Mouse-tracking:

A practical guide to implementation and analysis

Pascal J. Kieslich[a], Felix Henninger[a, b], Dirk U. Wulff[c, d],

Jonas M. B. Haslbeck[e] & Michael Schulte-Mecklenbeck[d, f]

University of Mannheim, Germany[a],

University of Koblenz-Landau, Germany[b],

University of Basel, Switzerland[c],

Max Planck Institute for Human Development, Berlin, Germany[d],

University of Amsterdam, Netherlands[e],

University of Bern, Switzerland[f]

This chapter will be published as: Kieslich, P. J., Henninger, F., Wulff, D. U., Haslbeck, J. M. B., & Schulte-Mecklenbeck, M. (in press). 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*. New York, NY: Routledge.

Correspondence should be addressed to: Pascal J. Kieslich, Experimental Psychology Lab, School of Social Sciences, University of Mannheim, L13, 17, 68161 Mannheim, Germany, E-Mail: kieslich@psychologie.uni-mannheim.de.

This work was supported by the University of Mannheim's Graduate School of Economic and Social Sciences funded by the German Research Foundation.

Mouse-tracking: A practical guide to implementation and analysis

The motivation behind process tracing is to go beyond the mere observation of a choice as the behavioral outcome and more directly observe the psychological process by collecting additional variables. A central unobserved quantity in choice tasks is the degree to which each alternative received consideration during the choice process, and how commitment to and conflict between options developed over time. Mouse-tracking is based on the assumption that motor movements in a given time interval contain a signal of the cognitive processes during that period (Spivey & Dale, 2006). Specifically, it is assumed that the direction of movement toward or away from alternatives reflects their relative attraction at a given time point during the decision process. To gain access to this information, mouse-tracking records hand movements indirectly by sampling the cursor position of a computer mouse with a high frequency while participants decide between (and move toward) options presented at different locations on the computer screen. Mouse-tracking is an increasingly popular process tracing technique that has been applied to a wide range of questions throughout many fields of psychology (see Chapters 9-10; see also Freeman, Dale, & Farmer, 2011; Stillman, Shen, & Ferguson, 2018).

This chapter provides an introduction to the collection, analysis and visualization of mouse-tracking data using free, open-source software. We show how to create mouse-tracking experiments using the graphical experiment builder OpenSesame (Mathôt, Schreij, & Theeuwes, 2012) in combination with the mousetrap plugin (Kieslich & Henninger, 2017). Analysis and visualization rely on the mousetrap package (Kieslich, Wulff, Henninger, Haslbeck, & Schulte-Mecklenbeck, 2018) for the statistical programming language R (R Core Team, 2016).[1]

To illustrate the method and its implementation in mousetrap, we replicate a mouse-tracking experiment by Dale, Kehoe, and Spivey (2007). In this study, participants classified exemplars (animals) into one of two categories (e.g., mammal or bird) by clicking on the corresponding buttons located at the top-left and top-right of the screen. The independent variable was the typicality of each exemplar for its category. The experiment included typical exemplars (e.g., dog for mammal) as well as atypical ones that shared features both with the correct and the competing

[1] Note that other options for creating mouse-tracking experiments and analyzing mouse-tracking data are available (e.g., MouseTracker, cf., Freeman & Ambady, 2010) and a discussion of the different software packages is provided elsewhere (Kieslich & Henninger, 2017; Kieslich et al., 2018).

category (e.g., a bat, sharing both features with the correct category mammal and the incorrect category bird). Dale et al. (2007) hypothesized that for atypical exemplars, both response options would receive some degree of activation, whereas for the typical exemplars, activation would largely be limited to the correct category. Consequently, for atypical exemplars, the incorrect category should exert a stronger attraction, and mouse movements should deviate more in its direction even if participants finally choose the correct option.[2]

Creating mouse-tracking experiments

In this section we demonstrate how a mouse-tracking experiment can be created in OpenSesame (Mathôt et al., 2012). OpenSesame is a free, open-source software for creating experiments via a graphical user interface which additionally allows for full customization of studies using Python code.[3] To simplify the creation of mouse-tracking experiments inside this framework, we developed the mousetrap plugin (Kieslich & Henninger, 2017) for OpenSesame. Installation instructions and additional documentation for the plugin are available in its GitHub repository at https://github.com/pascalkieslich/mousetrap-os.

Creating an experiment

The first step is to start OpenSesame and create a new experiment by clicking on File/New and selecting the default template. Experiments in OpenSesame are assembled from a set of items, for example, a *sketchpad* item for presenting graphical content on the screen, a *keyboard_response* item for collecting key presses, and a *logger* item for writing data into log files. Figure 1 shows the OpenSesame interface with the item toolbar on the left-hand side. To its right, the overview area represents the study's structure, in that the items therein are run sequentially from top to bottom. An experiment is built by dragging and dropping items from the toolbar into the overview area. *Sequences* can be used to run a number of items in succession. In addition, *loop* items can be used to repeatedly run sequences with some degree of variation, for example, trials with varying stimuli (Figure 1, right panel).

[2] The data for this replication were collected by Kieslich and Henninger (2017); the corresponding material, data, analyses, and results are available at https://github.com/pascalkieslich/mousetrap-resources.

[3] OpenSesame can be obtained free of charge from http://osdoc.cogsci.nl/, where a general introduction to the program and detailed documentation are also available.

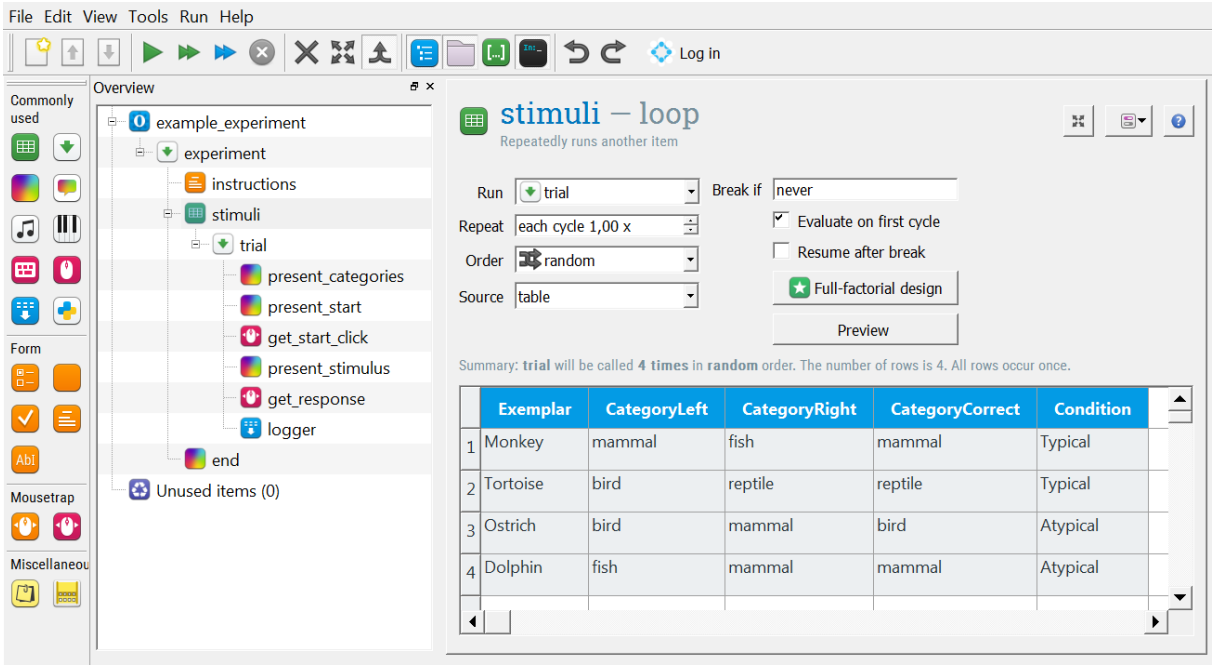

Figure 1. User interface of OpenSesame, showing the final state of the tutorial experiment. In the left-most panel, the item toolbar contains the available items, including the mousetrap plugin items visible toward the bottom. The overview area represents the study's structure. The right panel shows the user interface of the stimulus loop containing four exemplary stimuli.

Setting up the screen.

Mouse-tracking experiments are typically run in fullscreen mode. Therefore, before adding content to a new experiment, the screen resolution should be adjusted to match that of the computers used for data collection. This is done in the overall experiment settings, which are accessible by clicking on the topmost item in the study overview area ("example_experiment" in Figure 1).

