Mousetrap-Web: Mouse-tracking for the browser

Felix Henninger
University of Mannheim
University of Koblenz-Landau

Pascal J. Kieslich
University of Mannheim

Corresponding author Felix Henninger
mailbox@felixhenninger.com
Mannheimer Zentrum für
Europäische Sozialforschung (MZES)
A5, 6 (section A)
68159 Mannheim, Germany

Abstract

Mouse-tracking is a versatile method for monitoring the development of cognitive processes over time, particularly the commitment to and the degree of conflict between response options in forced choice tasks. So far, this method has been limited to laboratory-based software, and not easily been available to researchers looking to conduct studies online. Thus, researchers interested in mouse-tracking have had to forgo the advantages that internet-based research offers, such as the quick and efficient collection of larger and more diverse samples. As a solution, we introduce the mousetrap plugin for lab.js, a free and open-source, online study builder. It provides a graphical interface for constructing experiments without requiring programming skills, and allows for the easy implementation of browser-based mouse-tracking studies. Mousetrap-web integrates with the mousetrap R package for processing, analysis and visualization of the collected data. We also address several methodological challenges of moving beyond the laboratory and collecting mouse-tracking data in self-administered online studies, where the individual participants' hardware and environment are not as easily controlled. Finally, we show through technical validation that our software reliably records mouse movements in different browsers. Mousetrap-web is available free of charge, with source code openly available from `https://github.com/felixhenninger/mousetrap-web`.

Introduction

Mouse-tracking is a recent addition to the family of process tracing methods, which offer insight into the mental processes underlying any observed decision. By monitoring and recording the cursor trajectory as participants select between options presented onscreen, mouse-tracking enables inferences concerning the overall degree of conflict and coherence present in a choice, as well as their development over time (Freeman, 2018; Spivey & Dale, 2006). The recorded cursor coordinates are taken to reflect the position of the computer mouse, or a finger on a touchscreen or -pad, and the inference from the motor to cognitive processes is justified based on a presumed close link between both: "Hand in motion reveals mind in motion" (Freeman, Dale, & Farmer, 2011; see also Cisek & Kalaska, 2005; Song & Nakayama, 2009, for evidence of a close neural connection). Over the past years, mouse-tracking has enjoyed a surge in popularity likely due to the fact that it provides process tracing without specialized equipment, and its increased use has made possible more precise tests and refinement of psychological theories in many domains (see reviews by Freeman, 2018; Stillman, Shen, & Ferguson, 2018).

A prototypical mouse-tracking task presents participants with a repeated choice between multiple (often two) alternatives, with every trial consisting of several stages: First, and prior to the actual task, an initial screen presents all but the critical information necessary for completing it, allowing participants to familiarize themselves with the task data before mouse-tracking starts. For example, in a classic task by Dale, Kehoe, and Spivey (2007), participants were asked to sort animal species into one of two categories ('reptile' or 'mammal'), and these were shown onscreen for a two-second interval before participants were able to continue. After this interval, the start button appears, allowing participants to begin the actual choice task with a click. This button also serves to start the trajectory recording, and, through its placement, ensures that the starting position of the cursor (and mouse) is roughly aligned between trials. To prevent the decision process from occurring before this point, and in order to capture as much of the decision process through the ensuing mouse-tracking, critical information is withheld until the onset of the tracking. In Dale et al., for example, the to-be-classified exemplar (e.g. 'cat') was not shown until participants clicked the start button. To maximize the amount and information content of tracking data, the starting point and response options are typically separated as far on the screen as possible, and placed at its edges: In the prototypical configuration, the start button is located in the bottom center, and the responses in the top left and right screen corners. As participants now make their way from the initial position to their option of choice, their cursor positions are continuously recorded, and participants often indicate their decision with a click, completing the trial as well as the logging of mouse positions.

To test psychological theories, many mouse-tracking studies vary an aspect of the task as a within-participant factor, often with a directed hypothesis that one condition will lead to a greater degree of conflict, that, in turn, should be reflected in the cursor movements. Dale et al. (2007), for example, vary the typicality of the animals for their taxonomic class, with the expectation that atypical exemplars (e.g. 'whale', given the response options 'mammal' and 'fish') will lead to continuous activation for both response categories, and a stronger competition between response options as compared to typical exemplars (e.g. 'cat', for the response categories 'mammal' and 'reptile'). Accordingly, they hypothesize and demonstrate empirically that the mouse trajectories are more strongly attracted and therefore closer in space to the alternative option for atypical than for typical exemplars.

Several analysis approaches have been applied to the collected tracking data. The most common approach, broadly speaking, has been to summarize a particular characteristic of every observed trajectory into a single numeric measure, with different indices aiming to capture different aspects of the decision process (Freeman & Ambady, 2010; Koop & Johnson, 2011; Kieslich, Henninger, Wulff, Haslbeck, & Schulte-Mecklenbeck, 2019). For example, several indices reflect the degree to which trajectories are curved toward the non-chosen alternative as opposed to being directed straight to their endpoint (e.g., the maximum absolute deviation, MAD, of the observed trajectory from a direct path); others focus on their complexity by, for example, counting the changes in direction. For example, a series of replications of the experiment by Dale et al. (2007) consistently showed the effect that MAD values were significantly larger for atypical than for typical exemplars (Kieslich & Henninger, 2017; Kieslich, Schoemann, Grage, Hepp, & Scherbaum, 2019). The recent years have seen the development of additional, advanced analysis approaches, one, for example, focussing on the development of trajectory features over time and treating trajectories more as a time series than a monolithic entity (Scherbaum & Dshemuchadse, 2019). Another approach has dropped the assumption of continuous indices in favor of assigning similar trajectories to distinct types, and examining frequency distributions across categories (Wulff, Haslbeck, Kieslich, Henninger, & Schulte-Mecklenbeck, 2019). Many more advanced analysis and visualization approaches have been proposed for mouse-tracking data that are described elsewhere in far greater detail (Calcagnì, Lombardi, & Sulpizio, 2017; Dotan, Pinheiro-Chagas, Al Roumi, & Dehaene, 2019; Heck, Erdfelder, & Kieslich, 2018; Hehman, Stolier, & Freeman, 2015; Maldonado, Dunbar, & Chemla, 2019; Schulz, Speekenbrink, & Krause, 2018; Zgonnikov, Aleni, Piiroinen, O'Hora, & di Bernardo, 2017).

Over the past decade-and-a-half, mouse-tracking has been fruitfully applied to a wide range of tasks and substantive questions, ranging across a spectrum of research domains from language processing and basic perception to much more complex tasks including social cognition and decision making (for reviews, see Freeman, 2018;

Freeman et al., 2011; Stillman et al., 2018). While the majority of mouse-tracking studies has been conducted in the laboratory, there is a growing demand for running studies online, to enable the efficient collection of more diverse samples. In the current paper, we therefore introduce the mousetrap plugin for the free, open-source online study builder lab.js. This enables researchers to create browser-based mouse-tracking experiments via a graphical interface. Before introducing the software in detail, we will discuss past mouse-tracking tools and outline challenges when conducting mouse-tracking studies online.

Mouse-tracking tools

While mouse-tracking is undoubtedly among the process tracing methods that are easier to implement, it is hardly trivial, and frequently not supported in experimental software by default. Thus, with its growing popularity, researchers have developed specialized tools that make mouse-tracking data collection and analysis much more accessible. They have freed researchers from having to implement data collection and analysis logic anew for every project, and provided a common foundation for the field to build upon.

