CREATING MOUSE-TRACKING EXPERIMENTS AND ANALYZING MOUSE-TRACKING DATA

Pascal Kieslich (University of Mannheim) & Dirk Wulff (University of Basel)

Workshop at the EADM Summer School 2018 in Salzburg, Austria

Workshop agenda

Mouse-tracking introduction (Monday)

- General introduction
- Your task
- Develop & present experimental design

Creating mouse-tracking experiments (Tuesday)

- Introduction to OpenSesame & mousetrap-os plugin
- Build & preregister experiment
- Run experiments

Analyzing mouse-tracking data (Wednesday)

- Introduction to R & mousetrap package
- Covering both basic and advanced analyses and visualizations
- Analyze your data

Preparations (before the workshop)

- Read book chapter by Kieslich et al. (in press)
- Outline two example experiments in your group (meeting the outlined requirements) and describe them in a paragraph
- Upload your ideas in one file name 'GroupX.doc'
  onto OSF (Project Ideas)

DAY 1:
MOUSE-TRACKING INTRODUCTION

Pascal Kieslich (University of Mannheim)

Workshop at the EADM Summer School 2018 in Salzburg, Austria

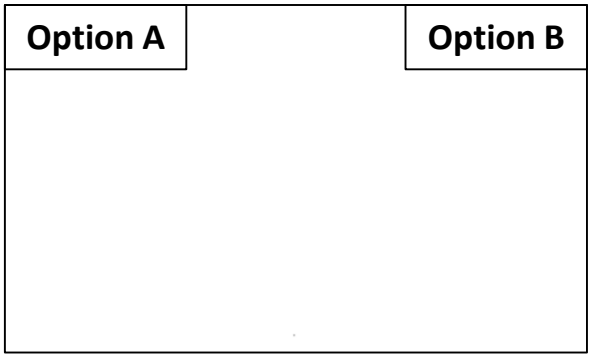
Mouse-tracking introduction (Monday)

- 13:00-14:30 General introduction to mouse-tracking
  - Paradigm and assumptions
  - Implementation and analysis
  - Previous applications
- 14:30-15:00 Introduction to task
  - Type of experiments considered
  - Your tasks during the workshop
- 15:00-17:00 Develop experimental design conceptually
- 17:00-18:00 Present experimental design in plenum

Paradigm & assumptions

Mouse-tracking
Paradigm & assumptions

- **Mouse-tracking** (aka. response dynamics)
  - **Continuous recording** of mouse movements
  - while participants decide between different **spatially separated options** on a screen