Creating the study structure.

The first item in the experiment provides the instructions. For this, we use a *form_text_display* item that presents text and a button to continue the study. It can be added to the study by dragging it from the item toolbar into the overview area (cf., Figure 1).

In the central part of our study, participants will make categorization decisions for different animal exemplars and pairs of response categories. To accommodate this recurring structure, we include a loop item that varies the information presented on each iteration. In the loop options, the stimulus material is represented as a table where rows reflect the different stimuli and columns

contain the variables that differ for each stimulus (Figure 1, right panel). In our case, the vital pieces of information are the name of the exemplar and the response categories, which are contained in the columns *Exemplar*, *CategoryLeft* and *CategoryRight*. The additional columns specify the correct response and typicality of each combination; though not presented to participants, they are stored in the dataset and facilitate later analysis. Using the default settings shown in Figure 1, the order of stimuli is randomized, and each stimulus is presented once.

Nested inside the loop, a *sequence* item is used to build each trial. It combines several screen pages as well as the collection of responses and logging of the stimulus and response information.

Building a mouse-tracking screen.

The central part of a mouse-tracking experiment is the stimulus display that presents the name of the exemplar and the two response buttons (located in the upper screen corners). We create this display by placing a *sketchpad* item into the trial sequence. In our example, it is named "present_stimulus" (Figure 2).[4] The content of the sketchpad item is added using the visual editor. The available types of elements for creating content are shown in the toolbar to the left of the preview. After selecting an element type, the contents can be drawn inside the preview (to move or edit them afterwards they can be selected using the topmost option in the toolbar). In our example, rectangles (*rect elements*) of equal size represent the response buttons, placed in the top left and right screen corners. Button labels are added in the center of each button using *textline elements*. An additional *textline element* is used to present the name of the to-be-categorized exemplar in the lower part of the screen. By default, the inserted text is presented verbatim. However, one can easily vary content across trials by replacing static text with the appropriate variable names in square brackets (i.e., "[CategoryLeft]" and "[CategoryRight]" for the button labels and "[Exemplar]" for the exemplar name). In every iteration of the loop, OpenSesame will replace the variable name with the variable's current value. To make sure that the button borders are identifiable in the subsequent *mousetrap_response* item (cf., next section), we must furthermore label the two *rect elements* using the *Name* field (cf., Figure 2 top row). Each button border is labeled using the corresponding variable name ("[CategoryLeft]" and "[CategoryRight]").

[4] The additional screens that are presented beforehand ("present_categories" and "present_start") will be described in the section *Design considerations*.

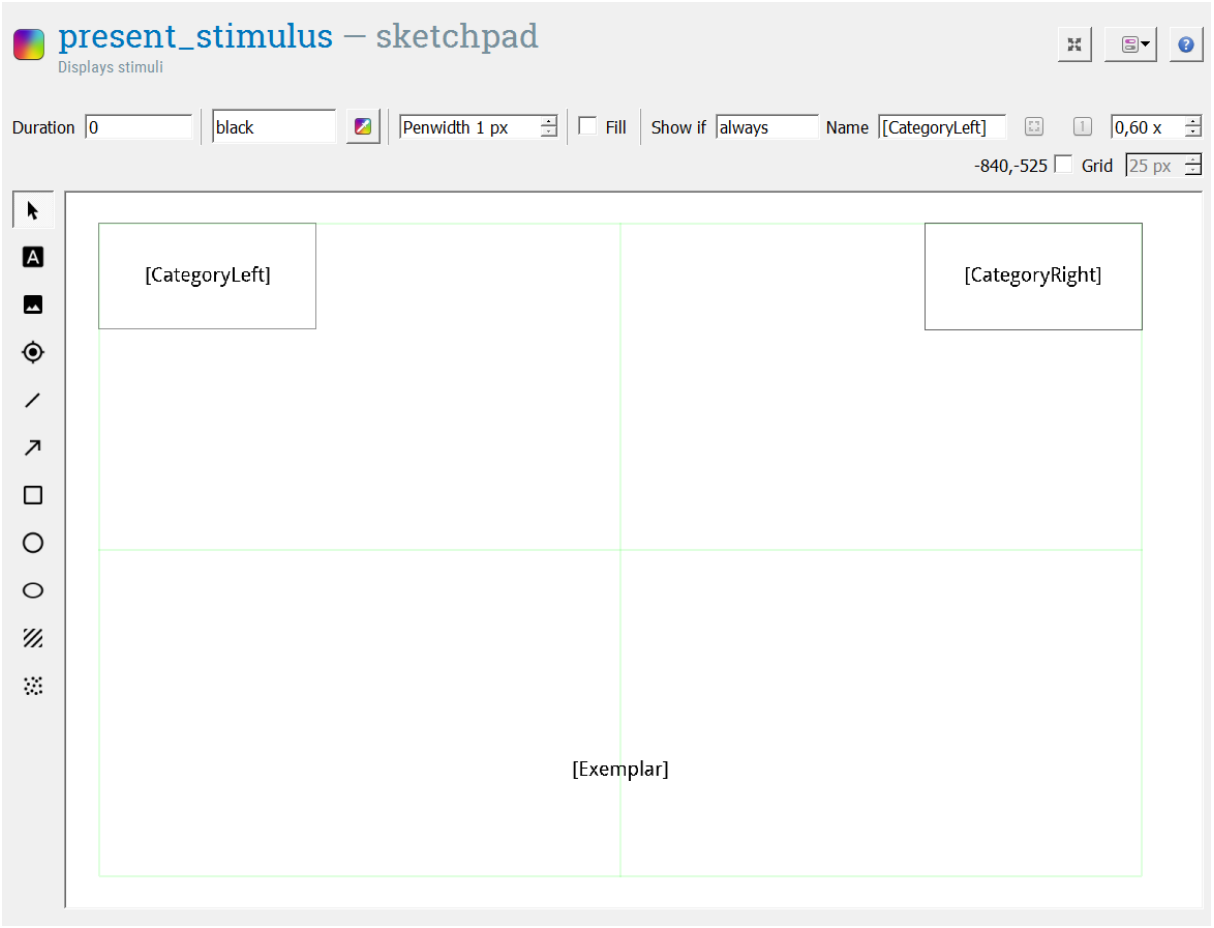

Figure 2. Sketchpad item used to create the main stimulus display. The exemplar is displayed using a *textline* element that contains the name of the corresponding variable from the stimulus loop (cf., Figure 1). The two button borders are created using *rect elements*. Each button border is labeled using the *Name* field (see top row) and as label the corresponding values from the stimuli loop are used. The button labels are displayed using *textline elements* that are placed in the center of each button.