The first published tool was *MouseTracker* (Freeman & Ambady, 2010), which combined experiment design, data collection and analysis in a single, general-purpose mouse-tracking software package. Without doubt, MouseTracker and its ability to quickly and easily put together mouse-tracking experiments has been instrumental to the spread of mouse-tracking, and its availability without charge has contributed to the method's rapid growth in popularity to the degree that method and tool have even been treated as synonymous. However, researchers looking for the level of control and flexibility of general-purpose experimental software, the ability to inspect, adapt and extend the source code, or those using an operating system besides Windows, have required other options.

As an alternative to a stand-alone tool for mouse-tracking, the aim of the *mousetrap* suite (created by the present authors, Kieslich & Henninger, 2017) was to tie into existing software for data collection and analysis, building upon modular tools that many researchers are already familiar with, by adding mouse-tracking functionality as another of these toolkits' wide-ranging capabilities. Its data collection component, mousetrap-os, extends the OpenSesame experiment builder (Mathôt, Schreij, & Theeuwes, 2012), allowing mouse-tracking experiments to be constructed using its graphical interface, and combined with its myriad other features. For analysis, the mousetrap package (Kieslich, Henninger, et al., 2019) for the R programming environment (R Core Team, 2020) handles loading and pre-processing mouse-tracking data from a variety of sources, and provides functions for most common analyses and visualizations. In combination with the frameworks they build upon, both

software packages offer a high degree of customization and are freely available as cross-platform, open-source software.

Mouse-tracking in the browser

In our experience, the vast majority of mouse-tracking studies in the literature has been conducted in a laboratory setting. Behavioral research in general, however, is increasingly moving to online data collection, which promises more efficient access to larger, more diverse samples as well as the opportunity to target specific populations of interest. However, conducting studies online requires them to run in the browser. Recent technical developments spurred by consumers' demand for rich, interactive user experiences on complex websites have made this viable, to the extent that fairly complex studies can now be realized in browsers. Usable both in the laboratory and online, browsers now support advanced designs as well as high-performance stimulus display and response capture, replicating timing-based results in experimental comparisons (de Leeuw & Motz, 2015; Chetverikov & Upravitelev, 2016; Hilbig, 2016; Miller, Schmidt, Kirschbaum, & Enge, 2018; Semmelmann & Weigelt, 2017) and performing favorably in synthetic benchmarks (Reimers & Stewart, 2015; Garaizar & Reips, 2018; Pronk, Wiers, Molenkamp, & Murre, 2019).

With the growing demand for in-browser experimentation, experimental toolkits have vastly facilitated the development of browser-based studies. However, these efforts have not, so far, included mouse-tracking support. Enterprising researchers have therefore coded their own, custom implementations from scratch (e.g. Dale & Duran, 2011; Travers, Rolison, & Feeney, 2016), and Mathur and Reichling (2019) recently introduced and demonstrated an open-source mouse-tracking template for the proprietary Qualtrics survey platform that is adaptable through programming. Beside these tools designed for mouse-tracking in the sense of the experimental paradigm, several further tools capture users' interaction with unstructured web pages for the purposes of user interaction research (e.g. Stieger & Reips, 2010; Diedenhofen & Musch, 2017). All of these, however, require specialized technical knowledge to apply, and are hard to integrate with typical experimental software. Thus, as we describe in the following, we extend these efforts in two regards: First, we endeavor to provide an easily applicable mouse-tracking framework for a general-purpose online experimental toolkit. Second, we attempt to tackle substantive questions surrounding mouse-tracking in online studies, and control variation between different screen and browser window sizes so that the appearance and behavior of the resulting studies is closer to a laboratory task than a regular web-page.

Our aim in this paper is to introduce mousetrap-web, a browser-based mouse-tracking library in the spirit of the mousetrap-os plugin for OpenSesame. Like mousetrap-os, we build on an existing tool for browser-based experiments, lab.js

(Henninger, Shevchenko, Mertens, Kieslich, & Hilbig, in press). It is open-source software available free of charge, providing an easy-to-use graphical user interface for designing experiments and a high-performance stimulus presentation framework. Studies built with lab.js can be run online as well as in a laboratory and integrate with a wide range of hosting platforms, ranging from a personal server or webspace to public data collection services such as Pavlovia or Open Lab, as well as questionnaire-focussed tools such as Qualtrics. Mousetrap-web extends lab.js' capabilities to capture mouse trajectories, and provides several further utilities designed to mitigate differences between participants' computers and ensure a consistent screen layout, participant experience and data format across the wide range of browsers and devices that participants online might use to partake in the study. As is the case with mousetrap-os, the data collected in mousetrap-web is structured to be directly compatible with the mousetrap R package for further analysis, so that the analysis strategies it supports can be applied directly.

Before we describe the application of the software in a brief tutorial, we outline the issues that arise when collecting mouse-tracking data outside of a laboratory context, and the tools mousetrap-web provides to address them. Finally, and also in line with our previous work on mousetrap-os, we validate the performance of mousetrap-web, and show that it is capable of continuously capturing mouse trajectories, subject to technical limitations described below.

Challenges for online mouse-tracking

Researchers using mouse-tracking in the laboratory have come to expect a high degree of control over the specifics of their experimental setup, and typically ensure that data is collected under precisely circumscribed conditions with regard to the stimulus display, system settings pertaining to the mouse, and equivalence of computer hardware more generally (Fischer & Hartmann, 2014; Freeman & Ambady, 2010; Kieslich & Henninger, 2017). By moving to the browser as a data collection platform, and to the world wide web as a medium for reaching participants, researchers cede — to a degree — the level of control they might achieve in a laboratory: First, studies run via the web will almost certainly encounter a vastly greater diversity of client hardware configurations then are present in a typical laboratory, and, second, researchers will not have the level of access to system settings they enjoy on their own computers, thus lacking the ability to standardize the configuration of the computers involved.

These technical variations and restrictions make many common features of mouse-tracking studies challenging to implement online. For example, a typical laboratory mouse-tracking study will be run in fullscreen mode with buttons placed in the corners of the screen to keep movement planning as easy as possible (Grage,

Schoemann, Kieslich, & Scherbaum, 2019). Researchers may also choose to reset the cursor to common coordinates at the onset of the trial to ensure a uniform starting point for all participants. Browser-based studies, on the contrary, often embed the study in a standard web page, leaving the browser interface visible throughout the task (and often with a wide margin around the study), meaning that buttons are not in the screen corners and that users can even accidentally leave the tracking area altogether. In addition, due to lack of control over the mouse cursor, resetting is not typically possible. Previous applications of mouse-tracking online by Mathur and Reichling (2019) and Dale and Duran (2011) provide some initial evidence that mouse-tracking can be conducted successfully even under these non-standard conditions. With mousetrap-web, we certainly support this type of design, and make it easy to implement as we hope to demonstrate below. However, our goal herein has been to go one step further and provide additional features that bridge the gap between laboratory and world wide web with regard to the appearance and behavior of mouse-tracking studies.

The first set of features unique to mousetrap-web addresses the vast diversity of client screen sizes and aspect ratios an online study is likely to encounter. Mousetrap-web accommodates this variability by offering users the option to run studies in fullscreen mode even though they are administered via the browser. This fullscreen mode is realized by creating a custom coordinate system within the study that remains constant for everyone regardless of screen size, and that will be scaled up or down to match participants' screen resolutions. Different aspect ratios are accommodated by adding a margin to the screen within which the study is shown. Mousetrap-web further ensures that cursors cannot enter this margin, keeping response buttons at the edges of the screen and restricting the cursor to the study window.

