Behav Res
DOI 10.3758/s13428-015-0567-2

Psychophysics in a Web browser? Comparing response times collected with JavaScript and Psychophysics Toolbox in a visual search task

Joshua R. de Leeuw · Benjamin A. Motz

Abstract Behavioral researchers are increasingly using Web-based software such as JavaScript to conduct response time experiments. Although there has been some research on the accuracy and reliability of response time measurements collected using JavaScript, it remains unclear how well this method performs relative to standard laboratory software in psychologically relevant experimental manipulations. Here we present results from a visual search experiment in which we measured response time distributions with both Psychophysics Toolbox (PTB) and JavaScript. We developed a methodology that allowed us to simultaneously run the visual search experiment with both systems, interleaving trials between two independent computers, thus minimizing the effects of factors other than the experimental software. The response times measured by JavaScript were approximately 25 ms longer than those measured by PTB. However, we found no reliable difference in the variability of the distributions related to the software, and both software packages were equally sensitive to changes in the response times as a result of the experimental manipulations. We concluded that JavaScript is a suitable tool for measuring response times in behavioral research.

Keywords JavaScript · Psychophysics Toolbox · Response times · Visual search · Online experiments

Psychologists have been measuring response times in online experiments for nearly two decades (Musch & Reips, 2000; see, e.g., Nosek, Banaji, & Greenwald, 2002). Recently, the growing popularity of online behavioral experiments, drawing from a diverse population of Internet users, has prompted increased interest in empirical validation of response time measures collected on the Web, as well as in online behavioral experiment methodology more broadly. This is an important area of research, because online data collection may present confounds to some methodologies, particularly those that depend on tight control of visual stimulus presentation and response recording (Crump, McDonnell, & Gureckis, 2013; Reimers & Stewart, 2014). Considering the importance of precise measurements in psychophysical research, psychologists who hope to measure response times in an online experiment would benefit from a strong body of scholarly work demonstrating the validity of these methods. Recent studies have begun to establish this literature, by either replicating previous laboratory response time research using exclusively online methods or directly measuring display lags using online or laboratory systems. In this article, we offer a novel contribution to this growing literature, directly comparing human participants' response times between browser-based and laboratory-based experimental platforms, in a within-subjects design.

Several experiments have replicated laboratory-based response time effects using online methods. Crump, McDonnell, and Gureckis (2013) replicated several classic response time tasks and demonstrated that both the direction and magnitude of the effects were consistent with values reported in the literature. In their sample of seven different experiments, the only response time study that failed to replicate was a masked-priming design that required very precise control over the stimulus presentation duration at very short (16-ms) intervals, a feature that might not be possible on all Internet users' displays. Barnhoorn, Haasnoot, Bocanegra, and van Steenbergen (2014) were subsequently able to replicate a masked-priming effect (as well as the Stroop effect and attentional blink effect) using more recently available JavaScript features for controlling stimulus presentation at very short intervals. Simcox and Fiex (2014) replicated both a flanker

J. R. de Leeuw (✉) · B. A. Motz
Department of Psychological and Brain Sciences, Indiana University, 1101 E. 10th Street, Bloomington, IN 47405, USA
e-mail: jodeleeu@indiana.edu

Published online: 12 March 2015

task and a lexical decision task, finding magnitudes of response times similar to those in the original laboratory studies. Zwaan and Pecher (2012) replicated several response-time-based findings related to mental simulation in language comprehension. They were able to replicate the directions and approximate magnitudes found in four previously published experiments, but failed to replicate two experiments (finding effects opposite from the ones previously reported).

It is encouraging that many of these laboratory experiments have been replicated in an online environment. Positive replications provide evidence that response times measured online are comparable to those measured in the laboratory, and being able to conduct such experiments online could improve the generalizability of findings, enable faster data collection from many more participants, and expand the range of possible methodologies. But when a replication attempt fails, there are many potential explanations: The hardware and software used for running the experiment and measuring response times are different between participants, these consumer systems may be of lower quality than laboratory equipment, the subject population will be different, the effect may not have been generalizable to a diverse population of Internet users, some replication attempts should fail just by chance (Francis, 2013), and so on.

To discover what, if anything, is different between response times measured online and response times measured in the laboratory requires isolating particular variables that differ between online and laboratory experiments. One crucial variable, and the focus of the experiment presented here, is software. Online experiments cannot use software that is widely accepted as providing quality response time data, such as MATLAB's Psychophysics Toolbox (Brainard, 1997). Instead, they rely on software developed in JavaScript (the only programming language supported natively by all major Web browsers) or software that requires plugins to extend Web browsers (like Java applets or Adobe Flash).

A widely used strategy for validating the response time properties of software is to compare the timings of visual stimulus displays when presented using Web browsers or laboratory software. One way to accomplish this is to use an external device (another computer or a special-purpose microcontroller) to monitor changes in the luminance of a computer monitor and to simulate responses (De Clercq, Crombez, Buysse, & Roeyers, 2003). The external device detects when the display changes, sends a signal to the computer (e.g., emulating a subject pressing a key), and the computer measures the amount of time from when the command to change the monitor display was issued to the time at which the response from the external device was received. Ideally, this time lag should be close to 0, but practical hardware and software limitations cause delays in the tens of milliseconds. This approach has been used to measure error in response times in a variety of Web-based programming environments, including Flash, JavaScript, and Java applets (Barnhoorn et al., 2014; Neath, Earle, Hallett, & Surprenant, 2011; Reimers & Stewart, 2014; Schubert, Murteira, Collins, & Lopes, 2013). A related methodological strategy is to have human subjects complete an experiment using a system with very low lag and with the system of interest. Using this method, Reimers and Stewart (2007, 2008) have investigated the response time properties of Adobe Flash and Adobe Flash Lite running on mobile phones.

All systems used for measuring response times, whether online or in the laboratory, will introduce some timing error, and for most research questions the amount of error (the time lag and variability introduced by software and hardware) generated by standard laboratory hardware and software seems to be acceptable to the research community. The question that is most relevant to researchers interested in online response time measurement is not how much error is generated by a particular online-ready software package, but rather, how does the error generated by software packages used for online data collection compare to that from software used in the lab?

Recent studies have suggested that JavaScript may introduce incrementally more measurement error than standard laboratory software. Neath et al. (2011) tested the errors of both JavaScript and MATLAB's Psychophysics Toolbox (henceforth PTB; Neath et al. measured other platforms, as well) using a device that directly recorded when a stimulus appeared on the screen and then immediately triggered a keyboard press, and found that the mean response time of JavaScript was approximately 20 ms greater than that of PTB, with significantly more variance in the distribution for JavaScript-based measurements (although the increased variance was isolated to the slower of two systems under investigation). However, their method—an external mechanical device triggering a keypress as soon as a change is detected in the monitor—produced very short detection times (<70 ms), which are far faster than a human subject would respond. It may be that human response times (on the order of hundreds of milliseconds) would have a multiplicative effect on the error of response recording, or that these longer response times would reduce error, allowing JavaScript to recover from display processing to record a more temporally precise response.