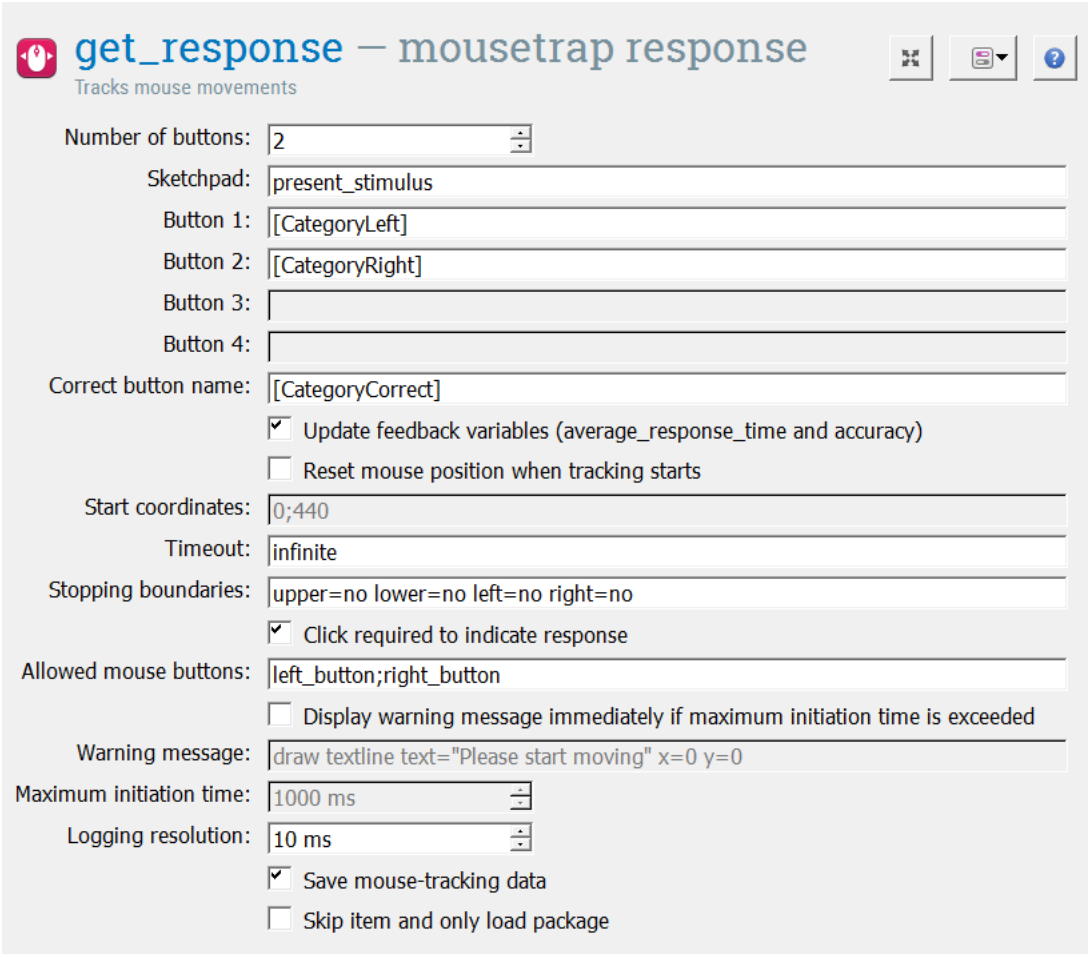

Figure 3. Settings of the mousetrap_response item.

Tracking mouse movements.

After creating the stimulus presentation, we specify the collection of mouse-tracking data and button clicks using the *mousetrap_response* item, which is inserted directly after the sketchpad item and called "get_response".[5] To start recording cursor positions immediately following stimulus presentation, the duration of the sketchpad is set to 0.

The *mousetrap_response* item records the cursor position at a constant sampling rate (10 ms by default) until the participant clicks on one of the buttons. To register responses, the corresponding buttons need to be defined (Figure 3, upper part): first, the number of buttons is specified. Then, the name of the sketchpad that presents the buttons is entered ("present_stimulus").

[5] The mousetrap plugin includes two items for tracking mouse movements. As an alternative to the *mousetrap_response* item, a *mousetrap_form* item combines stimulus presentation and response collection; its contents are defined using a basic syntax instead of a visual editor. More information is provided in Kieslich and Henninger (2017).

Finally, the buttons are specified via the labels of the button borders used on the sketchpad ("[CategoryLeft]" and "[CategoryRight]"). As a result, if the participant selects the left button, the value of the variable *CategoryLeft* is recorded as their response.

The *mousetrap_response* item also provides additional options (cf., lower part of Figure 3). For example, if the name of the correct button is specified, OpenSesame will automatically create a variable *correct* that is set to 1 or 0 for correct and incorrect answers, respectively (this is useful for analysis, as well as for providing feedback during the study). Additional design options are discussed in the section *Design considerations*.

Storing data.

As the final part of the trial sequence, a logger item writes the data from the current trial into a log file. This includes variables pertaining to the study as a whole (e.g., the *subject_nr*), the current values of all variables in the stimuli loop (cf., Figure 1) and the response variables. OpenSesame stores participants' responses in two places – global variables (*response*, *response_time* etc.) that always store the last recorded response and response time in the experiment, and item-specific variables named after the item that collected the response (e.g., *response_get_response* in the current example). The recorded mouse positions and associated timestamps are stored in item-specific variables only, in order to save memory (*xpos_get_response, ypos_get_response* and *timestamps_get_response*).

Design considerations

When setting up mouse-tracking experiments, researchers are faced with a number of design choices. These include decisions about the starting procedure, the cursor speed and acceleration settings, and the response mode (click or mouse-over). Each of these choices aims to ensure that all cognitive processes relevant to the decision take place while the tracking is active (which is, in many cases, the period between the click on a start button and the selection of one of the response options), so that the process of interest is captured in the trajectories. In the remainder of this section, we discuss available options for a number of important design choices, and their potential impact on the recorded mouse trajectories (see also Fischer & Hartmann, 2014; Hehman, Stolier, & Freeman, 2015; Kieslich & Henninger, 2017; Scherbaum & Kieslich, 2018, for additional discussions about design choices).

Start button.

Virtually all mouse-tracking experiments try to enforce a comparable start position of the cursor across trials, thereby ensuring that the cursor is centered horizontally and approximately equidistant to all response options at the beginning of each trial. To achieve this, another screen with a start button can be added prior to the display of the task stimulus. The button ensures that participants have to return to a common area before subsequently initiating mouse movements for a new choice. In the current experiment, this is implemented using a *sketchpad* called "present_start" combined with a *mousetrap_response* item called "get_start_click" (cf., Figure 1). As before, the screen content is assembled in the visual editor and a start button is placed in the lower center of the screen (and labeled "Start"). The name of the start button is entered in the options of the *mousetrap_response* item as the single possible response. As mouse-tracking data prior to the stimulus presentation are not of interest, the option *save mouse-tracking data* can be unchecked for the "get_start_click" item. While the start button ensures that the cursor position at tracking onset is comparable across trials, it does not guarantee that it is identical. If this is desired, one can select "Reset mouse position when tracking starts" and specify coordinates in the "get_response" item (cf., Figure 3).

Information presentation.

Another key challenge in designing mouse-tracking studies is the temporal order in which task-relevant information is presented to the participant. On the one hand, the amount of information presented after the onset of tracking should be minimized to ensure that the collected mouse-tracking data reflects the decision processes. On the other hand, the decision-critical information needs to be withheld until tracking begins, to prevent participants from making their decision beforehand. In the current example, these considerations are accommodated by presenting the information about the two response categories for 2000 ms prior to tracking onset, but presenting the to-be-categorized exemplar only after the click on the start button (following the original procedure of Dale et al., 2007). We implemented this procedure by including another *sketchpad* item called "present_categories" at the beginning of the trial that presents only the two response categories, before the start button is made available to participants (cf., Figure 1).

Counterbalancing.

Another design factor concerns the assignment of response options to the button positions on the screen. Specifically, in the current study we would like to ensure that the correct answer is not always presented on the same side. One solution for this is counterbalancing the position of the correct answer between stimuli, while keeping their position fixed for all participants (cf., Figure 1). Ideally, however, the position of both response options is drawn anew for each participant and stimulus (this can be achieved in OpenSesame through the advanced randomization operation shuffle horizontal).

Starting procedure.

For mouse-tracking to reflect the cognitive processes underlying the choice, movement must occur while the cognitive process is ongoing. It has been shown that the starting procedure has a considerable influence on the obtained trajectories (Scherbaum & Kieslich, 2018).

Many mouse-tracking studies have used a so-called *static starting procedure*, in which the stimulus is shown immediately after participants have clicked on the start button and without any specific measures taken to ensure movement during processing (our tutorial experiment following Dale et al., 2007, is an example for such a setup). While many mouse-tracking studies that use a static starting procedure find theoretically relevant effects in mouse trajectories, this procedure does not exclude the possibility that (in some trials) decision-relevant processes take place before the mouse movement is initiated and therefore are not captured by mouse trajectories.

To ensure that the cognitive processes under investigation do not take place before mouse movement initialization, some studies have modified the starting procedure. One option is the *static starting procedure with delay*, in which a brief lag of, for example, 500 ms, is inserted between clicking the start button and stimulus presentation. Previous studies reported that this often successfully led participants to initialize movement before the stimulus appeared (e.g., Spivey, Grosjean, & Knoblich, 2005). Other studies employ a static starting procedure with immediate stimulus presentation, but explicitly instruct participants to initiate their mouse movement within a certain time limit and display a warning to participants after the trial if the *initiation time* exceeds the threshold. The exact time limit depends on the task (a typical value is 400 ms; see Hehman et al., 2015, p. 388–389, for a discussion).

A more rigorous option, however, is to implement a *dynamic starting* procedure that presents the stimulus only after participants have moved the mouse upwards for a minimum distance (e.g., Scherbaum, Dshemuchadse, Fischer, & Goschke, 2010). The dynamic procedure forces participants to initiate their movement in order to receive the critical information needed to make the choice. It can be implemented by placing an invisible horizontal boundary slightly above the start button that triggers the presentation of the stimulus once it is crossed (cf., Frisch, Dshemuchadse, Görner, Goschke, & Scherbaum, 2015). This procedure has been shown to lead to more consistent movements and larger effects in within-trial temporal analyses (Scherbaum & Kieslich, 2018).[6]

Mouse sensitivity.