The second set of mousetrap-web's features are concerned with working around the lack of low-level system access in in-browser experiments as compared to native experimental software (which browsers put in place to protect users from malicious websites). In order to provide the same level of control over the cursor as laboratory studies, mousetrap-web hides the standard cursor and draws its own instead, the style and position of which can be freely chosen and adapted during the study. This also enables resetting of cursor position at trial onset as well as making sure that the cursor cannot leave the study window.

The lack of low-level system access also affects how the browser and, by extension, mousetrap-web, can record cursor movements. Specifically, code running on a web page cannot access the cursor position continuously, but rather receives periodic notifications when the cursor moves[1]. This means that, in the browser, cursor coor-

[1]Mouse-move events, see for example `https://developer.mozilla.org/en-US/docs/Web/Events/mousemove`

dinates can only be registered when movements take place, and that the recorded cursor trajectory may not be continuous. While this will change the data structure compared to laboratory studies, it will not affect the analysis because periods without mouse movements can later be imputed in data processing (the mousetrap R package, for example, offers this functionality). In addition, the update frequency is set by the browser and the operating system, and, at around 60Hz in our tests, it is typically only slightly lower than the typical sampling rate of laboratory mouse-tracking software (60-75Hz in MouseTracker, 100Hz in mousetrap-os). Importantly however, the precision of the recorded data should still be more than sufficient for research use: For example, many mouse-tracking analyses reduce the sampling rate of the recorded trajectories as a first step prior to analysis (e.g., through time-normalization, Spivey, Grosjean, & Knoblich, 2005).

Another result of the lack of access to low-level system data is that settings regarding cursor sensitivity, specifically the cursor speed (i.e., how physical mouse movements are translated into cursor movements) and the degree of cursor acceleration, cannot be monitored or modified, and will depend on the preferences of the participant. If left to their default settings, systems running Windows, for example, will enable cursor acceleration and set cursor speeds to a medium level, which some authors have argued is not optimal for mouse-tracking, recommending instead to disable acceleration and reduce cursor speed (Fischer & Hartmann, 2014). While the actual settings are often not reported, we suspect that previous laboratory-based mouse-tracking studies have used a mixture of default settings and reduced speed with disabled acceleration.

Concerning the question of how strongly variations in the cursor speed and acceleration affect the collected data, two experiments have been published to date which compare different settings and overall suggest that their influence on critical mouse-tracking results is not too strong (Kieslich, Schoemann, et al., 2019; Grage et al., 2019). Kieslich, Schoemann, et al. (2019) show that, while default settings increased mouse trajectory curvature overall compared to reduced cursor speed and disabled acceleration, the size of the psychologically relevant effects was not significantly affected. Grage et al. (2019) report that participants appeared to adapt to different cursor speeds to a certain extent by changing the speed with which they moved the mouse, but also that higher cursor speeds (acceleration was disabled throughout their study) emphasized (and possibly over-emphasized) initial movement tendencies.

A preliminary summary would therefore suggest that different cursor sensitivity settings should not affect results too strongly as long as they are in a comfortable range (and we would expect users to configure their systems in a way that suits them). Still, we would caution that, due to differences in system settings, online mouse-tracking studies are best suited for within- (as opposed to between-) participant

manipulations. However, if it is critical to ensure exactly identical cursor sensitivity settings between studies, studies created with mousetrap-web can also simply be run in the lab — with a precision that approaches dedicated laboratory software, as we hope to demonstrate in the following.

Building a mouse-tracking study with mousetrap-web

To demonstrate the use and functionality of mousetrap-web, the following brief tutorial describes the construction of a full mouse-tracking study, resulting in a simplified version of Study 1 by Dale et al. (2007), which we also used as an example above. As in our introduction of mousetrap for OpenSesame (Kieslich & Henninger, 2017), we chose this study because it combines the prototypical features of a mouse-tracking study: A two-alternative forced-choice task (categorizing animal species into classes) with a within-participant manipulation (of the typicality of an exemplar for its category).

We build the study step-by-step, briefly demonstrating how to construct the overall task in lab.js, then adding basic mouse-tracking functionality and finally demonstrating mousetrap's advanced capabilities for controlling the appearance and behavior of the study across participants.

Before we dive in, let it be noted that further material, including the final result, is available from the project repository at `https://github.com/felixhenninger/mousetrap-web`, and extensive further information regarding the lab.js experimental builder is compiled in its documentation at `https://lab.js.org/docs`. The lab.js software also includes an adaptable mouse-tracking template that provides a starting point for further studies. For the purposes of this tutorial, we will start from scratch, and walk through the entire process of creating a study.

Constructing the task

The initial stage of any mouse-tracking project is to design the task itself in the lab.js experiment builder. The graphical builder interface can be accessed through any recent browser by navigating to `https://lab.js.org/builder/beta/` [2]. Mousetrap-web is available within lab.js, and needs no prior installation to be used in a study. As a plugin, it can be added to any part of the study, and will record mouse-tracking data throughout that part.

After opening the lab.js builder interface in a browser, first-time visitors will be greeted by a welcome screen and an empty study waiting to be filled. Specifically, the left-hand sidebar provides, at its top, buttons to preview and save the study, and

[2] At the time of submission, mousetrap-web is only available in the beta version of lab.js (`https://lab.js.org/builder/beta/`). However, we will add it to the stable release shortly, at which point the functionality will be available through the lab.js builder at `https://lab.js.org/builder/`

the (as yet empty) study timeline. The larger area on the right-hand side initially occupied by the splash screen will serve us to inspect and modify constituent parts of the study (Figure 1).

The first step toward a study is to add a component to the nascent study timeline via the button marked with a plus in the sidebar. A click presents a selection of component types to add to the study. Of these, the most useful for our purposes is the leftmost: A *screen* component that allows us to design a display shown during the study. Selecting this component type adds it to the study.

With the newly added screen component present in the study, the welcome dialogue is now replaced with the component's options, and a graphical stimulus designer becomes visible. It, too, is initially empty, showing only a light grid and a gray rectangle. This area represents the content that lab.js ensures will be visible to participants — studies will be scaled to match the available screen space on any device, but only to the degree that this region will not be cut off.

Content can be added to the screen using the plus button in the toolbar below the visual editor. For a mouse-tracking screen, for example, we might want to add to rectangles in the top left and right screen corners to indicate the edges of the response buttons, add text labels in their center to reflect the available options, and add a stimulus (text, image, or otherwise) in the screen center. Using the blue button in the top left of the interface to open the study preview will show the general screen layout, but closer inspection will quickly reveal that clicks, even on the newly created 'buttons', have no effect: interaction with the screen is not possible yet.

To enable mouse responses to a screen in lab.js, the relevant parts of the screen must be designated areas of interest (AOIs) in order for the software to pick up clicks in the respective region. This is also possible within the graphical editor, in which AOIs can be added to the screen in the same way that other content would be: through the plus button in the visual editor toolbar. Like other content, AOIs can be moved and resized to match any particular experiment — for our purposes, the AOIs are best placed on top of the designated buttons in the upper screen corners.

In order to differentiate between multiple AOIs, labels need to be assigned to each by selecting the corresponding AOI and adding a label via the editor toolbar at the bottom of the screen (Figure 1). These can be set arbitrarily, for example we might choose the values `left` and `right` for the respective response buttons.