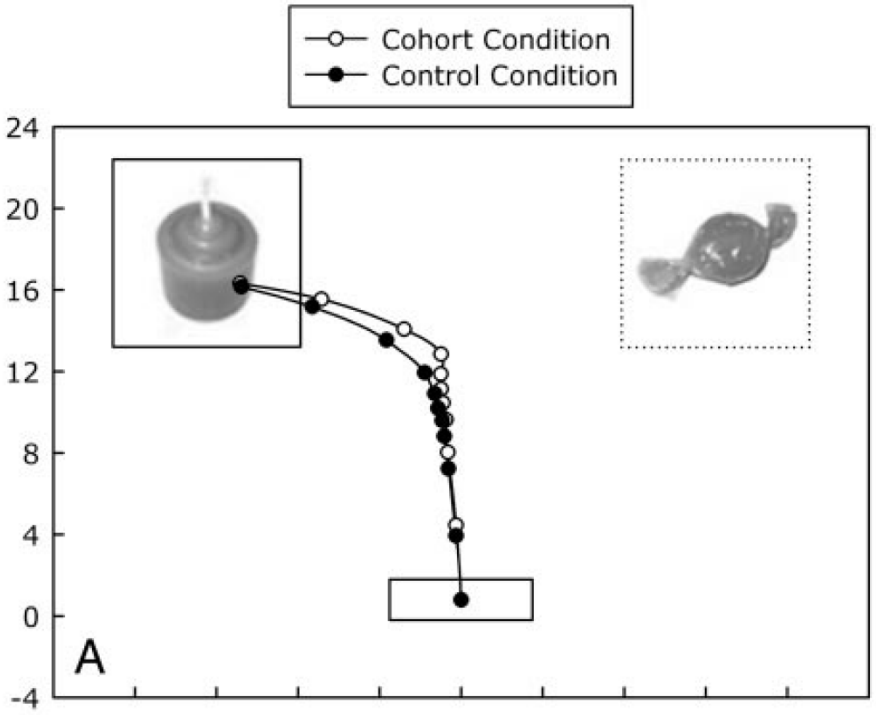


- **Assumptions**
  - Cognitive processing **continuously revealed** in motor responses (Spivey & Dale, 2006)
  - "Hand in motion reveals mind in motion" (Freeman et al., 2011)
    - Mouse movements reveal tentative **commitments** to and **conflict** between choice options during decision process

Mouse-tracking

Seminal article by Spivey et al. (2005)

- Study on **spoken word recognition**
  - Instruction: “Click the **candle**”
- Spatial attraction of hand movement
  - Greater towards phonologically similar distractor (“**candy**”)
  - Than towards phonologically dissimilar distractor (“**dice**”)
- Evidence
  - Suggests parallel processing of auditory input activating competing representations



Mouse-tracking
Main applications

- Mouse-tracking allows for **testing psychological theories**

- Two major applications (cf. review by Stillman et al., 2018)

  - Provides fine-grained measure for **amount of conflict** between response options
    → test predictions about which factors (contextual factors, individual differences) influence amount of conflict for specific decision

  - Assess **temporal development** of conflict and its resolution
    → test models that make predictions how decisions unfold over time (e.g., decide between single vs. dual process models)

Mouse-tracking
Application domains

- Application of mouse-tracking in a **growing number of psychological domains** (Reviews by Freeman, in press; Stillman et al., 2018)
  - Semantic processing (e.g., Spivey et al., 2005; Dale & Duran, 2011)
  - Social cognition (e.g., Freeman et al., 2008; Freeman & Ambady, 2011)
  - Learning and memory (e.g., Dale et al., 2008; Koop & Criss, 2016)
  - Self-control (e.g., Sullivan et al., 2015; Stillman et al., 2017)

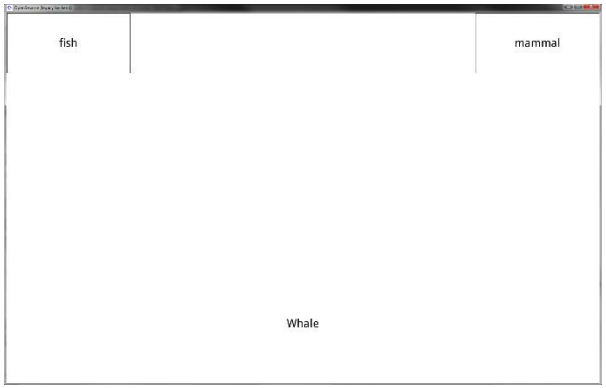
- In the last years also extended to **JDM research**
  - Intertemporal choice (Dshemuchadse et al., 2013)
  - Moral dilemmas (Koop, 2013)
  - Decisions under risk (Koop & Johnson, 2013)
  - Social dilemmas (Kieslich & Hilbig, 2014)
  - Judgmental biases (Szaszi et al., 2018; Travers et al., 2016)

Implementation & analysis

Implementation & analysis

Replication study of Dale et al. (2007)

- **Animal categorization task**
  - **Typical exemplars** only share features with correct category (e.g., cat as mammal)
  - **Atypical exemplars** share both features with correct and competing category (e.g., whale with mammal and fish)



- **Main hypothesis**
  - **Increased competition** when categorizing atypical exemplars
    - Mouse trajectories with deviation towards competing category

- **Replication study** (Kieslich & Henninger, 2017)
  - Same material (translated into German) and procedure, but higher resolution and different aspect ratio
  - *N* = 60 students from the University of Mannheim
  - Material, data, and analyses at https://github.com/pascalkieslich/mousetrap-resources