Another design choice is the computer's mouse sensitivity, in particular the cursor speed and acceleration. One option is to leave these settings to the operating system defaults (under Windows 7 and 10, medium speed with acceleration). However, it is often preferable to reduce mouse speed and switch off mouse acceleration (Fischer & Hartmann, 2014). This is particularly relevant when using a dynamic starting procedure to ensure that participants can read the dynamically presented stimulus information while continuously moving upwards. The mouse sensitivity settings cannot be adjusted directly within OpenSesame, but need to be set in the computer's system preferences.

Response mode.

The two main response modes in mouse-tracking studies are clicking on and moving over the response buttons. In the mousetrap plugin, users can switch between the two response modes by checking or unchecking the option *Click required to indicate response*, which is enabled by default (cf., Figure 3).

Data collection and testing.

After creating the experiment, it can be run from within OpenSesame for testing or using *OpenSesame Run* for data collection in the laboratory (see Kieslich & Henninger, 2017, for more information on running mouse-tracking experiments). Mouse-tracking studies also usually assess the handedness of participants and the hand participants use for moving the mouse (with some authors recommending only to include right-handed participants, cf., Hehman et al., 2015).

[6] An example experiment implementing this procedure can be found at https://github.com/pascalkieslich/mousetrap-os#examples.

Analyzing mouse-tracking data

We will now demonstrate the typical steps of a basic mouse-tracking analysis using the data from the replication experiment described above (Kieslich & Henninger, 2017). For this analysis, we will use the *mousetrap* package (Kieslich et al., 2018) in the statistical programming language R (R Core Team, 2016), which facilitates preprocessing, analysis and visualization of mouse-tracking data.[7] Once installed, mousetrap functions can then be made available within an R session by loading the package via:

```
library(mousetrap)
```

A detailed overview of its functionality is provided online at http://pascalkieslich.github.io/mousetrap/ or within R using the command:

```
package?mousetrap
```

In the following, we discuss the most important analysis steps, starting with data import and preprocessing operations, followed by the computation and analysis of common indices, temporal analyses, and visualizations.

Import

First, the raw data need to be read into R's workspace. OpenSesame stores the data for each participant in a separate csv file. To load all csv files from a directory and combine them into a single dataset, we use the *read_opensesame* function from the *readbulk* package (Kieslich & Henninger, 2016). The following command assumes that all data files can be found in the folder "raw_data" in the working directory and stores the imported data in the dataset "KH2017_raw" (this dataset is available once the mousetrap package has been loaded, so no raw data have to be imported to follow this tutorial):

```
library(readbulk)
KH2017_raw <- read_opensesame("raw_data")
```

[7] R is open-source and freely available from https://www.r-project.org/. We recommend using R in combination with RStudio (available from https://www.rstudio.com/), which greatly facilitates code development and analysis by providing many useful features such as code highlighting, debugging, and tools for data inspection.

Next, the data need to be transformed into a *mousetrap data object* to perform analyses using the mousetrap R package.[8] This results in a mousetrap data object (called "mt_data" in the current analysis), which is described in detail in Information box 1:

```
mt_data <- mt_import_mousetrap(KH2017_raw)
```

Using this two-step procedure of reading and importing the mouse-tracking data, the mousetrap R package can also be used for data collected in other software. An example for reading and importing raw data collected with MouseTracker (Freeman & Ambady, 2010) is given in the documentation of the *read_mt* function, which can be accessed by entering:

```
?read_mt
```

Preprocessing

Spatial transformations.

In a typical two-alternative choice design (as implemented in the example experiment, see Figure 2), trajectories end either at the left or the right response option. As the overall spatial direction is irrelevant for most analyses (as opposed to the substantive meaning of the response button, which varies across trials if the position of alternatives is counterbalanced), all trajectories are remapped so that they end on the same side. By default, mousetrap maps the trajectories to the left, implying that trajectories that end on the right-hand side are flipped from right to left:

```
mt_data <- mt_remap_symmetric(mt_data)
```

Similarly, differences in the trajectories' starting points are often not of substantive interest. If the cursor's starting position was not reset to exact coordinates during the experiment (as is the case for the example data set), it can be aligned by shifting the trajectories in preprocessing:

```
mt_data <- mt_align_start(mt_data, start=c(0,0))
```

[8] In case that only one mousetrap item in the experiment collected mouse-tracking data, the *mt_import_mousetrap* function automatically detects the mouse-tracking variables in the raw data. If more than one item stored mouse-tracking data, the variable names have to be set explicitly once using the *xpos_label*, *ypos_label*, and *timestamps_label* arguments when importing data via the *mt_import_mousetrap* function.

Information box 1. Working with mousetrap data objects

The mousetrap R package represents mouse-tracking data in a specialized data structure, a mousetrap data object. This allows the package to store and process mouse trajectories efficiently, and to link them to other information collected during the study. All mousetrap analysis functions use mousetrap data objects as input; therefore, the collected data must be imported before processing and analysis. A newly imported mousetrap data object consists of a *data.frame* called *data* containing the trial information (without mouse trajectories) and an *array* called *trajectories* containing the recorded mouse-tracking data.

The mousetrap data object can hold multiple sets of trajectories (e.g., *mt_time_normalize* adds the time-normalized trajectories as *tn_trajectories*). In subsequent analyses, the user can specify via the *use* argument whether an analysis (or visualization) should be performed based on the raw trajectories (`use="trajectories"`, which is used by default in most functions) or another trajectory array (e.g., `use="tn_trajectories"`). Other functions add new *data.frames* to the mousetrap object (e.g., *mt_measures* adds a *data.frame* called *measures* that contains trial-level indices).

The mousetrap package is designed for processing and visualizing trajectories and the computation of indices. For statistical analyses of the computed indices, they can be merged with the other trial data via:

```
results <- merge(mt_data$data, mt_data$measures, by="mt_id")
```

Similarly, mouse trajectories can be transformed into a format required for the statistical analysis using the *mt_export_long* or *mt_export_wide* functions. The resulting data can then be analyzed outside of the mousetrap package using any standard analysis method.

Resampling.

The cursor position is typically recorded at a constant sampling rate. The mousetrap plugin in OpenSesame records the mouse position every 10 ms by default (corresponding to a sampling rate of 100 Hz). Due to variation in trial durations, the number of recorded cursor positions may vary considerably across trials. To be able to aggregate trajectories or compare them statistically, one often requires an equal number of coordinates for all trajectories. To achieve this, studies commonly apply time-normalization:

```
mt_data <- mt_time_normalize(mt_data)
```

Time-normalization interpolates trajectories so that each is represented by the same number of positions (101 by default, following Spivey et al., 2005) separated by a (within-trial) constant time interval. Mousetrap stores the time-normalized data as a new set of trajectories within the mousetrap data object (see Information box 1).

Another possibility is to interpolate trajectories so that each is represented by the same number of spatially equidistant positions (using *mt_spatialize*). This processing step facilitates the comparison of trajectory shapes and is instrumental in type-based analyses of trajectories (cf., Chapter 9).

Data inspection and filtering.

As a final step prior to analysis, trials are typically screened and filtered based on one or more criteria. If choices can be graded as correct, studies often exclude trials with incorrect responses to ensure a consistent interpretation of curvature across all trials (i.e., that increased curvature always reflects attraction towards the distractor category). The *mt_subset* function can be used to select only correctly answered trials for further analysis (or to apply other filters):

```
mt_data <- mt_subset(mt_data, correct==1)
```

An additional concern in mouse-tracking analysis is whether the data contain movements that are presumably not related to the preference development but to other processes, such as information acquisition or slips of the hand. Information acquisition might, for example, be reflected by directed movements towards a point where information was presented on the screen. Slips of the hand, resulting, for example, from participants placing the mouse device somewhere else in order to avoid a physical obstacle (or in order to more comfortably move it), would lead to erratic movements and result in movements untypical for this context, for example, comparatively large amounts of up and down movements. The challenge is finding precise criteria to differentiate

between relevant and irrelevant movements. One possibility is an exploratory approach, for example, visually inspecting all trials by plotting them either in a single figure (using *mt_plot* or *mt_heatmap*, see also top panel of Figure 5 in the section *Trajectory types*) or separately (using *mt_plot_per_trajectory*). If to-be-excluded movement patterns have been specified, separate plots per trajectory might also be provided to human raters who can code whether these are present in a trial. Another possibility is to exclude trials based on a numeric criterion, such as trials exceeding an absolute or relative reaction time or trials containing several flips along the y-axis (which probably indicate large amounts of task-irrelevant tracking data). A more detailed discussion is provided in Kieslich et al. (2018). Especially if exclusion criteria were not defined a priori, the impact of the criterion applied should be reported; additional pre-registered studies might be conducted to validate the chosen criteria and to replicate the results under strictly confirmatory conditions.

