis
auth | permission | Can delete permiss
auth | user | Can add user
auth | user | Can change user
auth | user | Can delete user
contenttypes | content type | Can add

Figure 2. The OpenSR add group form to manage user permissions.


OpenSR Dashboard

Home / Test / Tests / Add test

Add test

Test name: *	
Introduction page:	---------
Informed consent page: *	---------
Passcode: *	
	☑ Is active
Left key bind: *	
Right key bind: *	
Survey url:	
Confirmation page: *	---------

Experimental Groups

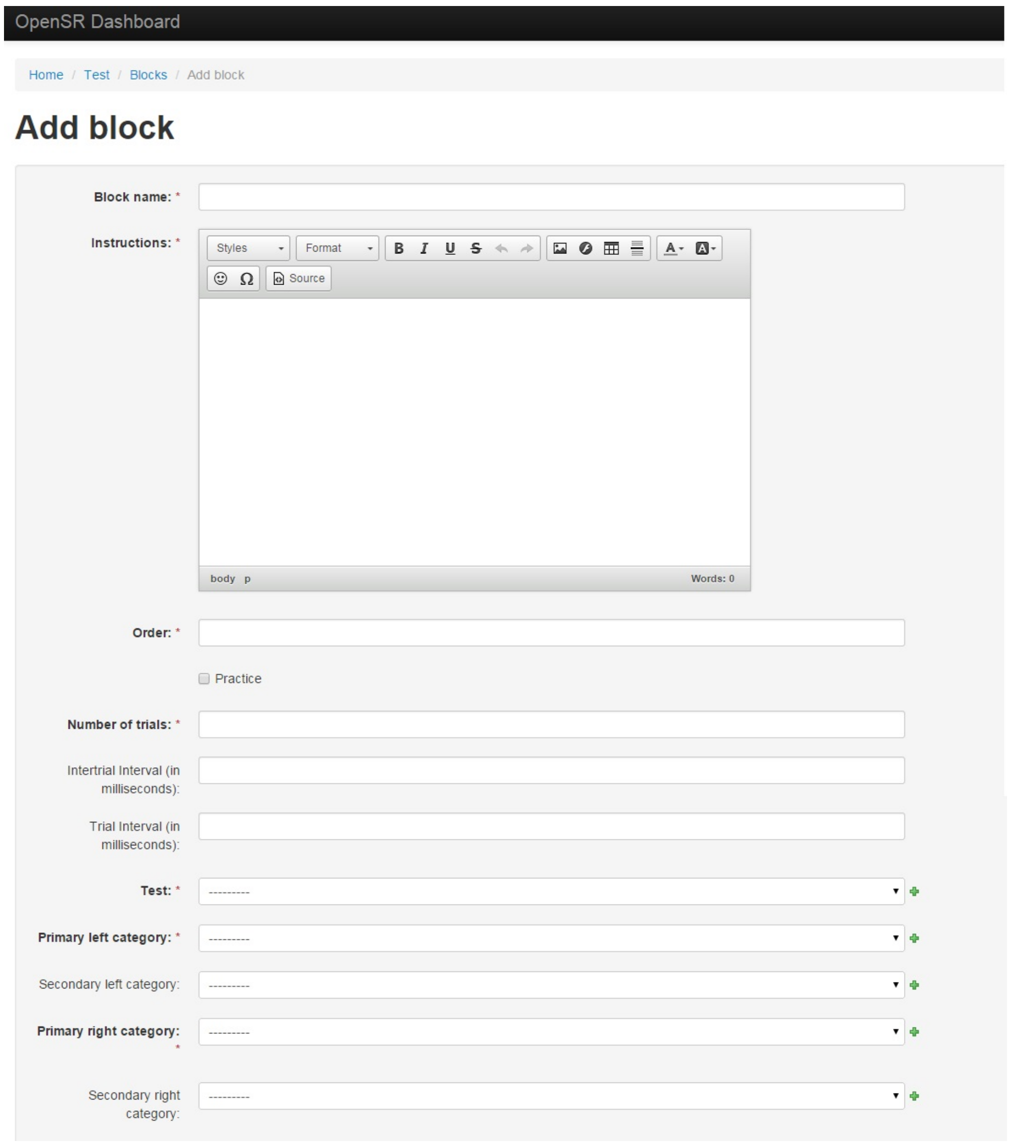
Experimental Group: #1

Group name: *	
Page:	---------

Figure 3. The OpenSR add test form to configure a new test.

components of tests. For example, a researcher can associate the introduction, informed consent, experimental and control group, and confirmation pages with a test, set a password, set key binds (e.g., specifying keyboard options for responses or mouse click), add a link to an external Web page or survey, and create a confirmation page. If a link to an external survey (e.g., SurveyMonkey or other Web survey application) is provided when setting up the test, participants will be automatically directed to the external survey Web site. Upon completion of the survey, participants will be automatically redirected back to OpenSR. If it is desired, survey order effects can also be configured at this point. For example, test participants with even identifiers can be directed to one version of a survey, whereas participants with odd identifiers can be directed to a different survey version.