Reimers and Stewart (2014) investigated further, also using a mechanical device to simulate keypresses to visual stimulus presentation, with automated delays of 150, 300, and 600 ms (to simulate a range of human response times), and similarly observed a clear additive lag for the responses measured in JavaScript. However, this lag appeared to be roughly stable across the three different response time conditions, and the standard deviations of the measured keypresses were small (mostly <10 ms). Thus, this more recent article should provide some comfort that the error introduced by JavaScript may be constant across experimental manipulations, or at least that the error would be sufficiently small that it would not affect the

results of a real experiment. Ultimately, the authors of both studies (Neath et al., 2011; Reimers & Stewart, 2014) concluded that, for common research applications, as long as a researcher is only interested to make within-subjects comparisons across conditions in an online experiment, the lag imposed by JavaScript should not substantively affect these comparisons. Toward the goal of advancing the literature on online behavioral research methods, the present study was performed to empirically evaluate this key prediction.

In this experiment, we compared human participants' response times measured by different software packages for the same experiment, making within-subjects comparisons between software packages, keeping all other experimental variables equal. The advantage of this method is that it allows for direct comparisons of response time distributions at the subject level across variations of meaningful psychological parameters. Thus, it is possible not only to check whether an effect replicates in a statistical sense, but also how the difference in the response time distributions measured by each software package changes over a range of possible human response times. The major disadvantage of this approach, as compared to approaches that use an external device to simulate responses, is that substantially more variation will be introduced by human responses, which will diminish the ability to detect statistical differences between the software packages. However, for behavioral researchers interested in response time measurements, the relevant question is probably not whether there are *any* differences, but whether the differences would systematically affect a distribution of the size typically collected for a behavioral experiment.

In this experiment, our target comparison is between JavaScript and MATLAB's Psychophysics Toolbox. We chose JavaScript because it is currently the best available option for developing online experiments. In combination with HTML and CSS, JavaScript can be used to develop a wide array of behavioral tasks that look and behave like laboratory tasks. It is natively supported by all major browsers, so subjects do not need to download additional software in order to run an experiment (this is a particularly important consideration when using Amazon's Mechanical Turk, because its terms of service prohibit requiring workers to download software to complete a human intelligence task). Other options, such as Flash or Java applets, are not natively supported (although adoption rates are quite high, so most users will be able to run experiments developed in this way). Given the flexibility of modern JavaScript Web development techniques, few, if any, tasks can be developed in Flash or Java that cannot be developed in JavaScript.

Although JavaScript is the best available solution for creating online experiments, some aspects of the language are not desirable for behavioral experiments. The most important concern is that JavaScript is primarily an event-driven language. To illustrate why this is important: Imagine you are trying to record the exact time that a key is pressed on a keyboard. A standard way to do this (e.g., in PTB) would be to create a very tight loop that continually checks whether a key has been pressed. In JavaScript, instead of checking to see whether a key has been pressed, we instead must specify an *event handler* that performs some action when the key is pressed (there is no other way to check whether a key has been pressed, so the strategy of a very tight loop does not work). When the key is pressed, the event is added to the event queue. When the browser processes the event in the event queue, the appropriate action is executed. This can introduce delays based on how many other events are in the queue and how fast the browser is at processing events. Although it is certainly possible to imagine situations in which this model of processing a keyboard press could cause significant problems for measuring response times, it is not obvious whether this would be a problem for a typical behavioral experiment, and thus it is best treated as an empirical question.

We chose PTB as a representative lab-oriented software package. PTB is a set of functions for MATLAB and GNU/Octave, oriented toward conducting vision experiments. It is widely used, and has been cited over 4,000 times, with over 1,000 citations in 2013 alone.[1] It has relatively low measurement error for response times, due in part to the ability to synchronize response time recording with the refresh of the visual display (Neath et al., 2011). Given that PTB is accepted as a tool for measuring response time data, our aim was to establish the relative performance of JavaScript as compared to PTB. If both software packages are equally sensitive to changes in response time distributions, then JavaScript measurements should be considered as valid as those collected by PTB.

We tested both platforms in a simple visual search task. Visual search was selected as a representative psychophysics paradigm because (1)it is a highly investigated visual task using largely standardized methods (Wolfe & Horowitz, 2004); (2)it is possible to test subjects on a large number of trials, and gain a large amount of response time data, in a relatively short amount of time (4 s per trial, in the present study); and most importantly, (3)experimental manipulations to the number of items in the search array yield robust, large, well-characterized changes in response times (that can easily be modeled as a simple linear function; Wolfe, 1998). On the basis of previous research (Shen & Reingold, 2001), we expected mean response times ranging from roughly 700 to 1,200 ms that were directly proportionate to the number of stimulus items in the search set.

[1] These data are based on citations to the three articles that the authors of PTB have indicated are the desired citations (Brainard, 1997; Kleiner, Brainard, & Pelli, 2007; Pelli, 1997). The data were collected from the Web of Science on September 23, 2014. Duplicate citations (i.e., an article citing more than one of the articles) were not counted toward the total.

To minimize the amount of noise in the measurements related to nonsoftware factors—such as subject fatigue, order effects, practice effects, and so on—we developed a methodology in which the subject was able to complete the experiment using both systems simultaneously (see Fig. 1). We interleaved the presentations of trials from each system by using two projectors that were aligned to project on the same display surface. From the subjects' perspective, they were completing one single experiment. In reality, the experiment was running on two independent computers, one running the JavaScript version of the experiment, and the other running the PTB version, with overlapping displays.

Method

Subjects

A total of 30 subjects (19 females, 11 males) participated in the experiment in exchange for $10. One subject was excluded from the analysis due to partial data loss. The subjects were 18–34 years old (mean 21.7), and most were students at Indiana University.

Task

Subjects completed a visual search task in which they identified the presence or absence of a target (N) in an array of distractors (И). The task closely matched a previously reported experiment by Wang, Cavanagh, and Green (1994).

In each trial, a fixation dot appeared in the center of the screen. After 1,000 ms, an array of two, three, four, or six items appeared. The number of items was randomly chosen on each trial, with equal probabilities for the different set sizes, and both systems (JavaScript and PTB) generated the array by loading individual stimulus images into the display. The items were equidistant from the fixation dot, spaced evenly along a circular path (Fig. 2). The items remained on the screen until the subject responded, but if no response was given after 2,000 ms, the trial ended. There was a 50% chance that any particular trial would contain a target. Subjects identified the presence or absence of a target by pressing a button on a custom hand-held response device. The device had two buttons, one held in each hand; the subject's thumbs operated the buttons. A subject responded that the target was present by pushing the button in his or her dominant hand (as determined by self-report), and when the target was absent, the subject pushed the button in his or her nondominant hand.

At the start of the experiment, subjects completed 30 practice trials, which were identical to the experimental trials except that corrective feedback was given after each response. The feedback remained on the screen for 2,000 ms. During the practice phase, a new trial began every 6,000 ms. There was a 20-s break after the practice trials. Subjects then performed 400 experimental trials, with a 45-s break after the first 200 trials. During the experimental trials, a new trial began every 4,000 ms.