As a final step to enable interactivity, lab.js requires users to define the responses available to participants. Doing so requires a switch from the content editor to the behavior tab, which is available in the menu at the top of the builder screen. Here, the Responses section collects the permissible answers (Figure 2), splitting them into a label column, which reflects the substantive meaning of any particular response, and is saved in the data, the action, which indicates the mode of response, and the target and filter columns, which help restrict where and how a response can be

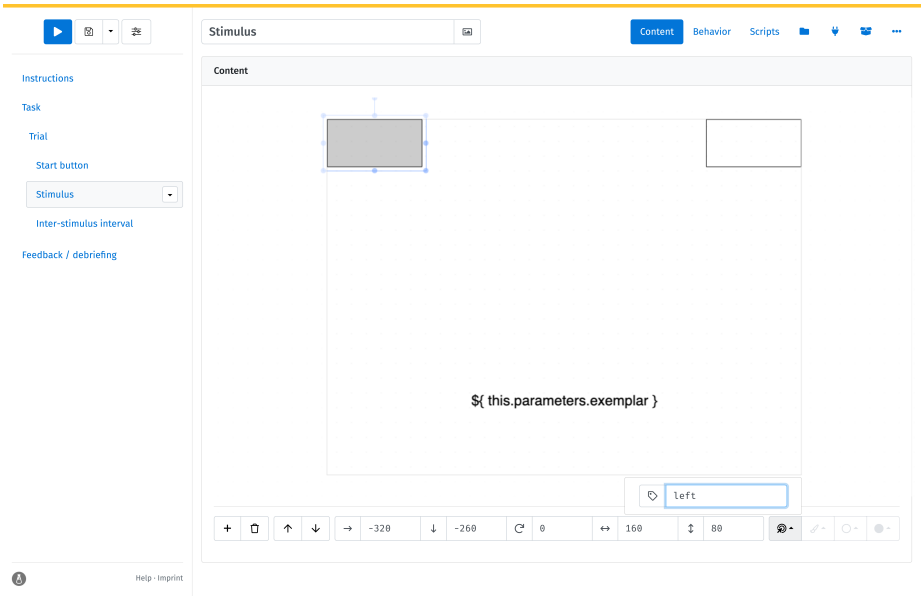

Figure 1: The lab.js builder, with a mouse-tracking screen open for editing. On the left-hand side, the timeline provides an overview of the study structure. On the right, the mouse-tracking screen is shown under construction. In the visual editor, the top-left response area of interest is selected, and the "left" label is assigned to it in the toolbar below.

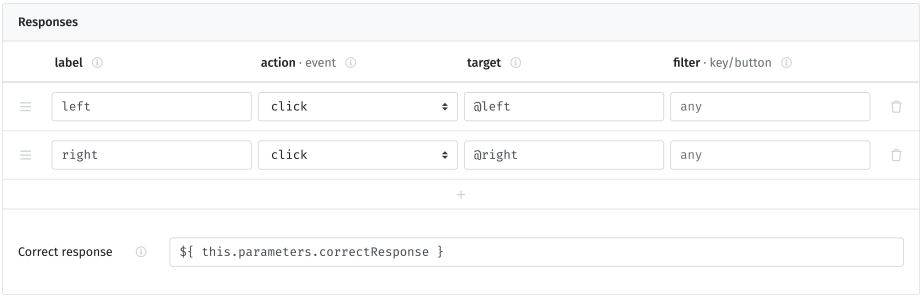

Figure 2: Response settings for mouse-tracking screen constructed in lab.js.

made. For our purposes, we can add two distinct response options with the labels left and right, each capturing a click on the respective button. To limit the click to the corresponding AOI, we can enter the AOI labels in the target column, prefixed with an `@` symbol (i.e., `@left` and `@right`). Finally, we might want to limit clicks to a specific mouse button using the filter column: By entering a `0` here, only clicks with the innermost button count toward responses (this corresponds to the left button on a mouse set up for use with the right hand, and vice versa). The full set of response options is visible in Figure 2, and with all possible responses in place, participants should now be able to respond via mouse click. Since the screen is the only component currently in the study, the click will end the experiment, and the preview will offer the dataset for download.

Closer inspection of the dataset at this point will reveal that the response is logged, but not the mouse trajectory itself. Adding this vital component is likewise possible through the graphical editor, by switching to the plugin tab (indicated in the navigation bar by a plug icon, see Figure 3), and adding the Mousetrap plugin to the screen (Figure 4). After this step, not only will the response be logged in the data, but also the movement trajectory, every measurement splitting the collected data across the `xpos`, `ypos` and `timestamp` columns for the coordinate pairs and timestamps respectively, thus matching the output of the mousetrap-os plugin for OpenSesame and ready for further analysis in the mousetrap R package.

Of course, a useful study will rarely consist of a single screen, and a mouse-tracking trial in particular typically consists of the sequence of screens described above: An initial screen to familiarize participants with the available response options, presented for a fixed amount of time, followed by a start screen for participants to self-pace the begin of the actual mouse-tracking task, and to encourage them to return their cursor to the bottom center of the screen. Only after participants start the task is the screen we have built above shown, the mouse-tracking activated and the information critical to the task revealed. Lastly, after participants have made their choice, an inter-stimulus interval provides a brief pause before re-starting the task with new information. In `lab.js`, these successive phases of the task are implemented as distinct screens in the same manner as the one above, by adding the appropriate screen content and activating the start button through the AOI mechanism, or by adding a timeout to the screen so that the study advances automatically.

Moving from a single trial to an experiment consisting of multiple decisions in sequence is possible through lab.js' flow control components. First, a sequence component can be used to group the multiple successive screens comprising a trial into a block of content. The trial can then be included in a loop component in order to repeat it multiple times with varying stimuli and response options. This process is

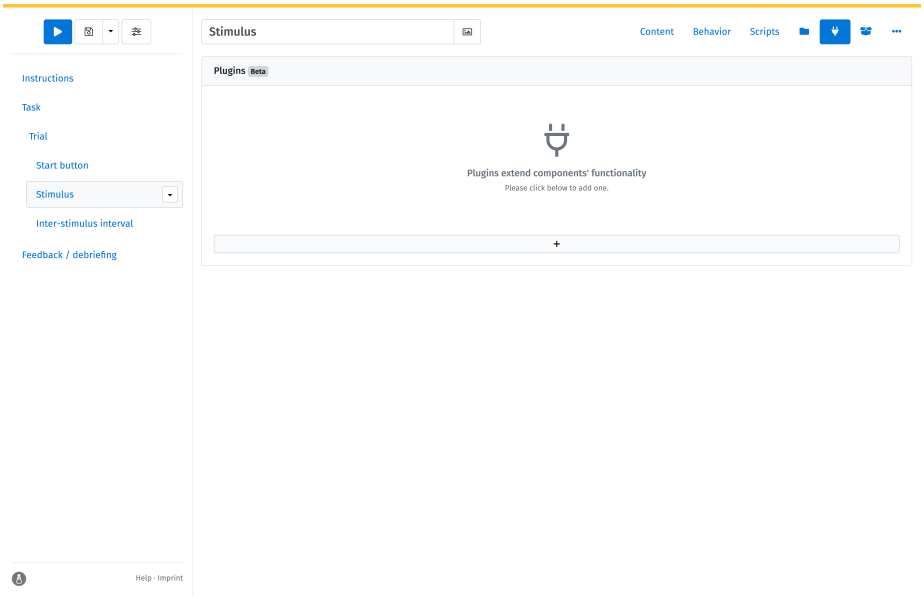

Figure 3: The plugin tab allows users to manage plugins for individual components.