The block module allows researchers to customize many aspects of the test block content. For example, the content for test instructions and test presentation (e.g., font style, color, and other formatting options) can be customized. Researchers can also specify stimuli to be presented in a particular order, designate practice versus testing blocks, set the number of trials per block, designate the intertrial interval (i.e., the specified time between the presentation of each stimulus) and the trial interval (i.e., the specified length of time each stimulus is presented). In addition, researchers can link blocks to a designated test and define the screen presentation and location of stimuli, designated as primary and secondary, left and right categories (see Figure 4).


OpenSR Dashboard

Home / Test / Blocks / Add block

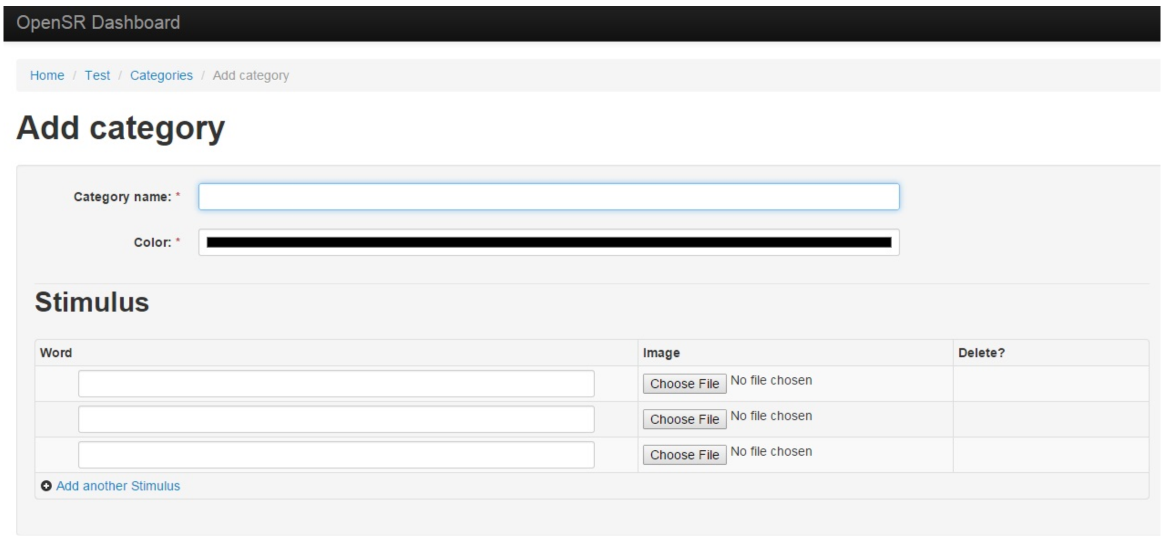
Add block

Block name: *	
Instructions: *	
Order: *	
	Practice
Number of trials: *	
Intertrial Interval (in milliseconds):	
Trial Interval (in milliseconds):	
Test: *	---------
Primary left category: *	---------
Secondary left category:	---------
Primary right category: *	---------
Secondary right category:	---------

Figure 4. The OpenSR add block form to customize content and process of test blocks.

The category module allows researchers to manage the categories of stimuli within each block. Researchers can specify the name of the category, choose text color, and designate the text and/or images for stimuli (see Figure 5). A category is not restricted to one particular type of stimuli.

The block stimuli order module allows a researcher to present the stimuli in a particular order, if desired, using drag-and-drop functionality. Alternatively, the stimuli can be presented in random order (see Figure 6) using Math.random(), a built-in JavaScript function for generating random numbers. Math.random() is a pseudorandom number generator and, as a result, the probability of generating each number is not equal but approximately the same. A pseudorandom number generator is used because true random number generators cannot exist on computers insofar as computers are deterministic machines. For the purposes of selecting stimuli during a participant's test, however, a pseudorandom number generator is sufficient because the user cannot perceive the slight unevenness of the probability distribution across every number.


OpenSR Dashboard

Home / Test / Categories / Add category

Add category

Category name: *

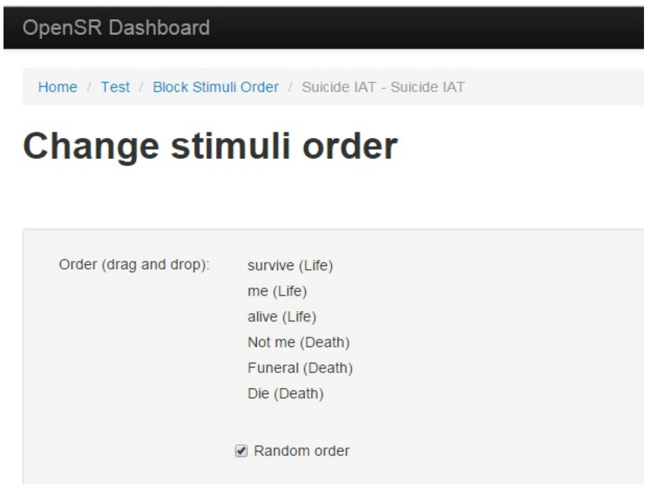
Color: *

Stimulus

Word	Image	Delete?
	Choose File No file chosen	
	Choose File No file chosen	
	Choose File No file chosen	

Add another Stimulus

Figure 5. The OpenSR add category form to customize the stimuli associated with designated categories.