Analysis

To analyze effects of the experimental manipulation, a common first step is the visual inspection of aggregate time-normalized mouse trajectories. Mousetrap provides the *mt_plot_aggregate* function, which, if used as below, aggregates the time-normalized trajectories for each condition first within and then across participants and plots the result:

```
mt_plot_aggregate(mt_data, use="tn_trajectories",
  x="xpos", y="ypos", color="Condition",
  subject_id="subject_nr")
```

As can be seen in Figure 4, the aggregate mouse trajectory in the current study is more curved towards the non-chosen option for atypical than for typical exemplars – consistent with the hypothesis by Dale et al. (2007). Whether the aggregate trajectories are an adequate summary of the trial-level trajectories is discussed in the section *Trajectory types*.

A wide range of analysis methods are available for mouse-tracking data (for overviews, see Hehman et al., 2015; Kieslich et al., 2018). They can roughly be categorized into analyses that focus on the temporal development of a certain characteristic over the course of a trial (such as x-position, velocity or movement direction, see section *Temporal analyses*) and those that summarize a particular characteristic of each trajectory by computing one index value per trial. Many common indices can be computed using the *mt_measures* function:

```
mt_data <- mt_measures(mt_data)
```

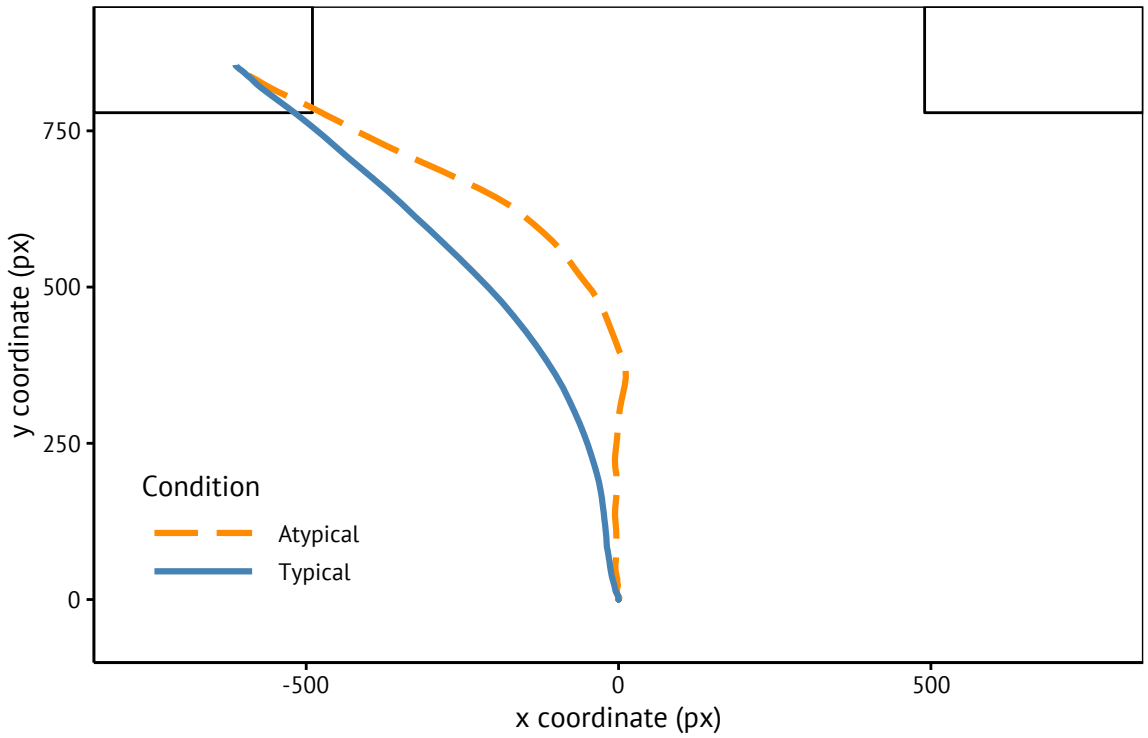

Figure 4. Aggregate time-normalized trajectories per typicality condition. Trajectories were first aligned to a common start position, remapped to the left, and finally aggregated first within and then across participants. Boxes representing the response buttons were added for clarity.

An overview of the different indices is given in Table 1 and further information about working with the computed indices is provided in Information box 1. Different types of indices and their interpretation will be discussed in the following.

Curvature.

The curvature of the response trajectory is used to assess the degree of its attraction towards the non-chosen option. It is assumed to be driven by the difference in activation between the non-chosen and the chosen option – in that a smaller difference in activations leads to a stronger curvature (Spivey, Dale, Knoblich, & Grosjean, 2010). A number of different indices have been suggested to quantify curvature (cf., Table 1). Their exact computation differs, but they are often highly correlated in practice (see Kieslich et al., 2018; Stillman et al., 2018).

Table 1. Selected mouse-tracking measures, their variable name (in brackets, as used by the *mt_measures* function of the mousetrap R package) and definition.

Type	**Measure**	**Definition**
Curvature	Maximum absolute deviation (MAD)	Signed maximum absolute deviation of observed trajectory from direct path
	Maximum deviation above (MD_above)	Maximum deviation above direct path
	Average deviation (AD)	Average deviation of observed trajectory from direct path
	Area under curve (AUC)	Geometric area between observed trajectory and direct path
Complexity	x-flips (xpos_flips)	Number of directional changes along x-axis
	x-reversals (xpos_reversals)	Number of crossings of y-axis
	Sample entropy (sample_entropy)	Degree of unpredictability of movement along x-axis
Time	Response time (RT)	Time until response is given
	Initiation time (initiation_time)	Time until first movement is initiated
	Idle time (idle_time)	Total time without movement across trial
Derivatives	Total distance (total_dist)	Euclidean distance traveled by trajectory
	Max velocity (vel_max)	Maximum movement velocity
	Max acceleration (acc_max)	Maximum movement acceleration

Note: The direct path refers to the straight line connecting the start and end point of the observed trajectory. Deviations/areas above the direct path receive a positive sign and deviations/areas below receive a negative sign. For all derivative measures, it is assumed that movements across both x and y dimensions are taken into account (derivatives have to be calculated using *mt_calculate_derivatives* before calling *mt_measures*). Sample entropy is computed using *mt_sample_entropy* (based on the time-normalized x-positions by default).

In the following, we focus on a frequently used index known as the *(signed) maximum absolute deviation* (MAD). To compute the MAD, imagine an idealized, direct line between the start and end point of the trajectory, and that lines perpendicular to this idealized line are drawn to connect it with every point on the original trajectory. The value of the MAD is defined as the length of the longest of these lines. The sign of the MAD is positive if the deviation is largest above the direct path (in the direction of the non-chosen alternative) and negative if the point of strongest deviation occurs below.

To assess whether the MAD differs between experimental conditions, mouse-tracking studies often aggregate the MAD values across trials per participant and condition, and then compare the aggregate MAD values between conditions using a paired *t*-test.[9] These operations can be performed using *mt_aggregate_per_subject* and R's standard *t.test* function:

```
agg_mad <- mt_aggregate_per_subject(mt_data,
  use_variables="MAD", use2_variables="Condition",
  subject_id="subject_nr")

t.test(MAD~Condition, data=agg_mad, paired=TRUE)
```

In line with the hypothesis by Dale et al. (2007), the MAD values indicate larger curvature in the atypical (M = 343.8 px, SD = 218.6 px) than in the typical condition (M = 172.2 px, SD = 110.8 px), $t(59) = 6.73$, $p < .001$. A replication of the original analyses by Dale et al. using the current dataset can be found online at https://github.com/pascalkieslich/mousetrap-resources.

Trajectory types.