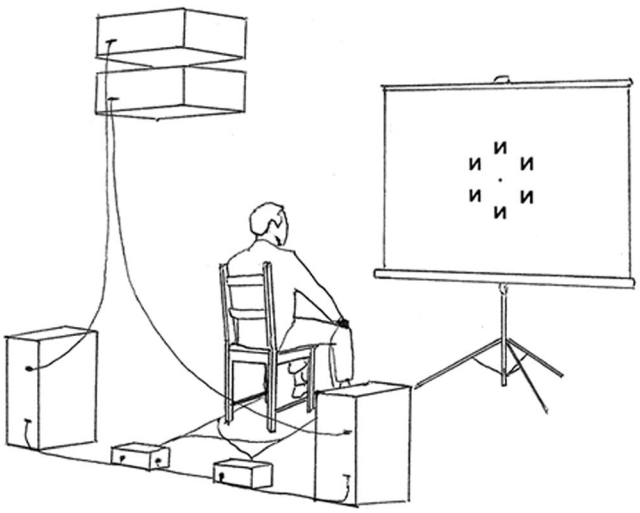

Fig. 1 Experimental setup. Subjects were seated approximately 2.1 m away from a projector screen and given a response device to hold in each hand. Behind the subject, two separate computers were running the experiment, one running the jsPsych JavaScript library and the other running Psychophysics Toolbox. Each computer was connected to a projector, and the projectors were vertically stacked and aligned to project onto the same area of the screen. Each computer was connected to an Arduino device, which handled response processing, as well as controlling the order of trials, ensuring that only one computer presented a trial at a time

Interleaved experiment design

We implemented the visual search task with two different software packages: PTB for MATLAB (Brainard, 1997) and jsPsych, a library for creating behavioral experiments with JavaScript (de Leeuw, 2014). We carefully matched the two implementations so that the displays were identical, regardless of which system was displaying the trial. We also matched, as best we could, the code for recording the response time data from both systems; both implementations started recording response times immediately after the stimulus was displayed and stopped recording immediately after a buttonpress was registered. The JavaScript trials were displayed in a full-screen Web browser (Google Chrome; accessing the experiment script from a remote Web server over the Internet), whereas the PTB trials were displayed natively in a local instance of MATLAB. The code for both experiments is available online at http://hdl.handle.net/2022/19253.

We ran the experiment simultaneously on two different computers, one running the PTB implementation and the other running the jsPsych implementation. The computers were identical-model iMacs (3.06-GHz Intel Core 2 Duo, 4 GB RAM) running OSX 10.6.8. Each computer was connected

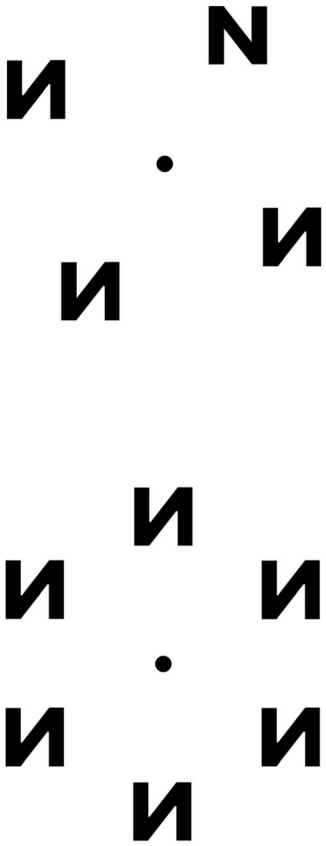

Fig. 2 The visual search task. Subjects were to report the presence or absence of a target (N) among distractors (И). The top image shows a target-present trial, and the bottom image shows a target-absent trial

to a Dell 5100MP projector. We alternated which computer ran the PTB and JavaScript trials, and we also alternated, every two sessions, which computer was connected to which projector, to control for the unlikely possibility of effects related to the individual computers or projectors. The projectors were stacked vertically, and the projected images were aligned by manually adjusting the location and angle of each projector. At the start of each experimental session, a calibration image was displayed by each system to check for alignment and equivalent brightness.

The subjects were seated 2.13 m away from the screen. The projected search array was 0.25 m in diameter, occupying approximately 6.5 deg of visual angle.

During the experiment, only one system displayed a trial at a time. PTB displayed all of the practice trials, but the test trials were split evenly between the two systems. We used two Arduino microcontrollers to control which system presented each trial. The Arduinos communicated with each other via a serial connection, and each Arduino could communicate with one of the two iMacs via a USB connection. One of the Arduinos, the "master," contained code to randomly generate a trial order, ensuring that equal numbers of trials were run on both systems. This controller was responsible for initiating a new trial: A message was sent to the other Arduino via the serial connection, and both Arduinos simultaneously relayed the start message to their respective computers via the USB connection. The Arduinos also recorded the responses generated by the subject pressing a button. The button devices were connected in parallel to each Arduino board, and both boards relayed the response to their computer via the USB connection, as if the response had been generated on a standard keyboard. The Arduinos sampled the digital ports connected to the response devices every 1 ms.

Results

Analysis overview

Response times were excluded from the analysis if the participant responded incorrectly or if the trial timed out (cutoff at 2,000 ms). The mean of the accuracy for all subjects (excluding practice) was 96.1%, with a range from 84.3% to 98.8%.

In the visual search task that we used, for both target-present and target-absent trials, response times are expected to increase monotonically as more items are presented on the display (increasing set size; Shen & Reingold, 2001; Wang, Cavanagh, & Green, 1994). This linear trend, the *search function*, can be parameterized as the estimated increase in response time for each additional item shown on the display (the linear trend's slope), added to a baseline response time (intercept). For each participant, we estimated the search functions separately for target-present and target-absent trials, and separately for trials presented via PTB and JavaScript. Our primary goal was to estimate the effect of software on the search function parameters as well as the effect of software on the variability of the estimates of the search function parameters.

Bayesian data analysis

We used Bayesian data analysis for all of our analyses. There are numerous reasons to prefer Bayesian data analysis techniques to null hypothesis significance testing (Kruschke, 2011). Our primary motivation in the context of this experiment was to have richer information about the effect of software on the response time distributions. For example, with Bayesian techniques we could determine a distribution of credible values that described the difference in mean response times between the two software packages, giving us an estimate of the magnitude of the difference *and* the uncertainty of this estimate, rather than relying on a p value to indicate whether a particular observed difference was likely to have occurred by chance. This estimate is often summarized by the 95% highest density interval (HDI), which is the range of values of a parameter that contains 95% of the distribution, with all values inside the HDI being more probable than all

values outside the HDI. The 95% HDI tells us what parameter values are most likely, given the model and the observed data.

We built a hierarchical model to describe the parameters of interest in the data. The full model specification is presented in the "Appendix". Conceptually, the model performed a linear regression for each subject, estimating the search function parameters (intercept and slope) in each of the four within-subjects conditions (2 software packages×2 trial types: target present or absent), while simultaneously estimating the group-level distribution for each of the subject-level parameters. This technique has the desirable property of introducing *shrinkage* into the estimates of the individual subjects' parameters, allowing the parameter estimates for individual subjects to mutually inform each other. This is helpful for dealing with noisy data, because it moves outliers toward the group mean (Kruschke, 2011). The parameters of interest for us were the group-level estimates. These describe the overall effects of the software environment across all subjects.

To estimate the parameters of the model, we used Just Another Gibbs Sampler (JAGS; Plummer, 2003) and the runjags R package (Denwood, 2014) for Markov-chain Monte Carlo (MCMC) sampling. The sample consisted of three independent chains, each sampled for 20,000 iterations after an adaptation period of 1,000 iterations and a burn-in period of 4,000 iterations. We assessed the convergence of the chains for each parameter of interest via the Gelman–Rubin test (Gelman & Rubin, 1992). The R values were less than 1.025 for all parameters. The full model specification, in JAGS format, and the complete MCMC sample, in .Rdata format, are available online at http://hdl.handle.net/2022/19253.