OpenSR Dashboard

Home / Test / Block Stimuli Order / Suicide IAT - Suicide IAT

Change stimuli order

Order (drag and drop):
survive (Life)
me (Life)
alive (Life)
Not me (Death)
Funeral (Death)
Die (Death)

☑ Random order

Figure 6. The OpenSR change stimuli order form to customize stimuli order in test design.

Test Presentation Phase

A key aspect of any IAT is the actual test: the test participants and their responses. OpenSR allows researchers to design and manage every aspect of the test participants' experience in the online test environment. In this section, we describe the experience of OpenSR from the test participant's perspective.

An IAT within the OpenSR system comprises several Web pages. Upon first accessing the Web application through a URL specified by a researcher, the test participant is presented with the Test Selection page. Test participants are then prompted to select a test and enter a password to gain access. Test participants can forego test selection and supply only a password if provided with a direct link to the test by the researcher. After successfully entering the password and clicking Continue, the test participant is presented with an Informed Consent page, which is standard protocol for research studies involving human subjects. Test participants are then prompted to click Next, at which point they are redirected to the optional Introduction page if it had been configured by the researcher. Clicking Next, from either the Informed Consent page or the Introduction page, the participant may be presented with a Primer page. The information and/or instructions on the Primer page are specified for each experimental or control group by the researcher to meet his/her particular testing goals. Following the Primer page, participants proceed to the Test page, where the first set of test instructions is presented. For example, verbiage for IAT instructions involving a keyboard input device could be,

> Put your middle or index fingers on the F and J keys of your keyboard. Words representing the categories at the top of your screen will appear one by one in the middle of the screen. If the item belongs to a category on the left, press the F key. If the item belongs to a category on the right, press the J key. Items belong to only one category. If you make an error, an X will appear. Fix the error by pressing the other key. Try to go as fast as you can while making as few mistakes as possible.

Test participants are then prompted to press the space bar in order to proceed and, thereafter, are presented with the blocks of trials as set up by the researcher. The flow of the IAT is linear. Once a section of the testing is complete, the test participant is not given an option to return to previous sections. This also implies that there are no branch points within the test: All test participants must follow the same sequence of pages until completion.

Upon completion of the test, participants are prompted to click Next to finish the test, or they may be redirected to an external Web site (e.g., SurveyMonkey) if the researcher configured the test that way. After completion of the external survey, the test participant is automatically redirected back into the OpenSR test site. Whether the test participant has been sent to an external site and returned to the Open SR site or has just pressed the Next button following the completion of the test blocks, the participant is directed to a confirmation page which notifies him/her that the test has been completed. At this point, the participant exits the site. Figure 7 presents a model of the process that the test participants' experience.

Data Analysis Phase

The results module of the OpenSR framework contains the test results of participants. As test participants complete blocks of trials, the data (e.g., each trial response from the block) are uploaded and stored in a database on a server. Researchers access the test results through the administrative dashboard, which displays a complete list of participant tests and results. Researchers can sort results by group or by test (see Figure 8).

OpenSR currently records and exports the following information for each trial of each practice and test block: date, time, group assignment (i.e., experimental versus control), block title, designation of practice or test block, primary and secondary right and left categories (i.e., category labels), stimulus presented in each trial, and latency and correct versus incorrect status for each trial. Researchers can export data as comma-separated values directly into an Excel spreadsheet or another statistical program, such as SPSS for statistical analysis (see Figure 9).