Implementation & analysis
Methodological considerations

- General challenge when designing a mouse-tracking study
  - Movements should reflect developing **commitment** not information search (≠ eye-tracking or Mouselab)
    → **minimize** amount of **new information** after tracking onset
  - **Preferences** should not develop **before** tracking starts
    → **critical information** should only be made available at the **last** moment

- Mouse **start positions** should be **comparable** across trials
  - Participants have to click on a **centered button** to start the trial
  - Exactly identical start positions across trials achieved by **resetting** mouse or by **computational alignment** during analysis

- **Counterbalancing positions** across trials / participants
  - Vary which option is presented on which side (left vs. right)
  - Can be done between trials or between participants (depending on study)


OpenSesame (legacy backend)
mammal
fish
Whale

Implementation & analysis

Typical analyses steps

Preprocess

- Filter trials
- Alignment of trajectories
- Remapping of trajectories
- Time (and space) normalization

Calculate

- Trial-level indices (e.g., MAD for curvature)
- Dynamic measures (e.g., development of acceleration over time)

Inspect

- Plot of individual trajectories
- Distribution of trial-level indices
- Identification of chaotic trajectories

Aggregate

- Potential aggregation of trajectories and trial measures
- Per participant and condition

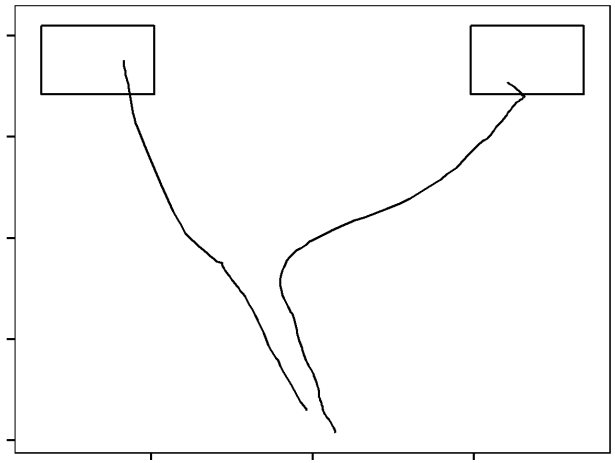
Compare

- Comparison of measures and trajectories
- Typically between experimental conditions in within designs

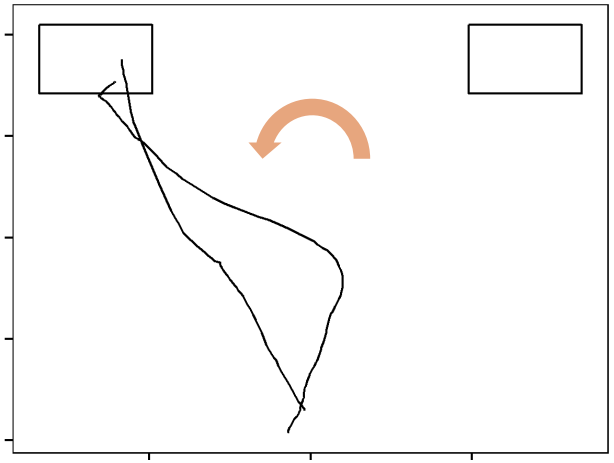
Analyses steps implemented in the mousetrap R package
More information: http://pascalkieslich.github.io/mousetrap/
Available from CRAN: `install.packages("mousetrap")`