Estimates of the search functions

Given that there are multiple instances of this exact visual search task in the literature (Shen & Reingold, 2001; Wang et al., 1994), our first analysis was simply to verify that we found search functions similar to those from previous studies (ignoring any possible effects of software). As in the previous research, adding additional items to the search set increased response times, and responses times during target-absent trials were longer than those during target-present trials. The group-level mean estimates for the slope of the search function for target-present trials was 81.5 ms/item (95% HDI: 69.8 to 93.6 ms/item). For target-absent trials, the mean estimate was 110 ms/item (95% HDI: 98.4 to 122 ms/item). The group-level mean estimate for the difference in intercepts between target-present and target-absent trials was 66.5 ms (95% HDI: 32.9 to 98.6 ms), with target-absent trials being longer.

Effects of software package

The analysis model included four parameters (see the "Appendix") that are relevant to potential differences between JavaScript and MATLAB's PTB: $\omega_{(b,\mu,t)_{group}}$, the difference between the search function intercepts for jsPsych and PTB (this can also be conceptualized as the overall average difference in response time measurements between jsPsych and PTB); $\omega_{(m,\mu,t)_{group}}$, the difference in the coefficients (slopes) of set size for jsPsych and PTB; $\omega_{(b,\sigma,t)_{group}}$, the difference in the standard deviations of response times for jsPsych and PTB; and $\omega_{(m,\sigma,t)_{group}}$, the difference in the standard deviations of the coefficients of set size for jsPsych and PTB. The full posterior distribution for each of these parameters is shown in Fig. 3.

We found that jsPsych measured longer response times than did PTB. The 95% HDI of the posterior distribution for $\omega_{(b,\mu,t)_{group}}$ spanned values from 8.18 to 44.5 ms, with a mean value of 26.8 ms. This suggests that JavaScript reports response times that are roughly 10–40 ms longer than those reported by PTB.

The other three parameters of interest showed no reliable difference between jsPsych and PTB. The estimates of the difference in the coefficients of set size (95% HDI: –4.2 to 7.01 ms/item), the difference in the standard deviations of the RT distributions (95% HDI: –9.93 to 21.1 ms), and the difference in the standard deviations of the estimates of set size (95% HDI: –5.71 to 3.13 ms/item) all spanned 0 and had means relatively close to 0.

Validity of the analysis model

Bayesian methods are only able to report the *relative* likelihood of parameter values for a particular model. Therefore, once the most likely parameter values are found, it is important to verify that the model can generate data that are a reasonable approximation of the empirical measurements (Kruschke, 2011, 2013). To examine the predictive validity of our analysis model, we generated credible regression lines from the posterior distributions and plotted them against the group-level data. As is shown in Fig. 4, the credible regression lines capture the overall patterns in the data very well, with the raw means of the data falling on top of the credible regression lines. We conducted a similar analysis for the distributions of subjects' search function parameters, generating credible normal distribution functions to describe the distributions of search function parameters, and found that the analysis model captured these patterns reasonably well.

Discussion

The aim of the present study was to compare human response time data obtained online using JavaScript to those based on a standard laboratory platform for behavioral experimentation. Previous research, using high-precision devices for directly

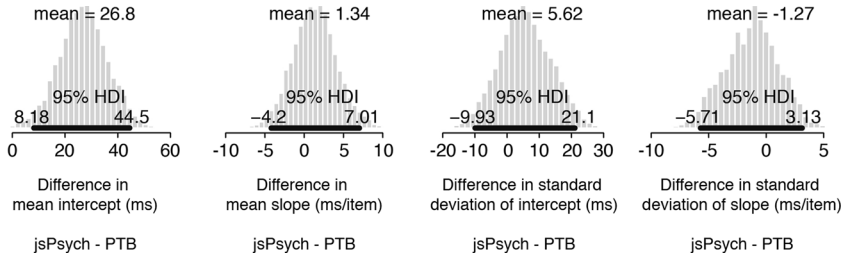

Fig. 3 Posterior estimates of software-relevant parameters. The posterior distribution for each parameter is displayed as a histogram. The 95% highest density interval (HDI) shows the range of values that contains 95% of the distribution, so that the values inside the HDI are more probable than those outside

measuring the timing properties of JavaScript, had predicted that JavaScript would perform comparably to a laboratory platform in a standard response time experiment (Neath et al., 2011; Reimers & Stewart, 2014). To test this prediction, our methodology was designed to eliminate any other possible sources of variance toward this comparison, isolating the software's effect on behavioral outcomes from an established psychophysical experiment. The results revealed that, although JavaScript imposes a slight delay, the online system was nevertheless as sensitive to real human response time differences between conditions as the laboratory platform.

Existing work had already shown that response times with JavaScript are slightly slower than those obtained with PTB (Neath et al., 2011), and our present analysis replicated this result (average difference=26.8 ms). This relative lag is of roughly the same scale as might be caused by other variations

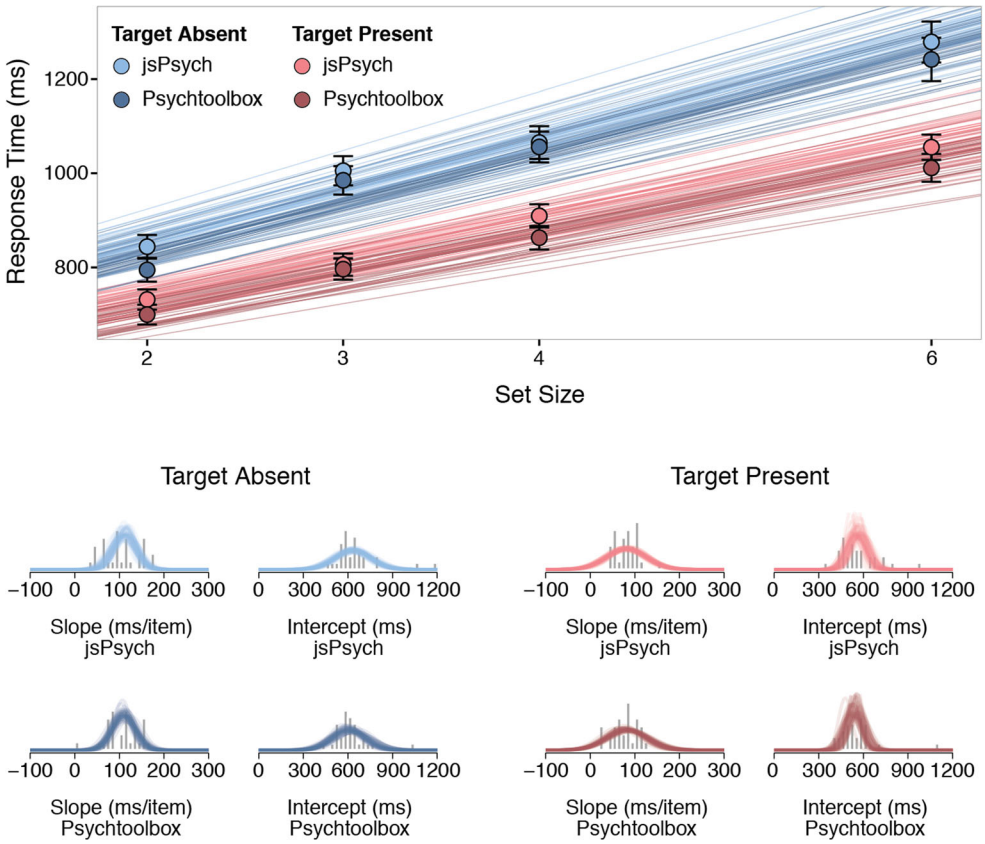

Fig. 4 Posterior predictions of the model. (Top) The group means from the data are shown as circles (error bars are standard errors of the means), colored according to condition. We sampled 50 sets of credible regression lines from the posterior distribution of the model and plotted them using the same color mapping. The model shows a good qualitative fit to the group-level data. In both the target-present (reds) and target-absent (blues) trials, Psychophysics Toolbox (darker lines) tends to have a lower intercept than jsPsych (lighter lines), though there is some overlap. (Bottom) We fit a linear regression to each subject's individual data, estimating the slope and intercept parameters for each of the four cells in our 2×2 design. These data are shown as the gray histograms. We then sampled 25 sets of credible parameters from the model that describe normal distributions that fit the actual data. These are superimposed over the histograms to show the qualitative fit of the model to the distributions of subject-level data. The normal distributions are colored according to the same mapping as in the top part of the figure