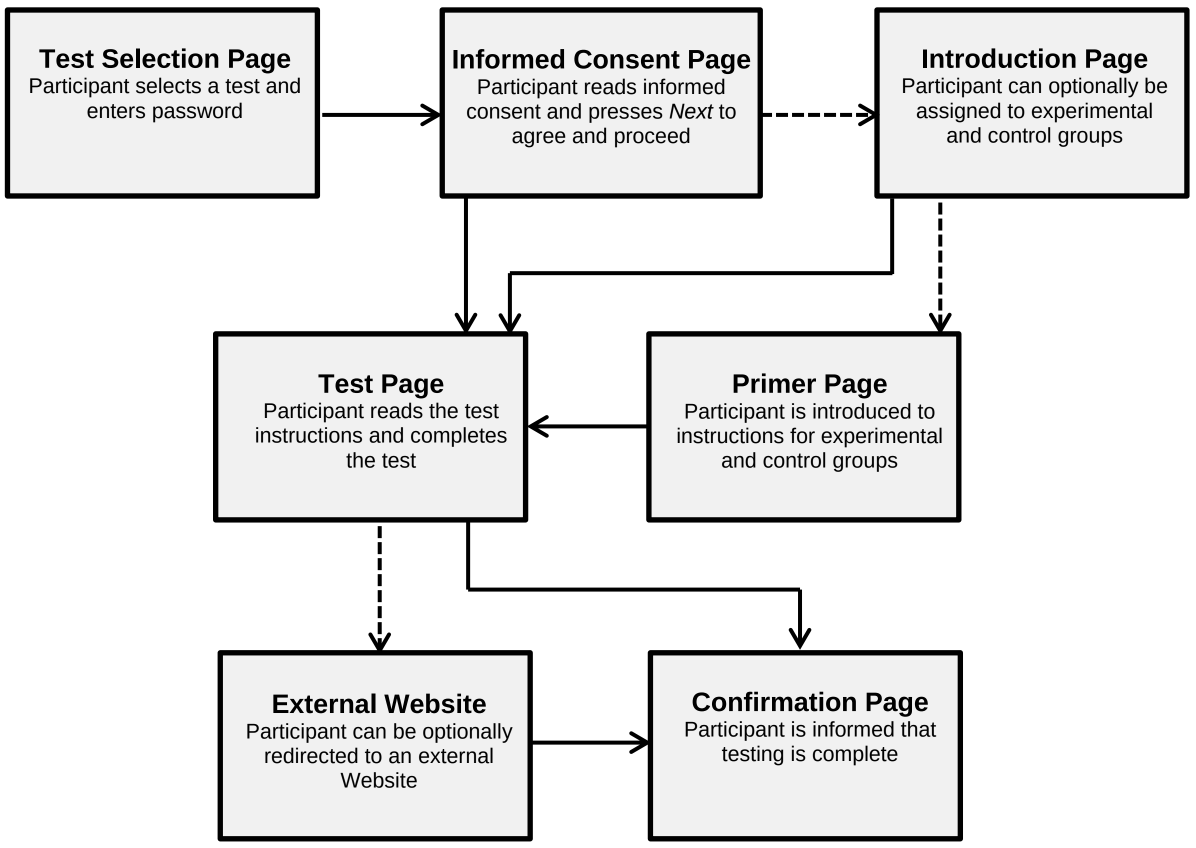

Test Selection Page
Participant selects a test and enters password
Informed Consent Page
Participant reads informed consent and presses *Next* to agree and proceed
Introduction Page
Participant can optionally be assigned to experimental and control groups
Test Page
Participant reads the test instructions and completes the test
Primer Page
Participant is introduced to instructions for experimental and control groups
External Website
Participant can be optionally redirected to an external Website
Confirmation Page
Participant is informed that testing is complete

Figure 7. Flow of the OpenSR process from participant's experience. Solid lines (⟶) represent mandatory testing path process. Dotted lines (- - ->) represent optional testing path process that can be specified by researchers during test configuration in the administrative dashboard.

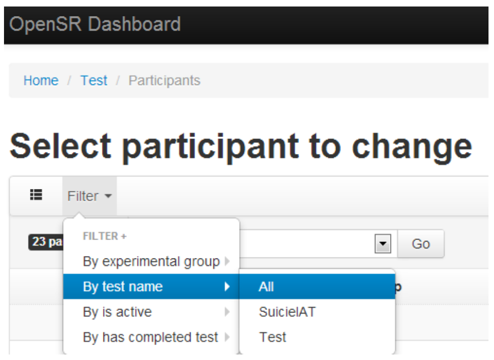


Figure 8. The OpenSR dashboard participant filter options.

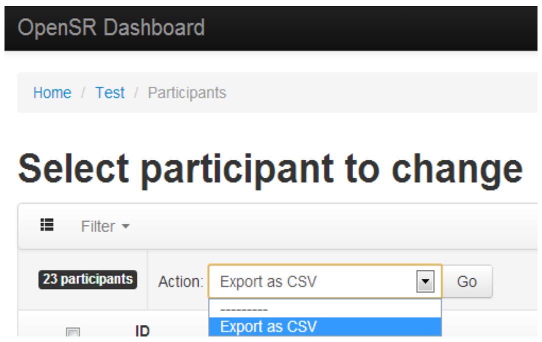


Figure 9. The OpenSR dashboard option for exporting data as CSV (comma-separated values) for analysis.

OpenSR FRAMEWORK DESIGN

OpenSR is a Web-based application that must be installed on a Web server. Depending on the needs of the researcher, the software can either be installed in a lightweight preview environment or a full production environment. The first approach leverages a cloud-based integrated development environment (IDE), which is an online source code editor that provides workspace to write, test, and run software on a dedicated Web server (examples of cloud-based IDEs include Cloud9, Codeanywhere, Kodingen, and SourceKit). These environments can simplify the installation process. However, because IDEs are used primarily for development and testing, they lack scalable databases and may be slow during periods of increased network traffic. Nevertheless, a lightweight cloud-based IDE provides an ideal environment for the development and testing phases. A more robust approach for installing and deploying OpenSR is a dedicated Web server that does not operate in the cloud. The downside to this approach is that more technical expertise is needed for setup. For example, the Web server must have the following software installed and configured: Django 1.5, Python 2.7.6, Apache, and a relational database system, such as PostgreSQL, MySQL, or SQLite. Additional Django and Python packages must be installed as well, including psycopg2 and django-sortedm2m. However, this approach provides several benefits above and beyond the lightweight IDE approach. Namely, the software can have a larger database, runs faster, and can handle increased network traffic depending on the underlying hardware.