While aggregate response trajectories (cf., Figure 4) and curvature indices provide a first indication of the average curvature of the trajectories in each condition, they do not necessarily represent the shape of the individual trajectories well. Specifically, an aggregate curved trajectory might result from different types of trajectories, for example, a mixture of straight lines and triangular "change of mind" trajectories which first head directly to the non-chosen and then to the chosen option (cf., Chapter 9). If this is the case, the average trajectory might not be representative

[9] Analyses can also be performed on the trial level using mixed-effects models that can account for individual differences between participants as well as trial-level predictors (see https://github.com/pascalkieslich/mousetrap-resources).

of the movement patterns observed in the study, but purely an artefact of aggregation. Under these circumstances, the shape of the aggregate trajectory would provide only limited (and potentially misleading) information about the underlying cognitive processes.

Several methods have been suggested to assess the degree of heterogeneity of the individual trajectories on the trial level. Previous approaches have focused on the distribution of trial-level curvature indices (such as area under curve or MAD, cf., Table 1) and tested them for indications of bimodality. The assumption behind these approaches is that gradually curved trajectories on the trial level should result in a unimodal distribution, while a combination of straight and extremely curved trajectories should result in a bimodal distribution (Hehman et al., 2015). The bimodality of the distribution is frequently assessed by computing the bimodality coefficient (BC; Pfister, Schwarz, Janczyk, Dale, & Freeman, 2013) which is interpreted as bimodal for values > .555 (Freeman & Ambady, 2010). Alternative methods for identifying bimodality have been discussed, especially the Hartigan's dip statistic (Freeman & Dale, 2013). Both methods are implemented in the *mt_check_bimodality* function.

Instead of attempting to detect mixtures of distinct trajectory types based on the distribution of curvature indices (which condense each trajectory to a single numeric value), more recent analysis methods take into account the complete shape of each trajectory by using every point of the trajectory. The shape of individual trajectories can be assessed visually by plotting raw or smoothed heatmaps with the *mt_heatmap* function and by comparing heatmaps between conditions using the *mt_diffmap* function (code examples are provided at https://github.com/pascalkieslich/mousetrap-resources).

As can be seen in Figure 5 (middle panel), there appear to be different types of trajectories on the trial level in the current study, with a large proportion of straight and mildly curved trajectories and a small proportion of extremely curved, "change of mind" trajectories. More importantly, a difference heatmap reveals that the relative occurrence of these types differs between conditions, with a higher proportion of extremely curved trajectories in the atypical condition (orange areas in Figure 5, bottom panel). Analyses that go beyond a visual inspection to identify trajectory types and instead use a clustering approach based on spatial similarity (or the assignment of trajectories to different prototypes) are also implemented in the mousetrap R package and described in Chapter 9 (see also Wulff, Haslbeck, & Schulte-Mecklenbeck, 2018).

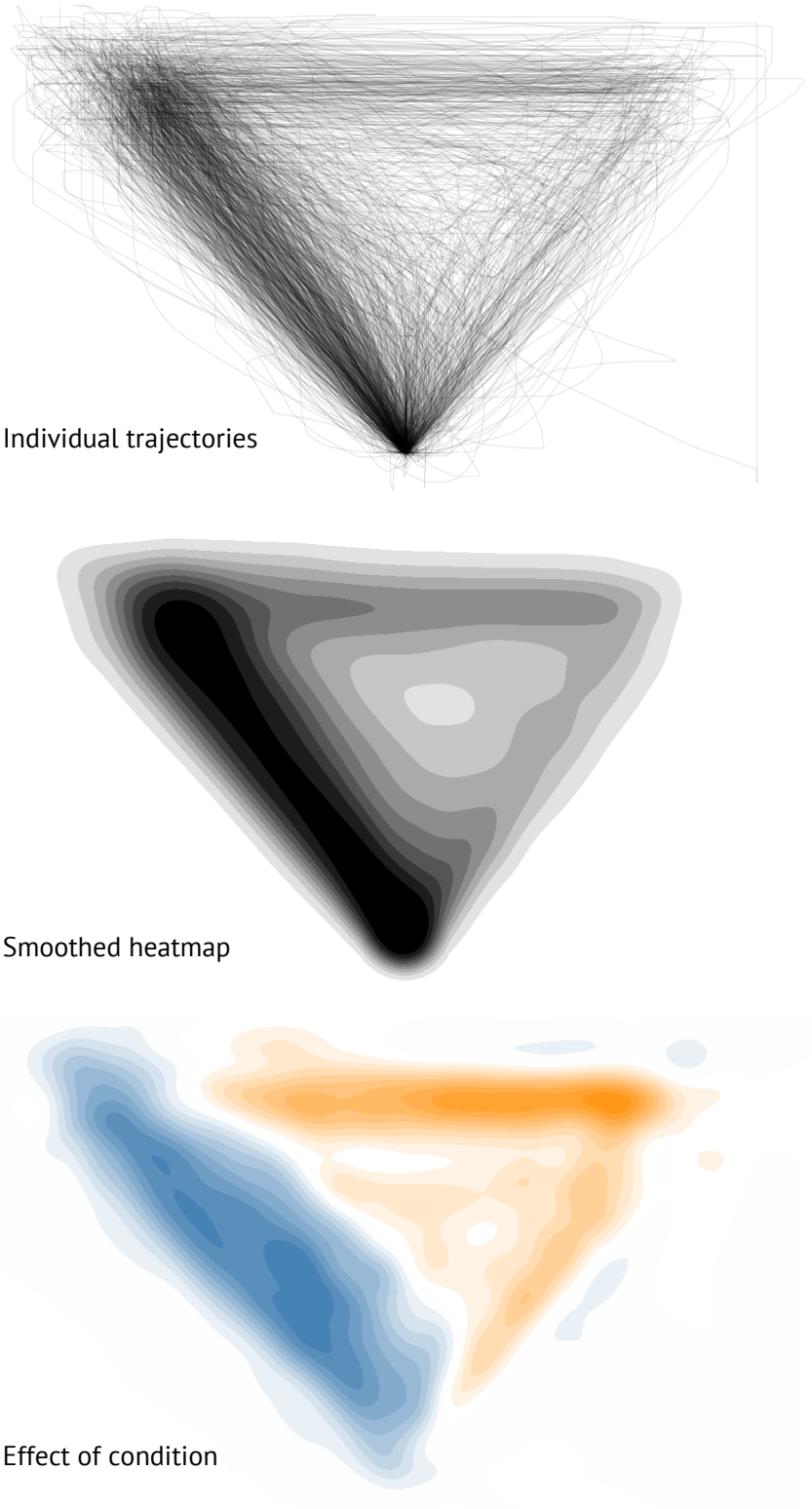

Figure 5. Heatmap of the (remapped) individual trajectories (top panel), smoothed heatmap (middle panel) and difference of smoothed heatmaps between conditions (bottom panel), where blue indicates higher density in the typical and orange higher density in the atypical condition (white indicates comparable density).

The different trajectory types and their frequency have also been used to distinguish between different theoretical models (see Chapters 9-10 for more information). When used to this end, it is important to keep in mind that the setup of mouse-tracking studies can influence the shape of individual trajectories (see section *Design considerations*). In the current study, the occurrence of rather "extreme" trajectory types (straight and "change of mind") may have been caused by the relatively simplistic setup of the study with a static starting procedure, default mouse sensitivity settings and the use of a click instead of a mouse-over response.

Complexity.

In addition to curvature, mouse-tracking studies have also used the complexity of the movement as an indicator of response competition. If multiple response options simultaneously attract the cursor, this should result in more complex, less smooth trajectories compared to cases where only one option exerts an influence (Dale et al., 2007).

In two-alternative tasks, complexity is typically assessed with regard to movements along the horizontal axis, since this is the dimension that separates the options. The most common measure of complexity is x-flips, the number of directional changes along the x-axis (Freeman & Ambady, 2010), which is calculated by the *mt_measures* function (Table 1). As response competition might not always lead to directional changes, other mouse-tracking studies have used sample entropy (Dale et al., 2007; McKinstry, Dale, & Spivey, 2008) which quantifies the degree of unpredictability of movement along the x-axis. Sample entropy can be computed using *mt_sample_entropy*, which uses time-normalized trajectories by default, following the recommendation that each trajectory be represented by the same number of positions (Hehman et al., 2015):

```
mt_data <- mt_sample_entropy(mt_data, use="tn_trajectories")
```

Koop and Johnson (2013) propose a substantive interpretation of complexity-related measures in preferential choice tasks, based on the assumption that the x-position at a specific point during the trial is a proxy for the current absolute preference. They suggest that x-flips reflect changes in the momentary valence whereas x-reversals (i.e., the number of times the cursor crosses the vertical axis at the midpoint between the two options) indicate changes of absolute preference. Recently, the sequence in which certain areas of interest (one for each choice option) are visited with the mouse cursor has also been used to identify how often participants changed their mind during the decision-making process (Szaszi, Palfi, Szollosi, Kieslich, & Aczel, 2018; see also Travers, Rolison, & Feeney, 2016).