in experimental setup, such as a change in keyboard, mouse, or monitor (Lagroix, Yanko, & Spalek, 2012; Neath et al., 2011; Plant & Turner, 2009). Thus, we do not view this effect as a major concern, although experimenters should be aware of it. And although JavaScript may record more variable response times on some slower systems (Neath et al., 2011), it is likely that these effects, if present, would be small (Reimers & Stewart, 2014), and there has been a general feeling that the error introduced by JavaScript would be insufficient to affect the outcome of within-subjects comparisons in a typical human response time study. Considering the growing popularity of online behavioral research, and the importance of precise measurements in response time studies, we felt that this prediction merited empirical investigation. Accordingly, the present study was performed to confirm JavaScript's practical ability to measure response times that are expected to vary as a function of psychological manipulations.

Our analysis showed that, in the context of a real human experiment, JavaScript and PTB recorded nearly identical response time differences between conditions, with constant delays for the JavaScript trials across conditions. In the present task, these measured differences between conditions ranged to about 80 ms, and both systems were equivalently sensitive to such effects. Moreover, considering that the standard deviations of the response times were very similar between the two systems (see Table 1), one might expect that nearly any response time effect that was measurable with a standard implementation of PTB might be similarly detectable using JavaScript, *ceteris paribus*. In sum, our data indicate that the difference between JavaScript and PTB response time measurements is consistent across experimental conditions and that both systems produce response time distributions with practically equal variances.

One potential concern is whether our results are specific to jsPsych, the JavaScript library that we used for creating our experiment, or whether they would generalize to other ways of creating experiments with JavaScript. We believe that the latter alternative is likely. jsPsych provides code for the sorts of things that are common to many different experiments: controlling the flow of trials that a subject sees, storing data, randomizing factors, and so on. The method of recording response times is fairly universal: The time of the system clock is noted when the stimulus is displayed (or very close to it, depending on monitor refresh cycles), and then again when a response is registered. The code for performing this operation would be similar or identical, even if we avoided jsPsych, and thus it is likely that our results will generalize beyond the particular library we used. Another potential concern is whether PTB, on its own, provides a suitable baseline for comparison. Ideally, we might have compared jsPsych with multiple alternatives, evaluating the results from a broader range of software implementations of the visual search experiment. However, PTB is currently the most popular platform for this type of research and is commonly accepted as providing some of the most temporally precise measures of stimulus presentation and response recording (see note 1). Although an

Table 1 Data and model estimates for each cell

Target	Set Size	Software	Data		Analysis Model	
			Mean RT Across Subjects (ms)	Mean Std Dev Across Subjects (ms)	Mean RT [95% HDI] (ms)	Std Dev of RT [95% HDI] (ms)
Present	2	jsPsych	731	154	731 [686 to 779]	148 [130 to 166]
		PTB	699	145	702 [656 to 749]	145 [127 to 163]
	3	jsPsych	805	200	814 [760 to 865]	191 [172 to 211]
		PTB	796	198	783 [730 to 835]	190 [170 to 209]
	4	jsPsych	909	235	896 [835 to 955]	235 [212 to 257]
		PTB	862	236	864 [803 to 923]	234 [212 to 257]
	6	jsPsych	1,055	304	1,060 [978 to 1,140]	321 [290 to 352]
		PTB	1,011	318	1,030 [946 to 1,100]	323 [292 to 354]
Absent	2	jsPsych	843	155	855 [808 to 902]	167 [149 to 186]
		PTB	794	160	826 [779 to 873]	164 [146 to 182]
	3	jsPsych	1,005	199	966 [913 to 1,020]	192 [172 to 211]
		PTB	984	196	935 [882 to 987]	190 [170 to 209]
	4	jsPsych	1,065	217	1,080 [1,020 to 1,140]	216 [194 to 238]
		PTB	1,056	220	1,040 [985 to 1,110]	215 [193 to 237]
	6	jsPsych	1,279	242	1,300 [1,220 to 1,380]	265 [235 to 295]
		PTB	1,241	236	1,260 [1,180 to 1,340]	267 [237 to 297]

HDI, highest density interval

evaluation of multiple software platforms might have been informative, the present study nevertheless provides a stringent and pragmatic test of JavaScript's ability to accurately implement response time experiments.

More recent developments in JavaScript allow for even more precise control over stimulus presentation than we used in this experiment. Some browsers now support the ability to synchronize code with the screen refresh rate of the browser. This is relevant for researchers because it improves the estimate of when stimuli are actually drawn on the screen, thus removing some of the variability in response time measurements (Barnhoorn et al., 2014; Garaizar, Vadillo, & López-de-Ipiña, 2014). This feature is not available for all Web browsers yet, and we did not use it in our experiment code. Thus, our comparisons of the variability of PTB and JavaScript are a lower-bound estimate, with JavaScript being likely to improve the variability of timing measurements as the language improves.

There are many reasons that a researcher might be interested to conduct a response time experiment online using JavaScript. Online experiments enable access to a more diverse subject population than in typical laboratory experiments, data can be collected much faster, and it is typically cheaper to run experiments online. JavaScript may be a better option than such technologies as Flash and Java applets, which require the user to download software to be able to run the experiment, because it is natively supported in all major Web browsers. Furthermore, numerous software libraries for JavaScript have been developed for non experimental purposes and can be very useful for programming online experiments (e.g., Hawkins, 2014), along with libraries developed specifically for behavioral research (e.g., Barnhoorn et al., 2014; de Leeuw, 2014).

However, for all of the practical benefits of online experimentation, there are pitfalls and uncontrollable sources of variance, including variability of the subjects and their computer hardware. It is known that the Internet users recruited for online studies are more demographically diverse than the population that is conventionally sampled for behavioral research (viz. undergraduate psychology students; Arnett, 2008; Henrich, Heine, & Norenzayan, 2010; Ross, Irani, Silberman, Zaldivar, & Tomlinson, 2010). These differences between participants will contribute to increased variance in research data collected online. Moreover, online participants cannot be expected to have identical hardware, and JavaScript's methods can only provide a coarse description of the Internet user's workstation. A researcher may be able to implement calibration tasks that will standardize the visual stimuli to some psychophysically determined threshold or baseline, but even so, different subjects may be seeing different stimuli. Future research will continue to isolate and explore these additional sources of error. Still, even though researchers conducting studies online may be sacrificing some amount of experimental control, any significant effects observed in an online experiment would be far more generalizable than those obtained in a controlled laboratory setting, considering that these would have been detected amidst the noise of a diversity of subjects and implementations.