Numerous technologies were used in the development the test configuration phase (i.e., the design of the IAT) to facilitate the presentation phase (i.e., the presentation of the IAT to the test participants) and the analysis phase (i.e., retrieving and formatting IAT data). OpenSR was developed using several computer languages, namely HTML5, CSS3, Python 2.7.2, and JavaScript. The core functionality includes an IAT generator, a configuration GUI, and data retrieval tools. Session handling, page serving, and test result output file generation are handled using Django, which is a Python Web framework. The interactive component of the IAT is built in JavaScript, a client-side scripting language, allowing participants to interact with the application without refreshing the Web browser.

Python

Python is an extensible open-source development language capable of interfacing with computer hardware. We chose Python for its excellent readability due to enforced formatting and concise yet intuitive syntax, making code easy to read and debug. Additional benefits are that it fosters rapid application development and features extensive object-oriented capabilities and many high-level data types, such as dictionaries, lists, and sets (Hughes, 2008; Raymond, 2000). The standard libraries integrated into the language are robust and several stimuli libraries—such as Pyglet and NumPy—are already built in, thus reducing the need for platform-specific code. In short, Python is an agile language that is intuitive and easy to use and provides advanced functionality leveraged by large-scale projects (Peirce, 2007, 2009). For OpenSR, Python was used in conjunction with Django to create the Web server. Specifically, when a test participant accesses OpenSR in his/her Web browser, a message is sent to the Python Web server. The Web server then generates the test and sends it back to the test participant's browser to be displayed.

Django

Django is a high-level Python Web framework that supports dynamic Web pages for the entire system. For example, as a researcher or test participant interacts with the program, Django generates and delivers the appropriate Web page to the client's browser window. Instead of using the scripting language PHP to generate dynamic Web pages, we selected Django for multiple reasons. First, the Django framework fully supports object-relational mapping. Data models implementing dynamic database access application programming interfaces can be defined rather than relying on advanced queries to interact with the database, thus decreasing complexity while also reducing development time. Second, an administrative interface is automatically generated upon data model creation and registration, allowing development to focus on the extensibility and the flow and logic of the application. Third, Django features a flexible template system consisting of an extensible and intuitive template language that decouples the application logic from the visual design of the Web site. This separates the design, content, and implementation of the application into distinct modules, making OpenSR extensible for future adaptations (e.g., researchers have the ability to add additional features). Finally, the general emphasis on the Don't Repeat Yourself principle reduces both the complexity and the length of application logic and aims to reduce the redundancy of information and application logic throughout a system. Django aids in facilitating test creation and presentation for OpenSR.

JavaScript and jQuery

JavaScript is a standard Web programming language; jQuery, a popular library written in JavaScript, gives the ability to dynamically change the content on the screen without refreshing the browser. This comprehensive framework, featuring HTML (Hypertext Markup Language) manipulation, asynchronous server interaction, event handling, and numerous animations and effects enhance the functionality of JavaScript. We used JavaScript because the code is executed on the client side, allowing dynamic interaction with the IAT without the need to refresh the test participant’s Web browser. Most importantly, the application logic for OpenSR uses JavaScript and jQuery, chosen for their accurate time measurement. Response times could not have been recorded with millisecond accuracy without using JavaScript and jQuery.

HTML5 and CSS3

There is an increasing need to design visually appealing Web pages that are easy to customize and interact with (Wu, Hu, & Shi, 2013). HTML5 and Cascading Style Sheets level 3 (CSS3) are Web technologies that allow users to control visual presentation and user interaction. In general, HTML is a language for describing elements and attributes of Web pages. CSS3 is a graphical guideline that incorporates style and formatting rules for specifying visual presentation properties (Bolin, Webber, Rha, Wilson, & Miller, 2005; Comai & Mazza, 2012; Raman, 2009). Both HTML5 and CSS3 are standard Web technologies to the extent that Web browsers (e.g., Opera, Google Chrome, Mozilla Firefox, Safari, Internet Explorer) can appropriately interpret and render HTML5/CSS3 pages. HTML5 is used to organize the content in OpenSR, and CSS3 is used to style the content, such as the background color, font family, and the visual rendering and formatting of various elements like input fields, buttons, and text areas. In essence, HTML5 organizes the content of the page, whereas CSS3 styles and lays out the content. Both HTML5 and CSS3 were selected for the development of OpenSR to handle the design of the entire Web application—including the administration control panel and the IAT—thus allowing researchers the ability to easily create, customize, and present an IAT for their own purposes.

Bootstrap

Bootstrap was used in the design of OpenSR’s IAT specifically to support the styling and design of pages and the dynamic features of the application. Bootstrap is a relatively easy framework for creating attractive Web sites. We utilized the Bootstrap framework because it eases the aesthetic design of the application by automatically styling all elements (Lerner, 2012) and ensures compliance with the standard Web development conventions and protocols for Web design and architecture. Most importantly, the framework supports a fluid layout. Pages built using Bootstrap work on any device regardless of the screen resolution, allowing an application to be used on mobile devices and traditional computers.