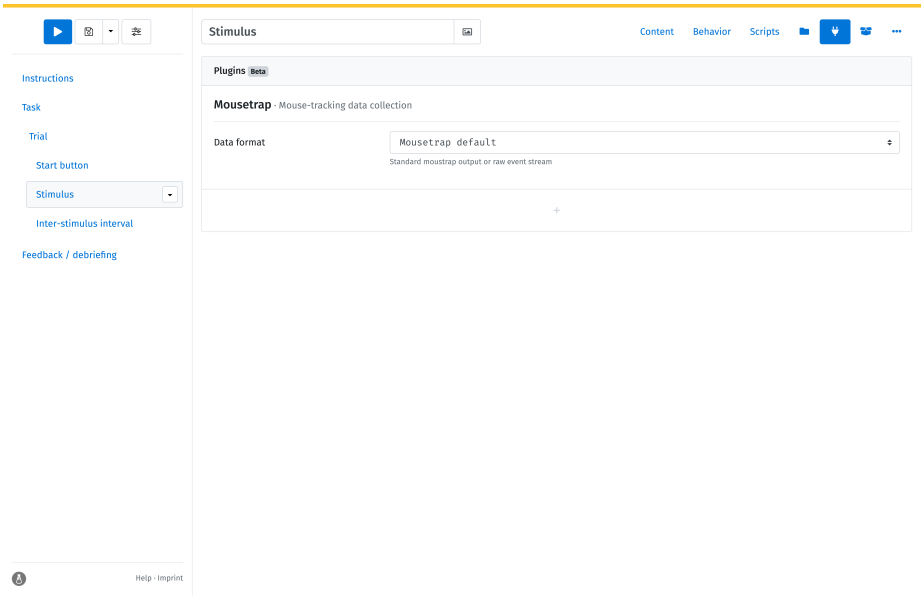

Figure 4: The mousetrap plugin in action on the stimulus screen. With the plugin in place, mouse coordinates are tracked for the duration of the screen.

described in detail in Henninger et al. (in press) as well as the lab.js documentation[3], and while it will be useful to go through the steps for any mouse-tracking experiment, it is not necessary for the purposes of our tutorial, and we do not cover it in detail.

Fine-tuning the study

At this point, the mouse-tracking study is functionally complete, and could, in principle, be used to collect mouse-tracking data — it matches the browser-based mouse-tracking paradigms constructed to date. However, our setup so far omits two standard design features that are present in most laboratory-based mouse-tracking studies. First, the paradigm in its present state would be contained in its browser window, while the overwhelming majority of mouse-tracking studies to date have been run in full-screen mode. Adding a fullscreen plugin to the study changes to full-screen mode for the duration of the affected components hiding other windows that might distract participants, and stretching trajectories across all available screen space. Second, we recommend that users activate the mousetrap overlay plugin in order to take control of the cursor and the screen aspect ratio. This plugin serves the dual purpose of both hiding extraneous screen content and extending response buttons to the screen edges which will catch the mouse cursor, allowing participants to move quickly toward the response button rather than having to slow down for fear of overshooting. While the screen layout we built above is scaled to match participant's screen resolutions, the relative position of the content is fixed, so that, for example, wide-screen displays will show a visible screen border to the left and the right of the response buttons that will also catch the mouse cursor and prevent it from leaving the study area.

With both of these plugins in place, the mouse-tracking experiment should now not only run in full-screen mode, but also, where possible, restrict the visible screen content to a predefined aspect ratio that is constant across screens.

Further methodological considerations

Regardless of whether a mouse-tracking study is run in the laboratory or online, and no matter the software used to implement the study, there are a number of general methodological considerations that researchers creating mouse-tracking experiments need to take into account. We have collected general recommendations previously (Kieslich & Henninger, 2017; Kieslich, Henninger, et al., 2019) and will only summarize the most salient points here.

First, to account for potential differences in left- versus rightward movements, it is important to counterbalance which response option is presented on which side so as not to inadvertently create confounds. Depending on the experimental design,

[3] Please see `https://lab.js.org/docs/en/latest/learn/builder/3-trial.html`

this can be implemented within or between participants, using the general randomization functionality in lab.js. Participants' handedness has also been discussed as a potential influence, with some authors recommending that researchers limit studies to right-handed participants (Hehman et al., 2015). While differences in right- and leftward movements have been found (e.g. Spivey et al., 2005), these have not, to our knowledge, been linked directly to handedness, therefore we believe that this question deserves further research. We do, however, recommend that researchers assess participant's handedness using, for example, a modified version of the Edinburgh Handedness Inventory (Oldfield, 1971; see Kieslich & Henninger, 2017) and also ask participants which hand they used to move the mouse during the study.

A second important design aspect in mouse-tracking studies concerns the so-called starting procedure, which determines the sequence of events around the stimulus onset. In our discussion so far, we have tacitly assumed a *static starting procedure*, meaning that the stimulus was shown immediately after the click on the start button, and participants received no instructions regarding movement initialization. This procedure is the simplest, and used in many studies, including Dale et al. (2007). However, it runs the risk that participants make their decision before starting their movement and, consequently, that their decision process is not optimally reflected in the cursor trajectory. To reduce this risk, some mouse-tracking studies have implemented countermeasures (see Kieslich, Schoemann, et al., 2019, for an overview and comparison). These include limiting the overall time participants have for responding, instructing participants to initiate their movement within a certain time limit, and a *dynamic starting procedure* in which the stimulus is only presented once participants have moved the mouse upwards for a fixed distance. In empirical comparisons, limiting the time for movement initiation and using the dynamic starting procedure in particular have proven to be more effective than a static starting procedure (e.g. Kieslich, Schoemann, et al., 2019; Scherbaum & Kieslich, 2018). However, the implementation of a dynamic starting procedure is challenging in an online setting, as different cursor sensitivity settings make it difficult to fine-tune the dynamic starting procedure: An ideal setting enforces a minimum movement distance to ensure that the mouse is in motion, but simultaneously leaves participants with enough time to acquire the stimulus information before their cursor hits the upper screen border. For similar reasons, enforcing an early movement initialization through a dynamic starting procedure or an explicit instruction may not generally be suitable for more complex tasks, even in a setting where cursor speed can be controlled. For all of these reasons, we would, for the time being, tentatively recommend to rely on a static starting procedure if a mouse-tracking study is conducted online. Like design factors in the laboratory, this is a question warrants empirical investigation. However, the existing evidence suggests that, if the analysis is based on the typical method of mouse-tracking indices, a static starting procedure

is sufficiently sensitive to detect many psychological effects (although effect sizes may be somewhat reduced).

Concerning the response mode, previous studies have required either a click on the response button or allowed responses by moving the cursor into the button area alone ('touch' / 'hover' procedure). Responses via click are most common, and result in stronger curvature effects, though a touch response procedure leads to more homogenous trajectories but should probably be combined with a dynamic starting procedure (Kieslich, Schoemann, et al., 2019).

The discussion around design choices in mouse-tracking experiments is ongoing in the literature, and as in the laboratory, we hope that empirical results will guide implementation decisions for online mouse-tracking experiments in the near future. In the meantime, mousetrap-web aims to support an increasing range of implementation options[4], giving researchers the choice of the most appropriate design whatever their research goals.

Working with the collected data

At this point, we have covered all the necessary ingredients for a complete mouse-tracking paradigm. The hosting of the study is covered in detail elsewhere (see `https://lab.js.org/docs/en/latest/learn/deploy/`), but one more time, the technical intricacies of in-browser mouse-tracking demand special attention.

In the browser, mouse positions are only logged during periods of movement, and no data is collected when the cursor is at rest. Most mouse-tracking analyses, however, assume a continuous time series of mouse coordinates at regular intervals, and artefacts may result if this assumption is not met. Therefore, we recommend that mouse-tracking data collected online be subjected to *resampling* prior to further analysis. This process interpolates the data points as if they were sampled continuously, approximating the data produced by packages with low-level access to the cursor position. The mousetrap R package includes the `mt_resample` function for this purpose.