Implementation & analysis

Data preparation: Remapping and alignment

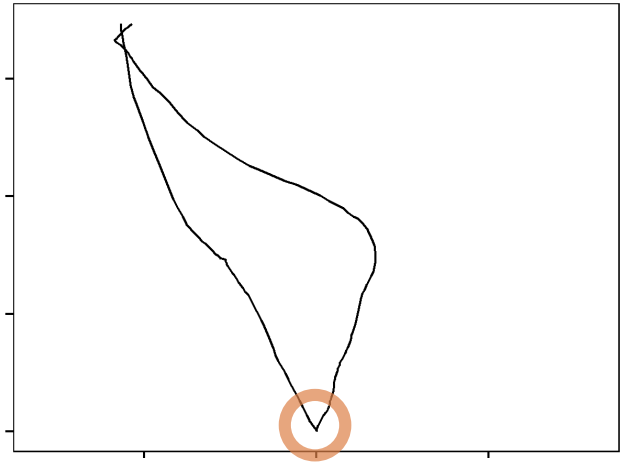


Raw data



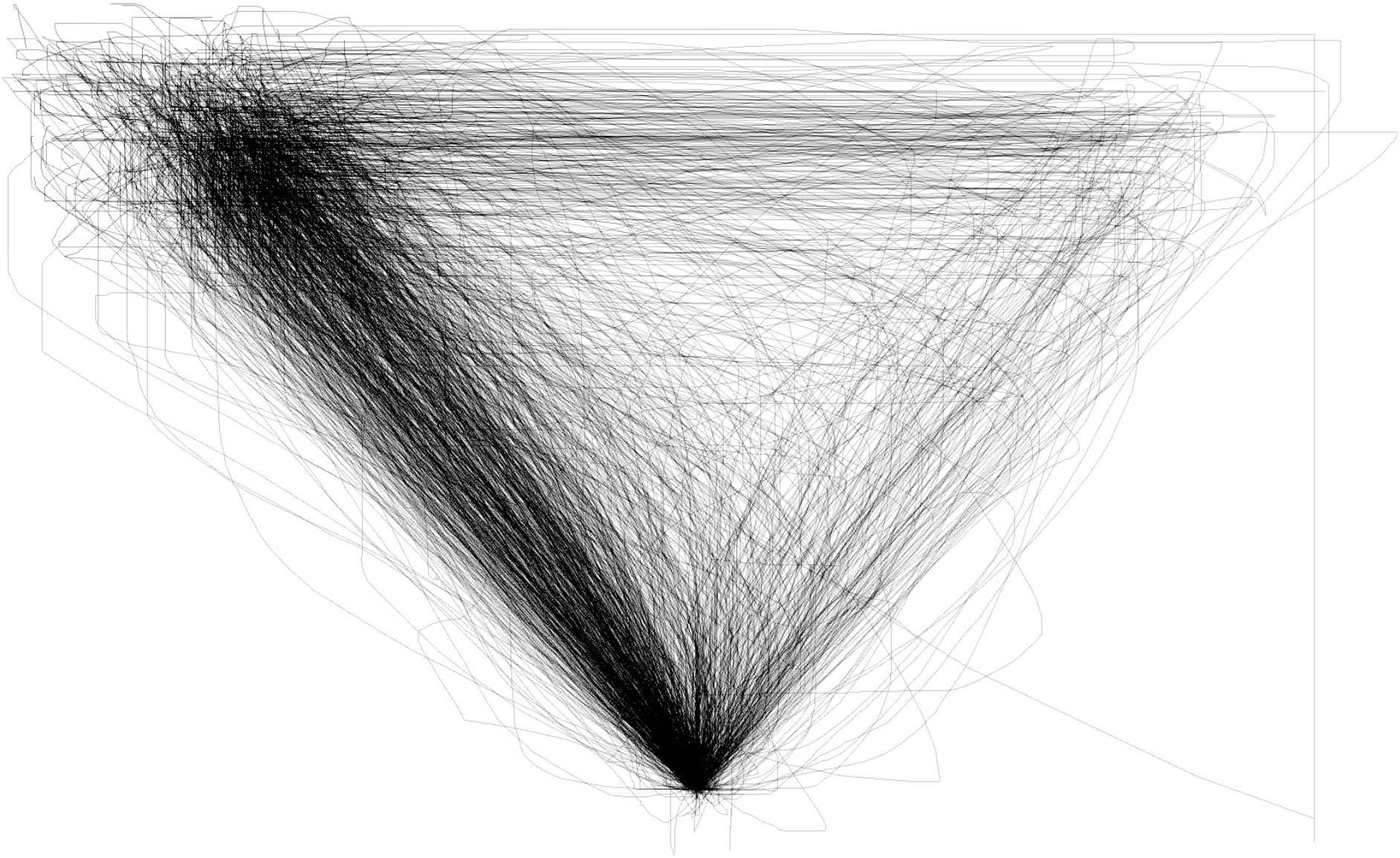
Remapping

Equal direction



Alignment

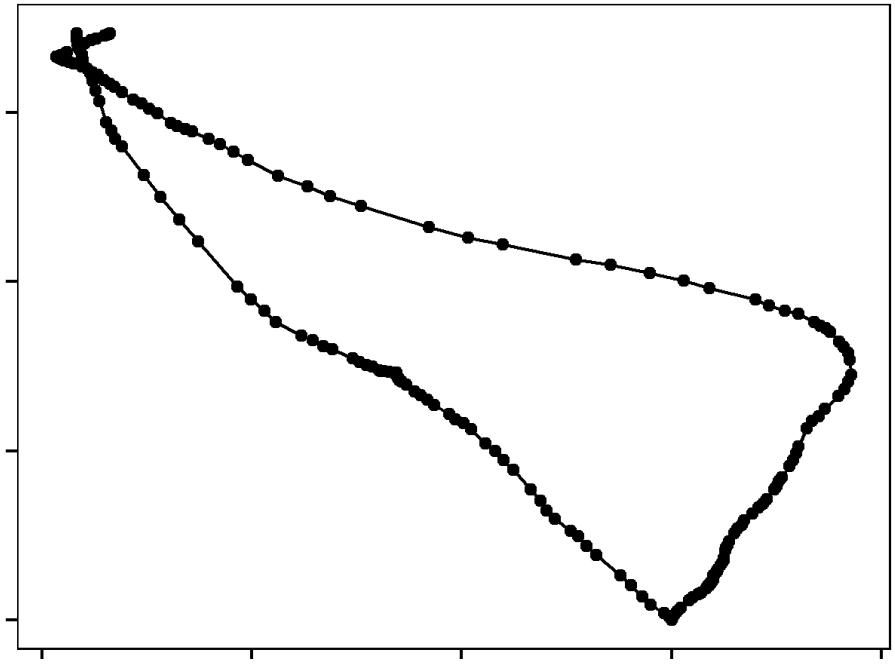
Equal starting point



Implementation & analysis