As behavioral research technologies continue to advance, future methodological innovations will undoubtedly merit critical scrutiny, and the adoption of these new technologies may carry trade-offs between experimental innovation and precision. Our present approach provides a working example of how technological differences between two systems might be isolated and compared in a reasonably naturalistic way. Synthetic devices for directly measuring luminance changes on the surface of a monitor, or immediately depressing keys when a stimulus is detected, have the advantage of being able to precisely quantify the temporal characteristics of stimulus presentation and data acquisition; and published raw data from such studies (e.g., Reimers & Stewart, 2014; see their supplementary materials) should enable researchers to quantitatively evaluate the suitability of JavaScript, and sample size requirements, for specific applications. But there is also value in testing the predictions from these synthetic studies in real, human experiments, building a literature that includes realistic demonstrations of the practical effects of research technologies on research outcomes. The present study provides a novel method for directly estimating the difference between two technologies at a human scale and evaluating whether these technologies will affect a researcher's ability to measure differences related to the experimental manipulations. We hope this enables researchers to empirically validate new technologies for specific research applications, rather than becoming hampered by concerns about trivial differences that might be irrelevant in practical experimental contexts.

Whether in the laboratory or online, many factors introduce error in the measurement of response times. The present study has isolated and described the error associated with one such factor, the relative effects of conducting an experiment using JavaScript. Our results demonstrate that JavaScript, in spite of introducing a slight lag in response times, offers suitable sensitivity for the measurement of response time differences between conditions in common psychophysical research. There may yet be plenty of reasons not to run psychophysics experiments online, but JavaScript should not be one of them.

Author note We thank Richard Viken for providing laboratory space to conduct the experiment, Chris Eller and the IU Advanced Visualization Laboratory for equipment and technical support, Michael Bailey for assistance in the data collection, John Kruschke for suggesting improvements to our analysis model, and Tony Walker and Alex Shroyer for assistance with the Arduino platform. This material is based on work that

was supported by a National Science Foundation Graduate Research Fellowship under Grant No. DGE-1342962.

Appendix: Analysis model

We model an individual response time from trial i, y_i, as coming from a lognormal distribution[2] that is specific for the subject, s_i; trial type (target present vs. absent), t_i; and software package, p_i, associated with that trial:

$$y_i \tilde{} \ln\mathcal{N}\left(\kappa_{s_i,t_i,p_i}, \eta_{s_i,t_i,p_i}\right).$$

The lognormal distribution is parameterized by location, κ, and shape, η, parameters; however, our main interest in conducting the analysis was to understand how the mean, μ, and standard deviation, σ, of the original (non-log-transformed) response time data were affected by the software package. To make this conceptually clear in the model, we constructed our model to estimate parameters in the scale of the original data, and then transformed these parameters into the location and shape parameters of the lognormal distribution:

$$\begin{aligned}\kappa_{s_i,t_i,p_i} &= \ln\left(\frac{\mu^2_{s_i,t_i,p_i}}{\sqrt{\sigma^2_{s_i,t_i,p_i}+\mu^2_{s_i,t_i,p_i}}}\right), \eta_{s_i,t_i,p_i}\\ &= \sqrt{\ln\left(1+\sigma^2_{s_i,t_i,p_i}/\mu^2_{s_i,t_i,p_i}\right)}.\end{aligned}$$

The regression portion of the model was built to find values of the mean and standard deviation parameters in the scale of the original data. The model estimated intercept, b, and slope, m, parameters (relative to the set size, x) for each unique combination of subject, trial type, and software package:

$$\begin{aligned}\mu_{s_i,t_i,p_i} &= m_{\mu_{s_i,t_i,p_i}} x_i + b_{\mu_{s_i,t_i,p_i}},\\ \sigma_{s_i,t_i,p_i} &= m_{\sigma_{s_i,t_i,p_i}} x_i + b_{\sigma_{s_i,t_i,p_i}}.\end{aligned}$$

[2] We used a lognormal distribution to model the response time data because we were only interested in the mean and variance of the distributions. Since the lognormal can be reparameterized into the mean and variance of the original (non-log-transformed) data, it was a reasonable option that provided a conceptually clear mapping between the model parameters and the basic research question we were asking, yet still acknowledged the skew inherent in response time distributions. Although other distributions, such as an ex-Gaussian, could have been used, the lognormal provided good fits to the data without additional parameters.

The intercept and slope parameters were linear combinations of a subject-level baseline, β, a subject-level estimate of the difference between the software packages, ϕ, and a subject-level estimate of the difference between trial types, λ. To estimate a particular difference (such as the difference in intercepts) between software packages, the model estimated a single subject-level difference parameter, ω, and then half of that parameter value was added to jsPsych trials and half of the parameter value was subtracted from PTB trials. We used this approach because we could then apply a group-level distribution to the difference parameter itself. In addition to this parameter mapping nicely onto our main analysis question (what are the differences between the software packages?), this particular implementation of the model also created shrinkage on the difference parameter directly, improving the estimate of the parameter in noise. We applied this strategy of directly estimating the difference parameter to both the trial type and software package differences.

$$\begin{aligned}
m_{\mu_{s_i,t_i,p_i}} &= \beta_{(\mathrm{m},\mu)_{s_i}} + \lambda_{(\mathrm{m},\mu)_{s_i,t_i}} + \phi_{(\mathrm{m},\mu)_{s_i,p_i}}\\
b_{\mu_{s_i,t_i,p_i}} &= \beta_{(\mathrm{b},\mu)_{s_i}} + \lambda_{(\mathrm{b},\mu)_{s_i,t_i}} + \phi_{(\mathrm{b},\mu)_{s_i,p_i}}\\
m_{\sigma_{s_i,t_i,p_i}} &= \beta_{(\mathrm{m},\sigma)_{s_i}} + \lambda_{(\mathrm{m},\sigma)_{s_i,t_i}} + \phi_{(\mathrm{m},\sigma)_{s_i,p_i}}\\
b_{\sigma_{s_i,t_i,p_i}} &= \beta_{(\mathrm{b},\sigma)_{s_i}} + \lambda_{(\mathrm{b},\sigma)_{s_i,t_i}} + \phi_{(\mathrm{b},\sigma)_{s_i,p_i}}\\
\lambda_{(\mathrm{m},\mu)_{s_i,t_i}} &= \begin{cases} -\omega_{(\mathrm{m},\mu,\mathrm{t})_{s_i}}/2, & t_i \; is' target\; present'\\ \omega_{(\mathrm{m},\mu,\mathrm{t})_{s_i}}/2, & t_i \; is\; 'target\; absent'\end{cases}\\
\lambda_{(\mathrm{m},\sigma)_{s_i,t_i}} &= \begin{cases} -\omega_{(\mathrm{m},\sigma,\mathrm{t})_{s_i}}/2, & t_i \; is\; 'target\; present'\\ \omega_{(\mathrm{m},\sigma,\mathrm{t})_{s_i}}/2, & t_i \; is\; 'target\; absent'\end{cases}\\
\lambda_{(\mathrm{b},\mu)_{s_i,t_i}} &= \begin{cases} -\omega_{(\mathrm{b},\mu,\mathrm{t})_{s_i}}/2, & t_i \; is\; 'target\; present'\\ \omega_{(\mathrm{b},\mu,\mathrm{t})_{s_i}}/2, & t_i \; is\; 'target\; absent'\end{cases}\\
\lambda_{(\mathrm{b},\sigma)_{s_i,t_i}} &= \begin{cases} -\omega_{(\mathrm{b},\sigma,\mathrm{t})_{s_i}}/2, & t_i \; is\; 'target\; present'\\ \omega_{(\mathrm{b},\sigma,\mathrm{t})_{s_i}}/2, & t_i \; is\; 'target\; absent'\end{cases}\\
\phi_{(\mathrm{m},\mu)_{s_i,p_i}} &= \begin{cases} -\omega_{(\mathrm{m},\mu,\mathrm{p})_{s_i}}/2, & p_i \; is\; 'P\,T\,B'\\ \omega_{(\mathrm{m},\mu,\mathrm{p})_{s_i}}/2, & p_i \; is\; 'jsPsych'\end{cases}\\
\phi_{(\mathrm{m},\sigma)_{s_i,p_i}} &= \begin{cases} -\omega_{(\mathrm{m},\sigma,\mathrm{p})_{s_i}}/2, & p_i \; is\; 'P\,T\,B'\\ \omega_{(\mathrm{m},\sigma,\mathrm{p})_{s_i}}/2, & p_i \; is\; 'jsPsych'\end{cases}\\
\phi_{(\mathrm{b},\mu)_{s_i,p_i}} &= \begin{cases} -\omega_{(\mathrm{b},\mu,\mathrm{p})_{s_i}}/2, & p_i \; is\; 'P\,T\,B'\\ \omega_{(\mathrm{b},\mu,\mathrm{p})_{s_i}}/2, & p_i \; is\; 'jsPsych'\end{cases}\\
\phi_{(\mathrm{b},\sigma)_{s_i,p_i}} &= \begin{cases} -\omega_{(\mathrm{b},\sigma,\mathrm{p})_{s_i}}/2, & p_i \; is\; 'P\,T\,B'\\ \omega_{(\mathrm{b},\sigma,\mathrm{p})_{s_i}}/2, & p_i \; is\; 'jsPsych'\end{cases}
\end{aligned}$$