However, a linear interpolation alone is not sufficient, since browsers do not provide movement updates during periods of rest, and filling in temporal gaps in the data through interpolation may result in the false appearance of ongoing movement. Therefore, we recommend that users apply resampling with the `constant_interpolation` parameter, which provides a threshold duration above which periods without cursor updates are interpreted as episodes without movement, and below which the trajectory is interpolated assuming an ongoing movement. Based on our testing below, we recommend that this threshold be set to two, or alternatively three sampling intervals (32 or 48ms) if a conservative value is desired. This ensures that the trajectory is

[4]The dynamic starting procedure and the hover response mode are currently not supported in mousetrap-web. However, they will be implemented in its next release.

interpolated if the browser drops individual mouse position updates, whereas longer periods without new trajectory data are treated as a cursor hovering in position.

Apart from this caveat, the data generated by the mousetrap-web package matches the format assumed by the mousetrap analysis package, and the analysis methods presented in detail elsewhere can be applied directly (e.g., Kieslich, Henninger, et al., 2019; Wulff et al., 2019).

Technical validation

To judge the quality of the collected data, we conducted a validation study following the example of Freeman and Ambady (2010) and Kieslich and Henninger (2017), who simulated cursor movements using software or external hardware, and compared the collected data to the known trajectories. In the following, we follow the procedure previously used to validate mousetrap-os.

Our simulated data consist of two paths from the start button to the top left response button, either in a straight diagonal line, or a triangular shape taking the cursor to the top of the screen and then to the left. In either case, the cursor moved 800px on both the horizontal and vertical dimension on a screen with a 1680 by 1050 resolution, moving a single pixel (either diagonally, or first vertically and then horizontally) along its preset path every 16ms. This mouse movements were generated by external hardware simulating a mouse[5], and connected to standard laboratory hardware[6]. We tested Firefox (Version 72) and Chrome (79) as the most popular browsers available to date. Our validation experiment was constructed to approximate an actual mouse-tracking task, with a start button to begin the trial, and a capture screen with options in the screen corners (the validation study is available via the project repository, as is the collected data). We captured a sequence of 500 identical trials for every browser and simulated trajectory.

The raw trajectories are visible in Figure 5. As a first more detailed analysis, we examined the sampling rate to assess the software's ability to continuously record the cursor position at regular intervals. We computed sampling intervals between subsequent timestamps expressed and rounded to multiples of the target sampling rate (16ms). The results are visible in Table 1. They show that Chrome met the desired sampling rate (one measurement per 16ms) in more than 95 percent of samples, and that in almost all remaining cases, skipped one sample. For Firefox, performance was slightly lower, with the target sampling rate being met in around 93 percent of samples, and the bulk of the remainder similarly separated by an additional frame.

[5] A Teensy 3.5 microcontroller board (PJRC Inc.) programmed using the Arduino microcontroller development environment.

[6] The computer used for testing was a PC running Windows 10, equipped with an Intel Pentium Dual-Core running at 3 GHz and 4 GB of RAM.

In a second step, we turned to the distance travelled by the cursor between successive samples. Here, we would expect to see a single pixel difference between subsequent coordinate pairs in accordance with the simulated movement. Results are shown in Table 2: For both Chrome and Firefox, the observed positions were one pixel apart in more than 95 percent of cases, and the majority of the remaining cases involved changes of two pixels.

Moving toward a validation of the trajectories as a whole, we determined the correlation between the observed and expected position of the cursor based on the time since trial onset. For both browsers, these correlations were > .9999 in both the diagonal and the triangular simulations.

In a final step, we computed a number of common mouse-tracking curvature indices: maximum absolute deviation (MAD), area under curve (AUC), and average deviation (AD). Table 3 reports the mean values of each measure as well as the expected values. In both browsers and simulations, the mean observed values matched the expected values closely. In addition, the variation of the indices at the trial level was small, with a $SD < 0.3$ px for all measures and browsers (except for AUC values in the triangular simulation on Firefox with a value of 114.6 pixel squared).

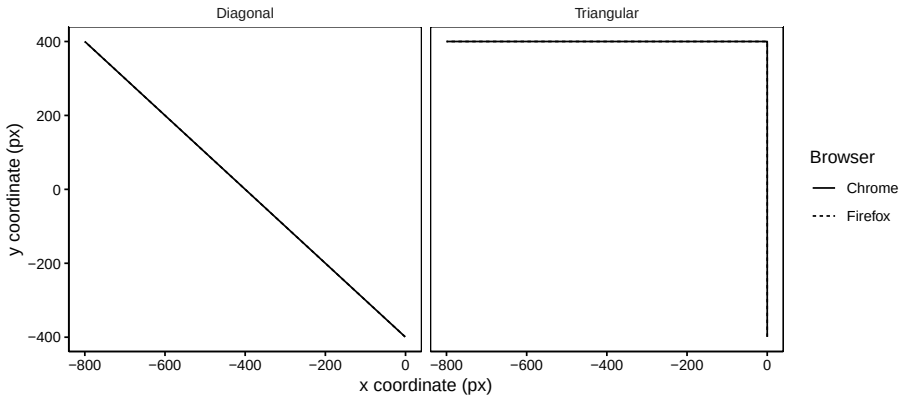

Figure 5: Raw trajectories across both browsers and simulated trajectory shapes.

Table 1: Relative frequency of the sampling intervals per browser and simulation. The sampling interval is expressed in multiples of the target sampling rate (16ms).

Browser	Simulation	0	1	2	3	4	5	>5
Chrome	Diagonal	0.002	0.954	0.044	<.001	<.001	0	0
	Triangular	0.001	0.956	0.043	<.001	<.001	<.001	0
Firefox	Diagonal	0.001	0.930	0.066	0.003	<.001	<.001	<.001
	Triangular	0.001	0.931	0.065	0.003	<.001	0	0

Table 2: Relative frequencies of the distance travelled (in pixels) by the cursor between successive samples per browser and simulation. For the diagonal simulation, changes always occurred both along the x- and y-dimension. For triangular simulation, changes could only occur along one of the dimensions and the corresponding changes are reported.

Browser	Simulation	0	1	2	3	4	5	>5
Chrome	Diagonal	0.001	0.955	0.043	<.001	<.001	0	0
	Triangular	0.001	0.956	0.043	<.001	<.001	0	0
Firefox	Diagonal	0.001	0.952	0.047	<.001	<.001	<.001	<.001
	Triangular	0.001	0.953	0.046	<.001	<.001	0	0

Table 3: Expected and observed mean values for selected mouse-tracking measures per browser and simulation.

	Diagonal			Triangular		
	MAD	AUC	AD	MAD	AUC	AD
Expected	0.00	0.00	0.00	565.33	319600.00	282.49
Chrome	0.00	0.00	0.00	565.30	319599.98	282.33
Firefox	0.00	0.00	0.00	565.27	319563.98	282.46

In sum, the mousetrap-web package is capable of capturing raw trajectory data with high precision, if not at quite the consistency with regard to the sampling rate offered by its native laboratory counterparts. Given the lack of low-level system access and the somewhat roundabout way of capturing cursor coordinates necessary in the browser, this is not entirely surprising. In practice, it implies that minor trajectory details may not be captured due to skipped trajectory logs. If skipped data is part of longer continuous movements, no information is lost at all, though slight deviations might occur when skipped mouse updates coincide with a kink or corner in the trajectory. However, given that most mouse-tracking analyses downsample trajectories as a first processing step, discarding vastly larger amounts of data, we are cautiously optimistic regarding the usability of browser-collected mouse-tracking data.