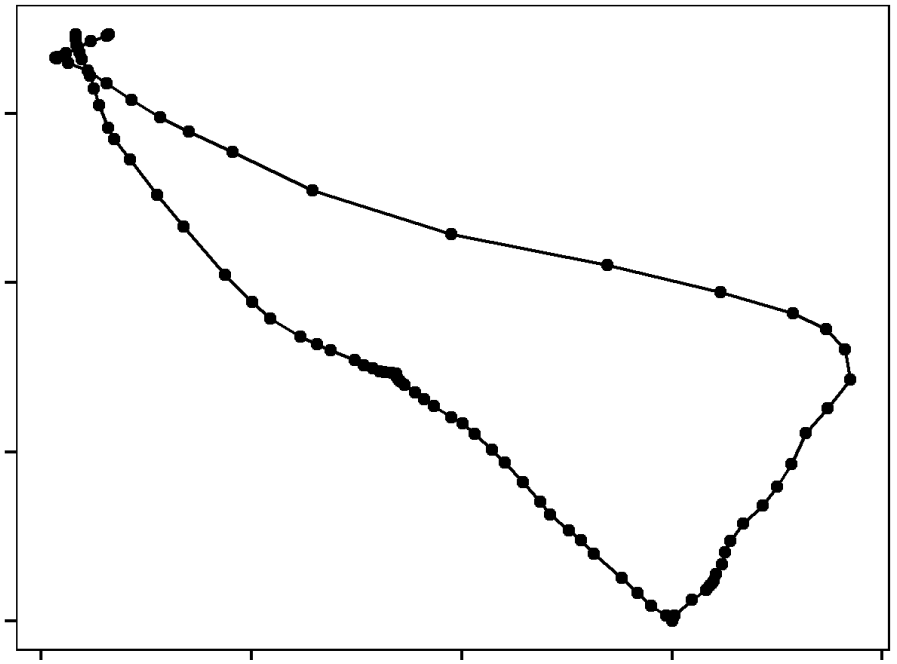
Data preparation: Time normalization

- Trials with differing response time vary regarding number of recorded coordinates
- To permit averaging across trials: time-normalization (cf. Spivey et al., 2005)
- Each trajectory divided into 101 equally spaced time steps using linear interpolation