The subject-level parameters were modeled as coming from higher-level group distributions. These group-level parameters that describe the distribution of subject-level parameters were the main parameters of interest for our analysis.

$$\beta_{(\mathrm{m},\mu)_{s_i}} \sim \mathcal{N}\left(\beta_{(\mathrm{m},\mu)_{\mathrm{group}}}, \beta_{(\mathrm{m},\mu)_{\mathrm{group}_\sigma}}\right)$$
$$\beta_{(\mathrm{m},\sigma)_{s_i}} \sim \mathcal{N}\left(\beta_{(\mathrm{m},\sigma)_{\mathrm{group}}}, \beta_{(\mathrm{m},\sigma)_{\mathrm{group}_\sigma}}\right)$$
$$\beta_{(\mathrm{b},\mu)_{s_i}} \sim \mathcal{N}\left(\beta_{(\mathrm{b},\mu)_{\mathrm{group}}}, \beta_{(\mathrm{b},\mu)_{\mathrm{group}_\sigma}}\right)$$
$$\beta_{(\mathrm{b},\sigma)_{s_i}} \sim \mathcal{N}\left(\beta_{(\mathrm{b},\sigma)_{\mathrm{group}}}, \beta_{(\mathrm{b},\sigma)_{\mathrm{group}_\sigma}}\right)$$

$$\omega_{(\mathrm{m},\mu,\mathrm{t})_{s_i}} \sim \mathcal{N}\left(\omega_{(\mathrm{m},\mu,\mathrm{t})_{\mathrm{group}}}, \omega_{(\mathrm{m},\mu,\mathrm{t})_{\mathrm{group}_\sigma}}\right)$$
$$\omega_{(\mathrm{m},\sigma,\mathrm{t})_{s_i}} \sim \mathcal{N}\left(\omega_{(\mathrm{m},\sigma,\mathrm{t})_{\mathrm{group}}}, \omega_{(\mathrm{m},\sigma,\mathrm{t})_{\mathrm{group}_\sigma}}\right)$$
$$\omega_{(\mathrm{b},\mu,\mathrm{t})_{s_i}} \sim \mathcal{N}\left(\omega_{(\mathrm{b},\mu,\mathrm{t})_{\mathrm{group}}}, \omega_{(\mathrm{b},\mu,\mathrm{t})_{\mathrm{group}_\sigma}}\right)$$
$$\omega_{(\mathrm{b},\sigma,\mathrm{t})_{s_i}} \sim \mathcal{N}\left(\omega_{(\mathrm{b},\sigma,\mathrm{t})_{\mathrm{group}}}, \omega_{(\mathrm{b},\sigma,\mathrm{t})_{\mathrm{group}_\sigma}}\right)$$

$$\omega_{(\mathrm{m},\mu,\mathrm{p})_{s_i}} \sim \mathcal{N}\left(\omega_{(\mathrm{m},\mu,\mathrm{p})_{\mathrm{group}}}, \omega_{(\mathrm{m},\mu,\mathrm{p})_{\mathrm{group}_\sigma}}\right)$$
$$\omega_{(\mathrm{m},\sigma,\mathrm{p})_{s_i}} \sim \mathcal{N}\left(\omega_{(\mathrm{m},\sigma,\mathrm{p})_{\mathrm{group}}}, \omega_{(\mathrm{m},\sigma,\mathrm{p})_{\mathrm{group}_\sigma}}\right)$$
$$\omega_{(\mathrm{b},\mu,\mathrm{p})_{s_i}} \sim \mathcal{N}\left(\omega_{(\mathrm{b},\mu,\mathrm{p})_{\mathrm{group}}}, \omega_{(\mathrm{b},\mu,\mathrm{p})_{\mathrm{group}_\sigma}}\right)$$
$$\omega_{(\mathrm{b},\sigma,\mathrm{p})_{s_i}} \sim \mathcal{N}\left(\omega_{(\mathrm{b},\sigma,\mathrm{p})_{\mathrm{group}}}, \omega_{(\mathrm{b},\sigma,\mathrm{p})_{\mathrm{group}_\sigma}}\right)$$

The group-level parameters had diffuse priors appropriate to the scale of the data.

$$\beta_{(\mathrm{b},\mu)_{\mathrm{group}}} \sim \mathcal{U}(0,\ 2000)$$
$$\beta_{(\mathrm{b},\mu)_{\mathrm{group}_\sigma}} \sim \mathcal{U}(0,\ 577.35)$$
$$\beta_{(\mathrm{b},\sigma)_{\mathrm{group}}} \sim \mathcal{U}(0,\ 577.35)$$
$$\beta_{(\mathrm{b},\sigma)_{\mathrm{group}_\sigma}} \sim \mathcal{U}(0,\ 166.7)$$
$$\beta_{(\mathrm{m},\mu)_{\mathrm{group}}} \sim \mathcal{N}\left(0,\ 10^5\right)$$
$$\beta_{(\mathrm{m},\mu)_{\mathrm{group}_\sigma}} \sim \mathcal{U}\left(0,\ 10^5\right)$$
$$\beta_{(\mathrm{m},\sigma)_{\mathrm{group}}} \sim \mathcal{N}\left(0,\ 10^5\right)$$
$$\beta_{(\mathrm{m},\sigma)_{\mathrm{group}_\sigma}} \sim \mathcal{U}\left(0,\ 10^5\right)$$

All deflection parameters had the same priors.

$$\omega_{(\ldots,\ldots,\ldots)_{\mathrm{group}}} \sim \mathcal{N}\left(0,\ 10^5\right)$$
$$\omega_{(\ldots,\ldots,\ldots)_{\mathrm{group}_\sigma}} \sim \mathcal{U}\left(0,\ 10^5\right)$$