As with curvature indices, complexity indices can be analyzed either by aggregating values per participant and condition (using *mt_aggregate_per_subject*) and comparing the result across conditions, or on the trial level using mixed-effects models.

Temporal analyses.

Although many studies use it in this sense, mouse-tracking is not limited to the analysis of aggregate indices that collapse each trajectory to a single value. Analyses of trajectories' temporal development can shed light on the time course of response option activations across the trial and, in particular, how and when different cognitive processes influence the trajectory (Hehman et al., 2015). In the following, we will briefly illustrate some simple use cases.

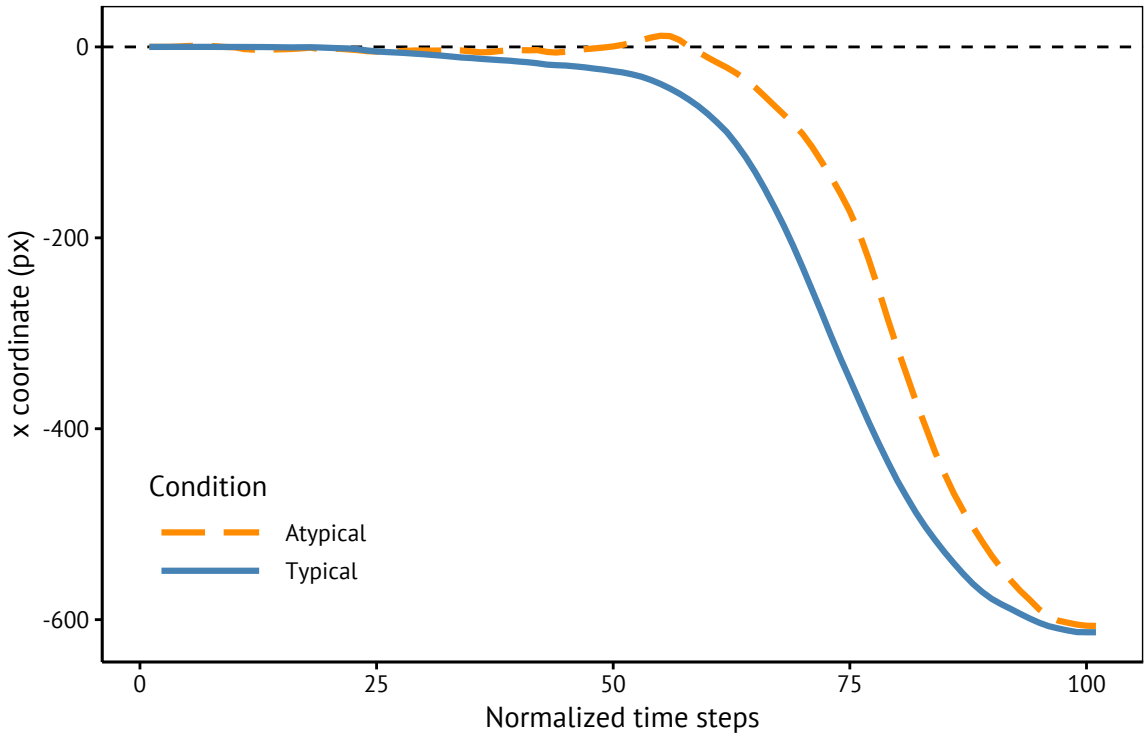

Figure 6. Plot of the average time-normalized x-position over time. For each time step, x-positions were first averaged within participants and condition.

One purpose of temporal analyses is to supplement aggregate analyses of trajectory curvature by showing at which point and for how long aggregate trajectories diverge between conditions. Previous studies (e.g., Dale et al., 2007) have examined this by comparing the horizontal positions of the time-normalized trajectories at each time step using a series of *t*-tests between

conditions (code examples can be found at https://github.com/pascalkieslich/mousetrap-resources). Using this approach reveals that for time steps from 54 to 95 (of 101 steps) the average x- coordinates differed between conditions (Figure 6). If a theory provides specific predictions with regard to the temporal development, for example, whether the divergence between conditions should occur early or late in the decision-making process, this can be used to test them. Note that the comparison of trajectories between conditions can be problematic if response time differences between conditions are large, and that temporal analyses can also be conducted based on raw instead of time-normalized trajectories (see also Hehman et al., 2015).

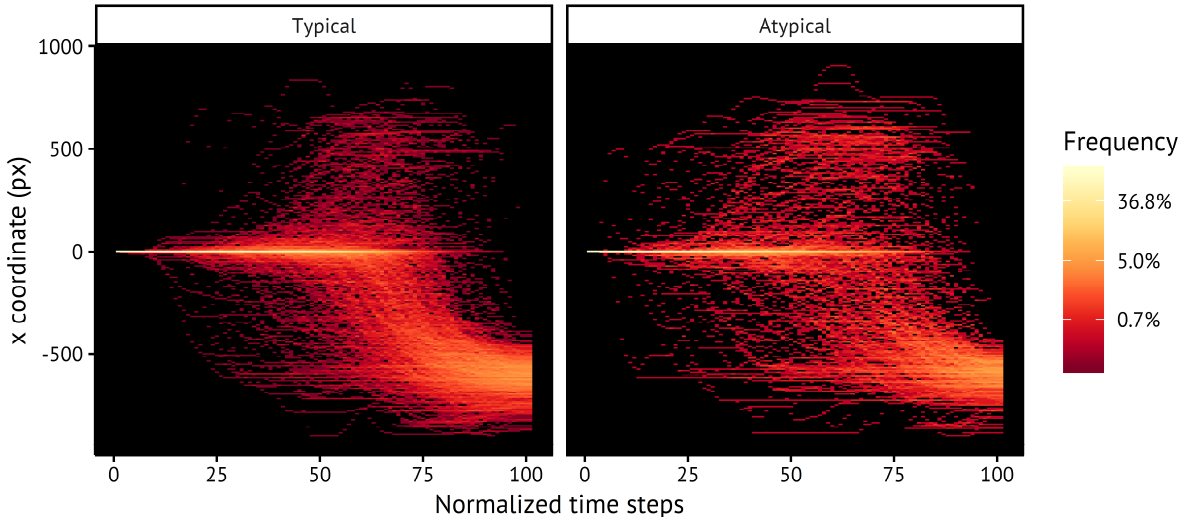

Figure 7. Riverbed plot of the distribution of x-positions across time for time-normalized trajectories separately for the two experimental conditions. For each time step, the colors indicate the relative frequency with which each bin of x-positions was observed.

As with aggregated trial-level indices, aggregated x-positions may not necessarily represent the underlying trial level trajectories well. To inspect whether this is the case it is useful to illustrate the full distribution of trial-level x-positions across normalized time using the *mt_plot_riverbed* function (following an approach by Scherbaum et al., 2010). As can be seen in Figure 7, the aggregate x-positions displayed in Figure 6 are a rather poor representation of the individual trajectories, which vary greatly. Specifically, while the majority of trajectories go directly to the eventually chosen option, a substantial number of trajectories first moves to the non-chosen option (crossing the midline). This means that the data may be better analyzed on the trial level using,

for instance, mixed-effects models or type-based analyses (Wulff et al., 2018; see also Chapter 9). Moreover, Figure 7 reveals that in most trials of both conditions the cursor remained in a neutral position (in many cases it stayed on the start button) for more than half of the trial, a behavior that is probably related to the use of a static starting condition that did not enforce early movement initiation (cf., Scherbaum & Kieslich, 2018).

In addition to analyzing the temporal development of the cursor position, previous mouse-tracking studies have also focused on other variables derived from it, especially velocity, acceleration, and movement angle. The analysis of velocity and acceleration has been used to investigate response activation and competition (Hehman et al., 2015). In mousetrap, velocity and acceleration can be computed using *mt_calculate_derivatives* which attaches velocity and acceleration values to each of the recorded cursor positions.[10] Subsequent analyses can be performed as sketched above, using the velocity values instead of x-positions as the dimension of interest.