Raw data

Constant sampling rate → Absolute time

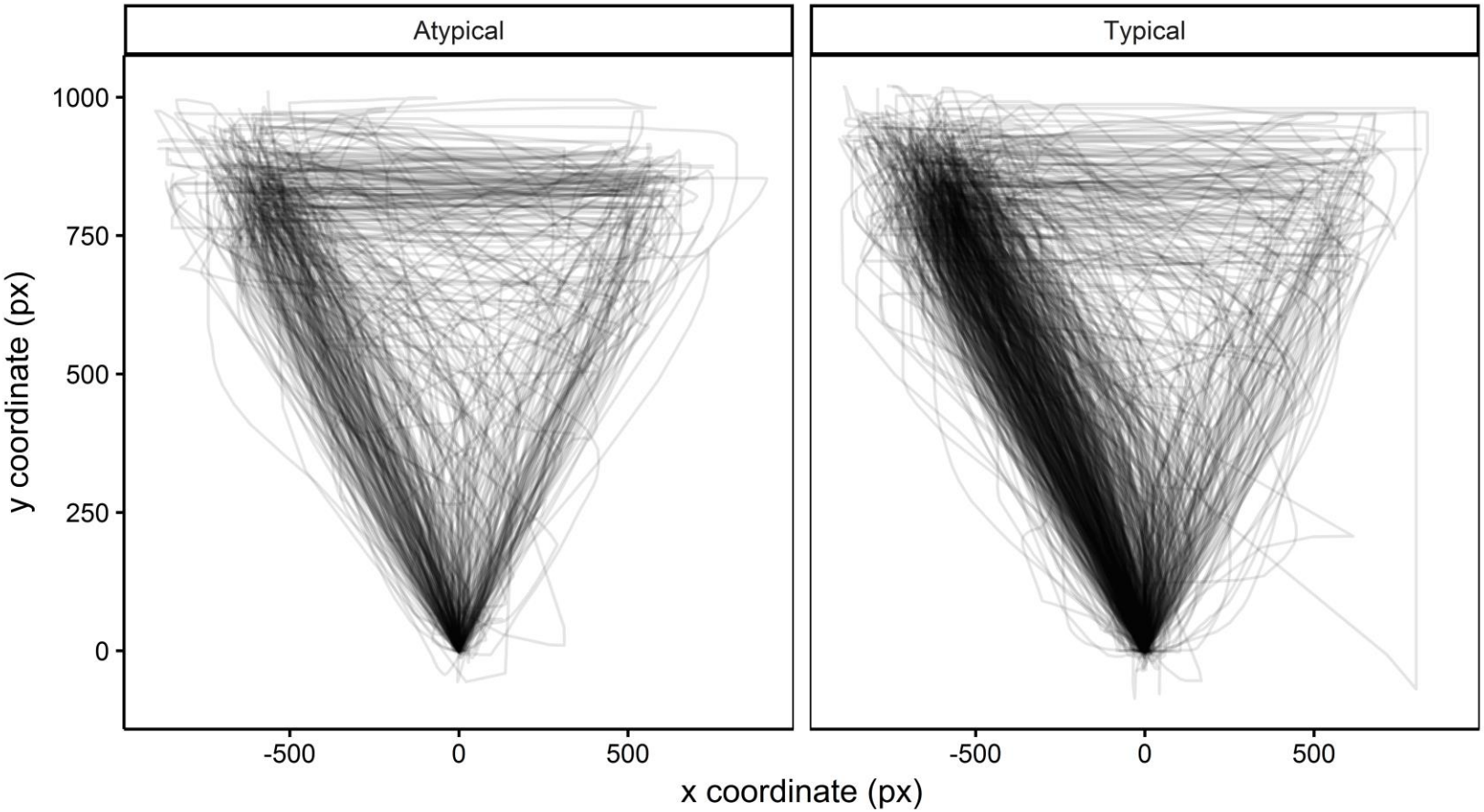


Time normalization

Relative time steps