References

Arnett, J. J. (2008). The neglected 95%: Why American psychology needs to become less American. *American Psychologist, 63,* 602–614. doi:10.1037/0003-066X.63.7.602

Barnhoorn, J. S., Haasnoot, E., Bocanegra, B. R., & van Steenbergen, H. (2014). QRTEngine: An easy solution for running online reaction time experiments using Qualtrics. *Behavior Research Methods*. doi:10.3758/s13428-014-0530-7

Brainard, D. H. (1997). The Psychophysics Toolbox. *Spatial Vision, 10,* 433–436. doi:10.1163/156856897X00357

Crump, M. J. C., McDonnell, J. V., & Gureckis, T. M. (2013). Evaluating Amazon's Mechanical Turk as a tool for experimental behavioral research. *PloS One, 8,* e51382. doi:10.1371/journal.pone.0057410

De Clercq, A., Crombez, G., Buysse, A., & Roeyers, H. (2003). A simple and sensitive method to measure timing accuracy. *Behavior Research Methods, Instruments, & Computers, 35,* 109–115. doi:10.3758/BF03195502

de Leeuw, J. R. (2014). jsPsych: A JavaScript library for creating behavioral experiments in a Web browser. *Behavior Research Methods, 47*(1), 1–12. doi:10.3758/s13428-014-0458-y

Denwood, M. J. (2014). runjags: Interface utilities, parallel computing methods and additional distributions for MCMC models in JAGS [Software]. Retrieved from http://cran.r-project.org/web/packages/runjags/index.html

Francis, G. (2013). Replication, statistical consistency, and publication bias. *Journal of Mathematical Psychology, 57,* 153–169. doi:10.1016/j.jmp.2013.02.003

Garaizar, P., Vadillo, M. A., & López-de-Ipiña, D. (2014). Presentation accuracy of the web revisited: Animation methods in the HTML5 era. *PloS One, 9,* e109812. doi:10.1371/journal.pone.0109812

Gelman, A., & Rubin, D. (1992). Inference from iterative simulation using multiple sequences. *Statistical Science, 7,* 457–472.

Hawkins, R. X. D. (2014). Conducting real-time multiplayer experiments on the web. *Behavior Research Methods*. doi:10.3758/s13428-014-0515-6

Henrich, J., Heine, S. J., & Norenzayan, A. (2010). The weirdest people in the world? *Behavioral and Brain Sciences, 33,* 61–83. doi:10.1017/S0140525X0999152X. disc. 83–135.

Kleiner, M., Brainard, D., & Pelli, D. (2007). What's new in Psychtoolbox-3? *Perception*, *36*(ECVP Abstract Supplement).

Kruschke, J. K. (2011). *Doing Bayesian data analysis: A tutorial with R and BUGS* (1st ed.). Orlando: Academic Press.

Kruschke, J. K. (2013). Posterior predictive checks can and should be Bayesian: Comment on Gelman and Shalizi, "Philosophy and the practice of Bayesian statistics.". *British Journal of Mathematical and Statistical Psychology, 66,* 45–56. doi:10.1111/j.2044-8317.2012.02063.x

Lagroix, H. E. P., Yanko, M. R., & Spalek, T. M. (2012). LCDs are better: Psychophysical and photometric estimates of the temporal characteristics of CRT and LCD monitors. *Attention, Perception, & Psychophysics, 74,* 1033–1041. doi:10.3758/s13414-012-0281-4

Musch, J., & Reips, U.-D. (2000). A brief history of Web experimenting. In M. H. Birnbaum (Ed.), *Psychological experiments on the Internet* (pp. 61–87). San Diego: Academic Press.

Neath, I., Earle, A., Hallett, D., & Surprenant, A. M. (2011). Response time accuracy in Apple Macintosh computers. *Behavior Research Methods, 43,* 353–362. doi:10.3758/s13428-011-0069-9

Nosek, B. A., Banaji, M., & Greenwald, A. G. (2002). Harvesting implicit group attitudes and beliefs from a demonstration web site. *Group Dynamics: Theory, Research, and Practice, 6,* 101–115. doi:10.1037/1089-2699.6.1.101

Pelli, D. G. (1997). The VideoToolbox software for visual psychophysics: Transforming numbers into movies. *Spatial Vision, 10,* 437–442. doi:10.1163/156856897X00366

Plant, R. R., & Turner, G. (2009). Millisecond precision psychological research in a world of commodity computers: New hardware, new problems? *Behavior Research Methods, 41,* 598–614. doi:10.3758/BRM.41.3.598

Plummer, M. (2003). JAGS: A program for analysis of Bayesian graphical models using Gibbs sampling. In K. Hornik, F. Leisch, & A. Zeileis (Eds.), *Proceedings of the 3rd International Workshop on Distributed Statistical Computing* (pp. 1–10). Retrieved from www.r-project.org/conferences/DSC-2003/Proceedings/

Reimers, S., & Stewart, N. (2007). Adobe Flash as a medium for online experimentation: A test of reaction time measurement capabilities. *Behavior Research Methods, 39,* 365–370. doi:10.3758/BF03193004

Reimers, S., & Stewart, N. (2008). Using Adobe Flash Lite on mobile phones for psychological research: Reaction time measurement reliability and interdevice variability. *Behavior Research Methods, 40,* 1170–1176. doi:10.3758/BRM.40.4.1170

Reimers, S., & Stewart, N. (2014). Presentation and response timing accuracy in Adobe Flash and HTML5/JavaScript Web experiments. *Behavior Research Methods*. doi:10.3758/s13428-014-0471-1

Ross, J., Irani, L., Silberman, M. S., Zaldivar, A., & Tomlinson, B. (2010). Who are the crowdworkers? Shifting demographics in Mechanical Turk. In *Proceedings of the 28th International Conference on Human Factors in Computing Systems, CHI 2010* (pp. 2863–2872). New York, NY: ACM. doi:10.1145/1753846.1753873

Schubert, T. W., Murteira, C., Collins, E. C., & Lopes, D. (2013). ScriptingRT: A software library for collecting response latencies in online studies of cognition. *PloS One, 8,* e67769. doi:10.1371/journal.pone.0067769

Shen, J., & Reingold, E. M. (2001). Visual search asymmetry: The influence of stimulus familiarity and low-level features. *Perception & Psychophysics, 63,* 464–475. doi:10.3758/BF03194413

Simcox, T., & Fiez, J. A. (2014). Collecting response times using Amazon Mechanical Turk and Adobe Flash. *Behavior Research Methods, 46,* 95–111. doi:10.3758/s13428-013-0345-y

Wang, Q., Cavanagh, P., & Green, M. (1994). Familiarity and pop-out in visual search. *Perception & Psychophysics, 56,* 495–500. doi:10.3758/BF03206946

Wolfe, J. M. (1998). What can 1 million trials tell us about visual search? *Psychological Science, 9,* 33–39. doi:10.1111/1467-9280.00006

Wolfe, J. M., & Horowitz, T. S. (2004). What attributes guide the deployment of visual attention and how do they do it? *Nature Reviews Neuroscience, 5,* 495–501. doi:10.1038/nrn1411

Zwaan, R. A., & Pecher, D. (2012). Revisiting mental simulation in language comprehension: Six replication attempts. *PloS One, 7,* e51382. doi:10.1371/journal.pone.0051382