Finally, an emerging class of analyses has focused on the movement angle, which quantifies the direction of movement over time indicating, for example, whether participants move towards or away from a specific response alternative. Previous studies have used movement angles to disentangle when and to which extent different factors influence the movement direction (e.g., Dshemuchadse, Scherbaum, & Goschke, 2013; Scherbaum et al., 2010; Sullivan, Hutcherson, Harris, & Rangel, 2015). For details on these approaches, see Scherbaum and Dshemuchadse (2018).

Summary and conclusion

In mouse-tracking studies, participants' cursor movements are recorded as they choose between different options represented as buttons on a computer screen. Thereby, mouse-tracking aims to measure the degree of conflict between the alternatives and the temporal development of its resolution. While Chapter 9 provides a detailed look at the types of trajectories revealed in this paradigm, and Chapter 10 provides an introduction to this method and its use in the literature, this chapter has shown how to construct a mouse-tracking study using the mousetrap plugin for the graphical experiment builder OpenSesame, and how to analyze the resulting data in the

[10] Velocity and acceleration can be calculated for raw trajectories (by default) as well as for time-normalized trajectories. In addition, both can be computed based on the Euclidean distance traveled along the x- and y-dimension (by default) or for a single dimension only.

mousetrap R package. We have covered technical issues surrounding the application of this method, and highlighted design considerations and their influence on the collected data.

The strength of mouse-tracking lies in the ease with which it can be applied. Using only standard laboratory hardware, cognitive processes can be tracked at high temporal resolution. It is also a flexible tool that can be adapted to many different tasks, and which is even more powerful in combination with other process tracing methods (e.g., eye-tracking, cf., Koop & Johnson, 2013; Quétard et al., 2016). Data collection and processing as described in this chapter are handled entirely by free, open-source software (Kieslich & Henninger, 2017; Kieslich et al., 2018), making mouse-tracking easily accessible to interested researchers and transparent to those looking to replicate findings or adapt and extend the methods described herein.

As a fairly recent addition to the family of process tracing methods, many aspects of the method are not yet fully standardized. Therefore, the degrees of freedom with regard to data collection, processing, and analysis are substantial. Where available, we have pointed to the current state of knowledge regarding best practices, which is bound to grow over time. We advise users of mouse-tracking to seek convergence between analyses and indices where no standard has been established so far. In doing so, they should also consider the effects of aggregation by inspecting the distribution of trajectories and indices on the trial level (cf., Chapter 9). While researchers may often explore different experimental setups and analyses if they apply mouse-tracking in a new domain, (additional) pre-registered studies should be conducted to perform strictly confirmatory hypothesis testing (Wagenmakers, Wetzels, Borsboom, van der Maas, & Kievit, 2012).

In sum, we have demonstrated the potential mouse-tracking has as a process tracing method for various areas of decision research. Given the limits of an introductory tutorial, we have only covered the most frequently used analyses. Similarly, the current chapter has limited itself to the frequently investigated two-option design, but mouse-tracking can easily be extended to situations with more than two alternatives (e.g., Koop & Johnson, 2011). Lastly, more sophisticated analysis methods are being developed to more fully harvest the rich potential of mouse-tracking data, such as time continuous multiple regression (Scherbaum & Dshemuchadse, 2018), entropy approaches (Calcagnì, Lombardi, & Sulpizio, 2017), generalized processing tree models (Heck, Erdfelder, & Kieslich, in press), and decision landscapes (Zgonnikov, Aleni, Piiroinen, O'Hora, & di Bernardo, 2017). Thus, we are confident that mouse-tracking will continue to offer researchers novel insights into how decision processes unfold over time.

Recommended reading list

- https://github.com/pascalkieslich/mousetrap-resources: resources for creating mouse-tracking experiments and analyzing mouse-tracking data (including the examples from the current chapter).
- Kieslich and Henninger (2017): an introduction into and validation of the mousetrap plugin for OpenSesame, which also provides detailed information about the example study used in the current chapter.
- Kieslich, Wulff, Henninger, Haslbeck, and Schulte-Mecklenbeck (2018): a detailed tutorial on analyzing hand- and mouse-tracking data using the mousetrap R package.
- Hehman, Stolier, and Freeman (2015): a description of several analytic approaches for mouse-tracking data.

References

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

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.

Dshemuchadse, M., Scherbaum, S., & Goschke, T. (2013). How decisions emerge: Action dynamics in intertemporal decision making. *Journal of Experimental Psychology: General*, *142*(1), 93–100.

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

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.

Freeman, J. B., & Dale, R. (2013). Assessing bimodality to detect the presence of a dual cognitive process. *Behavior Research Methods*, *45*(1), 83–97.

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

Frisch, S., Dshemuchadse, M., Görner, M., Goschke, T., & Scherbaum, S. (2015). Unraveling the sub-processes of selective attention: Insights from dynamic modeling and continuous behavior. *Cognitive Processing*, *16*(4), 377–388.

Heck, D. W., Erdfelder, E., & Kieslich, P. J. (in press). Generalized processing tree models: Jointly modeling discrete and continuous variables. *Psychometrika*.

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.

Kieslich, P. J., & Henninger, F. (2016). Readbulk: An R package for reading and combining multiple data files. Retrieved from https://doi.org/10.5281/zenodo.596649

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

Kieslich, P. J., Wulff, D. U., Henninger, F., Haslbeck, J. M. B., & Schulte-Mecklenbeck, M. (2018). *Mouse- and hand-tracking as a window to cognition: A tutorial on implementation, analysis, and visualization*. Manuscript in preparation.

Koop, G. J., & Johnson, J. G. (2011). Response dynamics: A new window on the decision process. *Judgment and Decision Making*, *6*(8), 750–758.

Koop, G. J., & Johnson, J. G. (2013). The response dynamics of preferential choice. *Cognitive Psychology*, *67*(4), 151–185.

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.

McKinstry, C., Dale, R., & Spivey, M. J. (2008). Action dynamics reveal parallel competition in decision making. *Psychological Science*, *19*(1), 22–24.

Pfister, R., Schwarz, K. A., Janczyk, M., Dale, R., & Freeman, J. B. (2013). Good things peak in pairs: A note on the bimodality coefficient. *Frontiers in Psychology*, *4*, 700.

Quétard, B., Quinton, J. C., Mermillod, M., Barca, L., Pezzulo, G., Colomb, M., & Izaute, M. (2016). Differential effects of visual uncertainty and contextual guidance on perceptual decisions: Evidence from eye and mouse tracking in visual search. *Journal of Vision*, *16*(11), 28.

R Core Team. (2016). *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. Retrieved from https://www.R-project.org/

Scherbaum, S., & Dshemuchadse, M. (2018). Psychometrics based on continuous measures: Exploiting the dynamics of computer mouse movements with time continuous multiple regression. Manuscript submitted for publication.

Scherbaum, S., Dshemuchadse, M., Fischer, R., & Goschke, T. (2010). How decisions evolve: The temporal dynamics of action selection. *Cognition*, *115*(3), 407–416.

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.

Spivey, M. J., & Dale, R. (2006). Continuous dynamics in real-time cognition. *Current Directions in Psychological Science*, *15*(5), 207–211.

Spivey, M. J., Dale, R., Knoblich, G., & Grosjean, M. (2010). Do curved reaching movements emerge from competing perceptions? A reply to van der Wel et al. (2009). *Journal of Experimental Psychology: Human Perception and Performance*, *36*(1), 251–254.

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.

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.

Sullivan, N., Hutcherson, C., Harris, A., & Rangel, A. (2015). Dietary self-control is related to the speed with which attributes of healthfulness and tastiness are processed. *Psychological Science*, *26*(2), 122–134.

Szaszi, B., Palfi, B., Szollosi, A., Kieslich, P. J., & Aczel, B. (2018). Thinking dynamics and individual differences: Mouse-tracking analysis of the denominator neglect task. *Judgment and Decision Making*, *13*(1), 23–32.

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

Wagenmakers, E.-J., Wetzels, R., Borsboom, D., van der Maas, H. L. J., & Kievit, R. A. (2012). An agenda for purely confirmatory research. *Perspectives on Psychological Science*, *7*(6), 632–638.

Wulff, D. U., Haslbeck, J. M. B., & Schulte-Mecklenbeck, M. (2018). *Measuring the (dis-) continuous mind*. Manuscript in preparation.

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.