Discussion

Mousetrap-web is a software package that enables researchers to collect mouse-tracking data in the browser, for use online or in-laboratory. Its integration with the lab.js graphical experiment builder interface makes it possible to construct studies easily, and we have shown that it reliably captures mouse trajectories.

As we have discussed above, online mouse-tracking presents a number of challenges: Moving outside the controlled laboratory environment by necessity results in increased hardware variation between participants, and a substantial portion of our effort has been directed to ensuring a consistent experience and behavior of our software across those aspects we can control, specifically different screen dimensions. This provides a degree of control that has not, to the best of our knowledge, been reached in the browser. However, as we note above, some variation cannot easily be accounted for, specifically all variation linked to participants' input devices and their respective settings. Previous successful applications of mouse-tracking in the browser (Dale & Duran, 2011; Mathur & Reichling, 2019) indicate that this does not preclude the method per se, but we would like to issue a number of recommendations to preempt potential issues: One potential concern is that some of the participants taking part in the study do not use a computer mouse. This could result in a mix of pointer devices within studies which might introduce additional, undesired variation in cursor movements between participants. We therefore strongly recommend that researchers ask participants complete the study using a computer mouse, and that they add a device self-report question to the study to verify that this was indeed the case. Note that we do not mean to imply that the mouse is the preferable input device in every case (several hand movement tracking studies have successfully used others, such as touch screens, e.g. Buc Calderon, Verguts, & Gevers, 2015; Wirth, Pfister, & Kunde, 2016) but that care should be taken to ensure that all partici-

pants use a comparable device, or that differences can be accounted for during the analysis.

Similarly, differences in cursor speed and acceleration are difficult to account for at present – though the existing mouse-tracking literature indicates that cursor speed settings do not have strong effects on the collected mouse-tracking data across a sensible range of values (Kieslich, Schoemann, et al., 2019; Grage et al., 2019). In principle, the overlay we present above could be used to transform not only the cursor position but also its speed. However, we are unaware of a way to determine an appropriate transformation that would reduce unwanted variation in the data, and must leave this as a challenge for future research.

Should any of these concerns become pressing, mousetrap-web can naturally be used in a laboratory setting for situations that require absolute control over the hardware involved, and our validation shows that it produces high-quality data in such an environment. Similarly, when individual differences may plausibly covary with participants' choice of devices, a more controlled environment may be required (Buchanan & Reips, 2001; Götz, Stieger, & Reips, 2017); however, the vast majority of mouse-tracking studies employ within-participant designs that should be largely resilient to such concerns. As a final note, we speculate that a case could be made for the benefits of collecting data on the devices and with the speed settings participants are familiar with, though that is, at present, without empirical foundation.

While we present them in the context of mouse-tracking, the features we introduce such as stimulus scaling, screen overlays and mouse cursor control are more generally useful in standardizing the experience of study participants across varying hardware. The environment that participants inhabit while completing online studies will never be under the complete control of the experimenter; however, these tools serve to standardize studies' visual presentation across varying devices, to ensure consistency and remove distractions, and may be of use to experimenters even if they do not use mouse-tracking.

With mousetrap-web, our goal has been to make mouse-tracking accessible to researchers looking to collect data online, integrating with an in-browser study builder to facilitate study construction, and an established, powerful analysis toolkit for data processing. As we have noted throughout, applying mouse-tracking in an in-browser and online setting comes with technical and practical challenges, and we hope to have provided researchers with the knowledge and tools necessary to tackle them.

Open Practices Statement

The data and materials for all experiments are available through the project repository.

Acknowledgements

The work reported herein was supported by the German Research Foundation project "Statistical Modeling Using Mouse Movements to Model Measurement Error and Improve Data Quality in Web Surveys" (KR2211/5-1), and an Open Science Mini-Grant by Mozilla to the first author. The authors would like to thank Mila Rüdiger and Franziska Leipold for their comments on an earlier version of this manuscript. This work was supported by the University of Mannheim's Graduate School of Economic and Social Sciences, funded by the German Research Foundation.

References

Buc Calderon, C., Verguts, T., & Gevers, W. (2015). Losing the boundary: Cognition biases action well after action selection. *Journal of Experimental Psychology: General*, *144*(4), 737–743. doi:10.1037/xge0000087

Buchanan, T., & Reips, U.-D. (2001). Platform-dependent biases in Online Research: Do Mac users really think different? In K. J. Jonas, P. Breuer, B. Schauenburg, & M. Boos (Eds.), *Perspectives on Internet Research: Concepts and Methods*. Retrieved December 16, 2018, from http://www.uni-konstanz.de/iscience/reips/pubs/papers/Buchanan_Reips2001.pdf

Calcagnì, A., Lombardi, L., & Sulpizio, S. (2017). Analyzing spatial data from mouse tracker methodology: An entropic approach. *Behavior Research Methods*, *49*(6), 2012–2030. doi:10.3758/s13428-016-0839-5

Chetverikov, A., & Upravitelev, P. (2016). Online versus offline: The Web as a medium for response time data collection. *Behavior Research Methods*, *48*(3), 1086–1099. doi:10.3758/s13428-015-0632-x

Cisek, P., & Kalaska, J. F. (2005). Neural correlates of reaching decisions in dorsal premotor cortex: Specification of multiple direction choices and final selection of action. *Neuron*, *45*(5), 801–814. doi:10.1016/j.neuron.2005.01.027

Dale, R., & Duran, N. D. (2011). The cognitive dynamics of negated sentence verification. *Cognitive Science*, *35*(5), 983–996. doi:10.1111/j.1551-6709.2010.01164.x

Dale, R., Kehoe, C., & Spivey, M. J. (2007). Graded motor responses in the time course of categorizing atypical exemplars. *Memory & Cognition*, *35*(1), 15–28. doi:10.3758/BF03195938

de Leeuw, J. R., & Motz, B. A. (2015). Psychophysics in a Web browser? Comparing response times collected with JavaScript and Psychophysics Toolbox in a visual search task. *Behavior Research Methods*, *48*(1), 1–12. doi:10.3758/s13428-015-0567-2

Diedenhofen, B., & Musch, J. (2017). PageFocus: Using paradata to detect and prevent cheating on online achievement tests. *Behavior Research Methods*, *49*(4), 1444–1459. doi:10.3758/s13428-016-0800-7

Dotan, D., Pinheiro-Chagas, P., Al Roumi, F., & Dehaene, S. (2019). Track it to crack it: Dissecting processing stages with finger tracking. *Trends in Cognitive Sciences*, *23*(12), 1058–1070. doi:10.1016/j.tics.2019.10.002

Fischer, M. H., & Hartmann, M. (2014). Pushing forward in embodied cognition: May we mouse the mathematical mind? *Frontiers in Psychology*, *5*, 1315. doi:10.3389/fpsyg.2014.01315

Freeman, J. B. (2018). Doing psychological science by hand. *Current Directions in Psychological Science*, *27*(5), 315–323. doi:10.1177/0963721417746793

Freeman, J. B., & Ambady, N. (2010). MouseTracker: Software for studying real-time mental processing using a computer mouse-tracking method. *Behavior Research Methods*, *42*(1), 226–241. doi:10.3758/BRM.42.1.226

Freeman, J. B., Dale, R., & Farmer, T. A. (2011). Hand in motion reveals mind in motion. *Frontiers in Psychology*, *2*, 59. doi:10.3389/fpsyg.2011.00059

Garaizar, P., & Reips, U.-D. (2018). Best practices: Two Web-browser-based methods for stimulus presentation in behavioral experiments with high-resolution timing requirements. *Behavior Research Methods*. doi:10.3758/s13428-018-1126-4