In summary, the technologies used in the development of OpenSR provide a user-friendly experience for anyone interested in creating, customizing, and administering an S–R test such as the IAT. Python, an extensible open-source development language, was used to develop the OpenSR framework. Django, a Python Web framework, was used to generate and deliver Web page requests to any user’s browser window. HTML5 and CSS3 were used to handle the design of OpenSR—including the administration control panel and the IAT—thus allowing research designers the ability to easily create, customize, and present an IAT for their own purposes. The use of JavaScript and jQuery allows test participants to interact with the IAT without refreshing their Web browser because the scripts are executed on test participants’ computers. Most importantly, JavaScript provides the capability for response times to be recorded with millisecond accuracy. Bootstrap was used for the styling and design of pages as well as the dynamic features of the application (e.g., the ability for OpenSR to work on any device regardless of the screen resolution).

HARDWARE AND SOFTWARE CONSIDERATIONS

The use of S–R tests necessitates examining hardware and software issues pertaining to accurately capturing and recording response times. Advances in most contemporary hardware and software have decreased many of the challenges that might have impacted the use of a program such as OpenSR. We would be remiss, however, if we did not consider and note the potential impact of hardware and software that do not meet certain requirements. Therefore, we provide an overview of three hardware and/or software considerations for researchers conducting S–R tests: variability in operating systems, variability in monitors, and variability in input devices. We also address the important consideration of Internet connectivity during the test.

Preemptive multitasking enables the operating system to interrupt any program that may be running in the background or foreground, which can skew how a program performs operations and when instructions are executed. Interruptions by operating systems are less of an issue nowadays because multicore processors, which are increasingly a standard in computers, enable parallel execution of instruction. In other words, software (e.g., an IAT) can execute more quickly and consistently now on multicore processor systems than when computers had only one processor to handle requests from both the operating systems and the user programs. Variability in testing performance (e.g., recording of response times) due to different operating systems is thus negated in newer computers.

Different types of monitors (e.g., LCD, CRT) have specific mechanisms for “painting the screen” and related delay characteristics for displaying images on the screen. Older CRT monitors, although having a faster refresh rate, paint the screen only one pixel at a time starting from the upper left corner and moving systematically to the lower right. This results in about a 10 ms lag between when the test participant sees the first and last pixels rendered on the screen (Straw, 2008). An LCD monitor paints all the pixels simultaneously but has approximately a 10 ms delay in doing so and a refresh rate of approximately 60 Hz (Straw, 2008). If a program was instructed to leave an image on the screen for a given number of seconds, the image would either be present slightly longer or shorter, depending on when it is initially painted with respect to the 60 Hz clock cycle for refresh rate. Despite differences across types of monitors, human reaction time to visual stimuli is approximately 200 ms (Kosinski, 2013; Ohyanagi & Sengoku, 2010). Therefore, differences in types of monitors should produce negligible variance in response times. Even if a difference in recorded response time based on a factor such as monitor type was present, the results for an individual would be consistent and mathematically averaged out across all trials. For example, a 60 Hz monitor repaints the screen 60 times per second and displays a frame in roughly 17 ms. As a result, a constant variable (e.g., 17 ms) is added to the reaction time of every trial.

Another hardware consideration is the input device and its delay characteristics. Examples of common input devices are the keyboard and the mouse. Keyboard strokes and mouse clicks send a priority interrupt to the operating system, resulting in immediate execution of the command. However, USB-, Bluetooth wireless-, and serial-connected keyboards all have different signaling delay factors from the device to the operating system.

USB has a 20–30-ms delay overall (Peirce, 2009), but this is considered a constant because all responses are being measured on an individual computer during an individual testing session and should be a mathematically constant factor for each participant.

OpenSR is a browser-based tool that requires an Internet connection while a test participant sorts the stimuli (e.g., when the participant presses the left category key, the F, or the right category key, the J). When a test participant starts a test, all necessary data (e.g., the trial and block data) are retrieved from the server and loaded into the test participant's Web browser. Thereafter, the client-side code running in the participant's browser displays the retrieved test data. Responses (e.g., pressing F or J) and associated reaction times for each block of trials are recorded locally (i.e., in the test participant's browser) and the data are sent asynchronously to the server, without interrupting the experience of the test participant. The overall speed of the Internet connection has no impact on recording participant reaction time because response time is measured on the client side, not the server side. If the participant is not connected to the Internet, the server cannot receive the results and the results cannot be saved. It is possible that a participant could start taking a test and subsequently lose Internet connectivity during the test. In this case, when the participant reconnects to the Internet and refreshes the test page, the test will start at the beginning of the block that was active when connectivity was lost. For example, if a participant successfully completed the first and second block but then lost connectivity during the third block, he/she would start at the beginning of the third block when the test page is refreshed after regaining Internet connectivity.

Advances in technology allow personal computers to execute multiple programs without altering the performance of any given task. In addition, regardless of the performance of a test participant's computer, response time on a single trial is less important than overall difference scores between blocks of trials. In addition, increasing the number of trials in a response-time test such as the IAT can improve the reliability of the test because extraneous factors should be mathematically negated over many trials (Lane et al., 2007). Likewise, any differences based on hardware should be statistically averaged out across all trials on a participant's test.