Götz, F. M., Stieger, S., & Reips, U.-D. (2017). Users of the main smartphone operating systems (iOS, Android) differ only little in personality. *PLOS ONE*, *12*(5), e0176921. doi:10.1371/journal.pone.0176921

Grage, T., Schoemann, M., Kieslich, P. J., & Scherbaum, S. (2019). Lost to translation: How design factors of the mouse-tracking procedure impact the inference from action to cognition. *Attention, Perception, & Psychophysics*, *81*(7), 2538–2557. doi:10.3758/s13414-019-01889-z

Heck, D. W., Erdfelder, E., & Kieslich, P. J. (2018). Generalized processing tree models: Jointly modeling discrete and continuous variables. *Psychometrika*, *83*(4), 893–918. doi:10.1007/s11336-018-9622-0

Hehman, E., Stolier, R. M., & Freeman, J. B. (2015). Advanced mouse-tracking analytic techniques for enhancing psychological science. *Group Processes & Intergroup Relations*, *18*(3), 384–401. doi:10.1177/1368430214538325

Henninger, F., Shevchenko, Y., Mertens, U. K., Kieslich, P. J., & Hilbig, B. E. (In press). Lab.js: A free, open, online experiment builder. *Behavior Research Methods*.

Hilbig, B. E. (2016). Reaction time effects in lab- versus Web-based research: Experimental evidence. *Behavior Research Methods*, *48*(4), 1718–1724. doi:10.3758/s13428-015-0678-9

Kieslich, P. J., & Henninger, F. (2017). Mousetrap: An integrated, open-source mouse-tracking package. *Behavior Research Methods*, *49*(5), 1652–1667. doi:10.3758/s13428-017-0900-z

Kieslich, P. J., Henninger, F., Wulff, D. U., Haslbeck, J. M. B., & Schulte-Mecklenbeck, M. (2019). Mouse-tracking: A practical guide to implementation and analysis. In M. Schulte-Mecklenbeck, A. Kühberger, & J. G. Johnson (Eds.), *A Handbook of Process Tracing Methods* (pp. 111–130). New York, NY: Routledge.

Kieslich, P. J., Schoemann, M., Grage, T., Hepp, J., & Scherbaum, S. (2019). Design factors in mouse-tracking: What makes a difference? *Behavior Research Methods*. doi:10.3758/s13428-019-01228-y

Koop, G. J., & Johnson, J. G. (2011). Response dynamics: A new window on the decision process. *Judgment and Decision Making*, *6*(8), 750–758. Retrieved May 25, 2016, from http://search.proquest.com/docview/1011297217/abstract/9E425198DD974A3EPQ/1

Maldonado, M., Dunbar, E., & Chemla, E. (2019). Mouse tracking as a window into decision making. *Behavior Research Methods*, *51*(3), 1085–1101. doi:10.3758/s13428-018-01194-x

Mathôt, S., Schreij, D., & Theeuwes, J. (2012). OpenSesame: An open-source, graphical experiment builder for the social sciences. *Behavior Research Methods*, *44*(2), 314–324. doi:10.3758/s13428-011-0168-7

Mathur, M. B., & Reichling, D. B. (2019). Open-source software for mouse-tracking in Qualtrics to measure category competition. *Behavior Research Methods*, *51*(5), 1987–1997. doi:10.3758/s13428-019-01258-6

Miller, R., Schmidt, K., Kirschbaum, C., & Enge, S. (2018). Comparability, stability, and reliability of internet-based mental chronometry in domestic and laboratory settings. *Behavior Research Methods*, *50*(4), 1345–1358. doi:10.3758/s13428-018-1036-5

Oldfield, R. C. (1971). The assessment and analysis of handedness: The Edinburgh inventory. *Neuropsychologia*, *9*(1), 97–113. doi:10.1016/0028-3932(71)90067-4

Pronk, T., Wiers, R. W., Molenkamp, B., & Murre, J. (2019). Mental chronometry in the pocket? Timing accuracy of web applications on touchscreen and keyboard devices. *Behavior Research Methods*. doi:10.3758/s13428-019-01321-2

R Core Team. (2020). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing. Vienna, Austria. Retrieved from https://www.R-project.org

Reimers, S., & Stewart, N. (2015). Presentation and response timing accuracy in Adobe Flash and HTML5/JavaScript Web experiments. *Behavior Research Methods*, *47*(2), 309–327. doi:10.3758/s13428-014-0471-1

Scherbaum, S., & Dshemuchadse, M. (2019). Psychometrics of the continuous mind: Measuring cognitive sub-processes via mouse tracking. *Memory & Cognition*. doi:10.3758/s13421-019-00981-x

Scherbaum, S., & Kieslich, P. J. (2018). Stuck at the starting line: How the starting procedure influences mouse-tracking data. *Behavior Research Methods*, *50*(5), 2097–2110. doi:10.3758/s13428-017-0977-4

Schulz, E., Speekenbrink, M., & Krause, A. (2018). A tutorial on Gaussian process regression: Modelling, exploring, and exploiting functions. *Journal of Mathematical Psychology*, *85*, 1–16. doi:10.1016/j.jmp.2018.03.001

Semmelmann, K., & Weigelt, S. (2017). Online psychophysics: Reaction time effects in cognitive experiments. *Behavior Research Methods*, *49*(4), 1241–1260. doi:10.3758/s13428-016-0783-4

Song, J.-H., & Nakayama, K. (2009). Hidden cognitive states revealed in choice reaching tasks. *Trends in Cognitive Sciences*, *13*(8), 360–366. doi:10.1016/j.tics.2009.04.009

Spivey, M. J., & Dale, R. (2006). Continuous dynamics in real-time cognition. *Current Directions in Psychological Science*, *15*(5), 207–211. doi:10.1111/j.1467-8721.2006.00437.x

Spivey, M. J., Grosjean, M., & Knoblich, G. (2005). Continuous attraction toward phonological competitors. *Proceedings of the National Academy of Sciences of the United States of America*, *102*(29), 10393–10398. doi:10.1073/pnas.0503903102. pmid: 15985550

Stieger, S., & Reips, U.-D. (2010). What are participants doing while filling in an online questionnaire: A paradata collection tool and an empirical study. *Computers in Human Behavior*. Online Interactivity: Role of Technology in Behavior Change, *26*(6), 1488–1495. doi:10.1016/j.chb.2010.05.013

Stillman, P. E., Shen, X., & Ferguson, M. J. (2018). How mouse-tracking can advance social cognitive theory. *Trends in Cognitive Sciences*, *22*(6), 531–543. doi:10.1016/j.tics.2018.03.012

Travers, E., Rolison, J. J., & Feeney, A. (2016). The time course of conflict on the Cognitive Reflection Test. *Cognition*, *150*, 109–118. doi:10.1016/j.cognition.2016.01.015

Wirth, R., Pfister, R., & Kunde, W. (2016). Asymmetric transfer effects between cognitive and affective task disturbances. *Cognition and Emotion*, *30*(3), 399–416. doi:10.1080/02699931.2015.1009002. pmid: 25758946

Wulff, D. U., Haslbeck, J. M. B., Kieslich, P. J., Henninger, F., & Schulte-Mecklenbeck, M. (2019). Mouse-tracking: Detecting types in movement trajectories. In M. Schulte-Mecklenbeck, A. Kühberger, & J. G. Johnson (Eds.), *A Handbook of Process Tracing Methods* (pp. 131–145). New York, NY: Routledge.

Zgonnikov, A., Aleni, A., Piiroinen, P. T., O'Hora, D., & di Bernardo, M. (2017). Decision landscapes: Visualizing mouse-tracking data. *Royal Society Open Science*, *4*(11), 170482. doi:10.1098/rsos.170482