CONCLUSION

This paper presents OpenSR, a program that facilitates a specific type of S–R test in the field of psychology: the IAT. OpenSR could be adapted to other S–R testing environments to meet researchers' needs. For example, the OpenSR framework could be used for priming tasks, in which exposure to a stimulus influences subsequent responses to another stimulus. OpenSR currently provides all the functionality for creating a priming experiment using text or images in a designated order specified by a researcher. That is, no additional extensions of the program are required for setting up a priming task. Extended applications of the OpenSR framework could include a wide range of research focuses, such as political science (e.g., for assessing preferences for different political candidates) or Web design (e.g., for assessing ideal placement of text and images on Web sites). In addition, OpenSR could be adapted for use in tasks in any discipline requiring the sorting of text or images into designated categories. Currently, OpenSR is set up to create and administer an IAT. However, the code base was developed with the idea that it could be extended for other uses. Extensions of the code base for other applications, however, would require programming knowledge, as would the addition of features to the current framework.

OpenSR addresses many of the challenges faced by researchers interested in conducting S–R research via an IAT. Specifically, it provides an open-source Web-based framework with a GUI that can be used for the creation and administration of an IAT, including the ability to easily customize the testing parameters. In addition, it includes capabilities for collecting and exporting data online (as opposed to the need for lab-only settings and software). OpenSR's user-centered approach thus makes it possible for researchers across disciplines to easily create and administer an IAT without paying expensive licensing fees. OpenSR is available for free download.

ENDNOTES

1. OpenSR can be downloaded for free from https://github.com/justinsvegliato/opensr
2. In an IAT, an error occurs when a stimulus (e.g., a word) is not sorted into the appropriate category. Each stimulus has one correct classification, which is designated in the test configuration phase.

REFERENCES

Barkley, R. A. (1997). Behavioral inhibition, sustained attention, and executive functions: Constructing a unifying theory of ADHD. *Psychological Bulletin, 121*(1), 65–94.

Bolin, M., Webber, M., Rha, P., Wilson, T., & Miller, R. (2005, October). *Automation and customization of rendered web pages.* Paper presented at the 18th Annual Association for Computing Machinery Symposium on User Interface Software and Technology, Seattle, WA, USA. Retrieved July 5, 2015, from http://groups.csail.mit.edu/uid/projects/chickenfoot/uist05.pdf

Bompas, A., & Sumner, P. (2009). Temporal dynamics of saccadic distraction. *Journal of Vision, 9*(9), 11–14.

Brainard, D. H. (1997). The psychophysics toolbox. *Spatial Vision, 10*(4), 433–436.

Buonocore, A., & McIntosh, R. D. (2008). Saccadic inhibition underlies the remote distractor eVect. *Experimental Brain Research, 191*(1), 117–122.

Comai, S., & Mazza, D. (2012). A model-driven methodology to the content layout problem in web applications. *ACM Transactions on the Web, 6*(3), 1–38.

Davis, F. D. (1989). Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS Quarterly, 13*(3), 319–340.

DeLone, W. H., & McLean, E. R. (1992). Information systems success: The quest for the dependent variable. *Information Systems Research, 3*(1), 60–95.

DeLone, W. H., & McLean, E. R. (2003). The DeLone and McLean model of information systems success: A ten-year update. *Journal of Management Information Systems, 19*(4), 9–30.

Dennis, A., Wixon, B. H., & Tegarden, D. (2012). *Systems analysis & design: An object-oriented approach* (4th ed.). Hoboken, NJ, USA: Wiley.

Deubel, H. (2008). The time course of presaccadic attention shifts. *Psychological Research, 72,* 630–640.

Edelman, J. A., & Xu, K. Z. (2009). Inhibition of voluntary saccadic eye movement commands by abrupt visual onsets. *Journal of Neurophysiology, 101*(3), 1222–1234.

Fielding, R. T., & Taylor, R. N. (2002). Principled design of the modern web architecture. *ACM Transactions on Internet Technology, 2*(2), 115–150.

Goto, Y., Hatakeyama, K., Kitama, T., Sato, Y., Kanemura, H., Aoyagi, K., Sugita, K., & Aihara, M. (2010). Saccade eye movements as a quantitative measure of frontostriatal network in children with ADHD. *Brain and Development, 32,* 347–355.

Greenwald, A. G., McGhee, D. E., & Schwartz, J. L. K. (1998). Measuring individual differences in implicit cognition: The implicit associations test. *Journal of Personality and Social Psychology, 74*(6), 1464–1480.

Greenwald, A. G., Nosek, B. A., & Banaji, M. R. (2003). Understanding and using the implicit associations test: I. An improved scoring algorithm. *Journal of Personality and Social Psychology, 85*(3), 197–216.

Greenwald, A. G., Poehlman, T. A., Uhlmann, E. L., & Banaji, M. R. (2009). Understanding and using the implicit association test: III. Meta-analysis of predictive validity. *Journal of Personality and Social Psychology, 97*(1), 17–41.

Hughes, P. (2008, June 10). Why python is the best. *Linux Journal* [online]. Retrieved on May 15, 2014, from http://www.linuxjournal.com/content/why-python-best

Kleiner, M., Brainard, D., & Pelli, D. (2007, August). What's new in psychtoolbox-3? *Perception, 36* (Supplement; abstracts). Retrieved June 2, 2015, from http://www.perceptionWeb.com/ecvp/ecvp07.pdf

Koelewijn, T., Bronkhorst, A., & Theeuwes, J. (2009). Auditory and visual capture during focused visual attention. *Journal of Experimental Psychology*: *Human Perception and Performance, 35*(5), 1303–1315.

Kosinski, R. J. (2013). *A literature review on reaction time.* Retrieved June 25, 2015, from http://biae.clemson.edu/bpc/bp/lab/110/reaction.htm

Lane, K. A., Banaji, M. R., Nosek, B. A., & Greenwald, A. G. (2007). Understanding and using the implicit association test: IV. What we know (so far). In B. Wittenbrink & N. S. Schwarz (Eds.), *Implicit measures of attitudes: Procedures and controversies* (pp. 59–102). New York, NY, USA: Guilford Press.

Lerner, R. M. (2012, June). At the forge: Twitter bootstrap. *Linux Journal, 218,* Article 6.

Ludwig, C. J., Gilchrist, I. D., & McSorley, E. (2005). The remote distractor eVect in saccade programming: Channel interactions and lateral inhibition. *Vision Research, 45*(9), 1177–1190.

Matheus, A. B., Matheus, C. C., & Neely, M. P. (2014). Examining the relationship between IQ, DQ, usefulness, EoU, and task performance. *Communications of the Association for Information Systems, 36*(15), 285–304.

Matheus, C. C., & Svegliato, J. (2013, May). *Stimulus-response tests: An applied demonstration.* Paper presented at IEEE International Conference on Research Challenges in Information Science, Paris, France.

Nielsen, J. (2005). Usability for the masses. *Journal of Usability Studies, 1*(1), 2–3.

Ohyanagi, T., & Sengoku, Y. (2010). A solution for measuring accurate reaction time to visual stimuli realized with a programmable microcontroller. *Behavior Research Methods, 42*(1), 242–253.

Pearlson, K., & Saunders, C. (2013). *Managing and using information systems: A strategic approach.* Hoboken, NJ, USA: John Wiley.

Peirce, J. W. (2007). PsychoPy: Psychophysics software in python. *Journal of Neuroscience Methods, 162*(1–2), 8–13.

Peirce, J. W. (2009). Generating stimuli for neuroscience using PsychoPy. *Frontiers in Neuroinformatics, 2*(10), 1–8.

Pelli, D. G. (1997). The video toolbox software for visual psychophysics: Transforming numbers into movies. *Spatial Vision, 10*(4), 437–442.

Quay, H. C. (1997). Inhibition and attention deficit hyperactivity disorder. *Journal of Abnormal Child Psychology, 25*(1), 7–13.

Raman, T. V. (2009). Toward 2w: Beyond Web 2.0. *Communications of the ACM, 52*(2), 52–59.

Raymond, E. (2000, April 30). Why python? *Linux Journal* [online]. Retrieved on July 5, 2015, from http://www.linuxjournal.com/article/3882

Rudman, L. A., Glick, P., & Phelan, J. E. (2008). From the laboratory to the bench: Gender stereotyping research in the courtroom. In E. Borgida & S. T. Fiske (Eds.), *Beyond common sense: Psychological science in the courtroom* (pp. 83–102). Malden, MA, USA: Blackwell.

Rueda, M. R., Fan, J., McCandliss, B. D., Halparin, J. D., Gruber, D. B., Lercari, L. P., & Posner, M. I. (2004). Development of attentional networks in childhood. *Neuropsychologia, 42,* 1029–1040.

Straw, A. D. (2008). Vision Egg: An open-source library for realtime visual stimulus generation. *Frontier in Neuroinformatics, 2*(4), 1–10.

Tingley, K. (2013, June 30). The suicide detective. *The New York Times* [online]. Retrieved May 24, 2015, from http://www.nytimes.com/2013/06/30/magazine/the-suicide-detective.html?_r=0

van Stockum, S., MacAskill, M. R., & Anderson, T. J. (2011). Bottom-up effects modulate saccadic latencies in well-known eye movement paradigm. *Psychological Research, 75,* 272–278.

Venkatesh, V., Morris, M. G., Davis, G. B., & Davis, F. D. (2003). User acceptance of information technology: Toward a unified view. *MIS Quarterly, 27*(3), 425–472.

White, M. J., & White, G. B. (2006). Implicit and explicit occupational gender stereotypes. *Sex Roles, 55*, 259–256.

Wu, O., Hu, W., & Shi, L. (2013). Measuring the visual complexities of web pages. *ACM Transactions on the Web, 7*(1), 1–34.

Authors' Note

All correspondence should be addressed to
Carolyn C. Matheus
Marist College
3399 North Road
Poughkeepsie, NY 12601
Carolyn.Matheus@Marist.edu

Human Technology: An Interdisciplinary Journal on Humans in ICT Environments
ISSN 1795-6889
www.humantechnology.jyu.fi