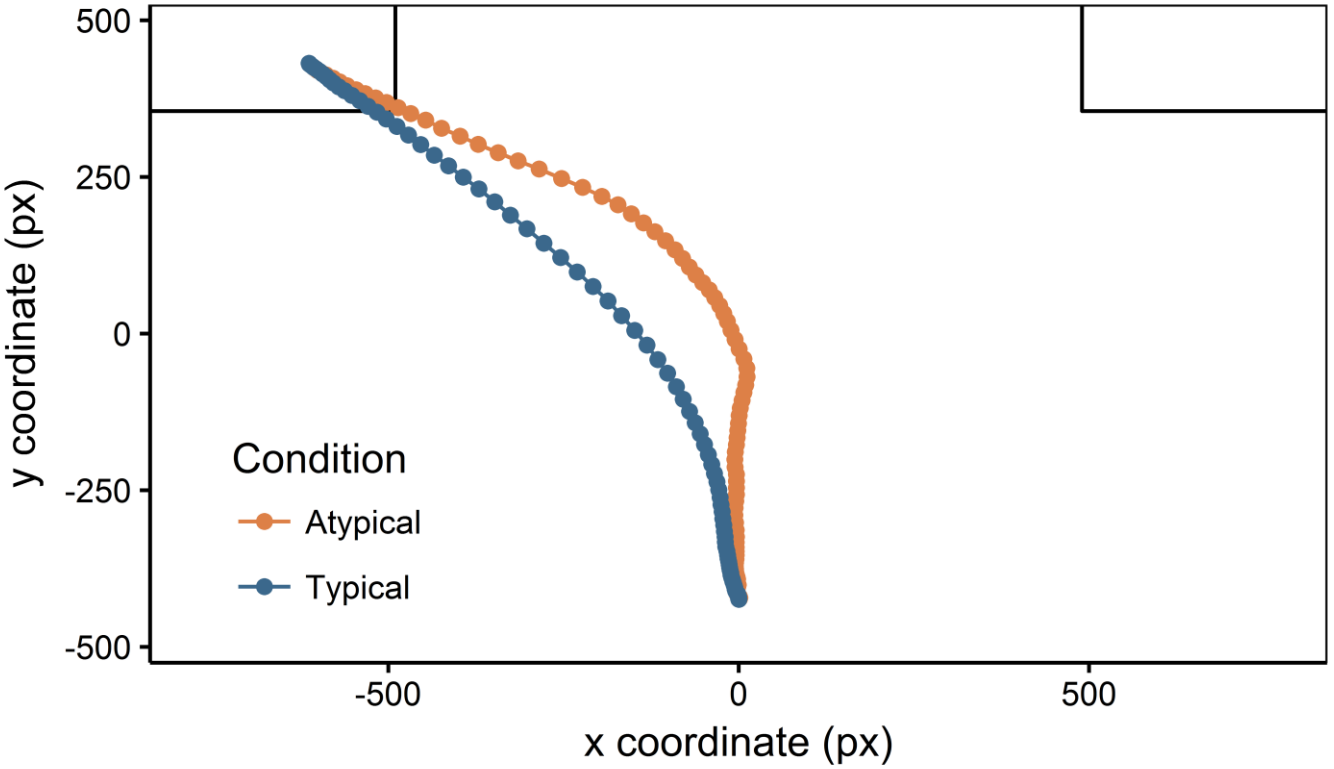
Implementation & analysis

Time-normalized trajectories per condition



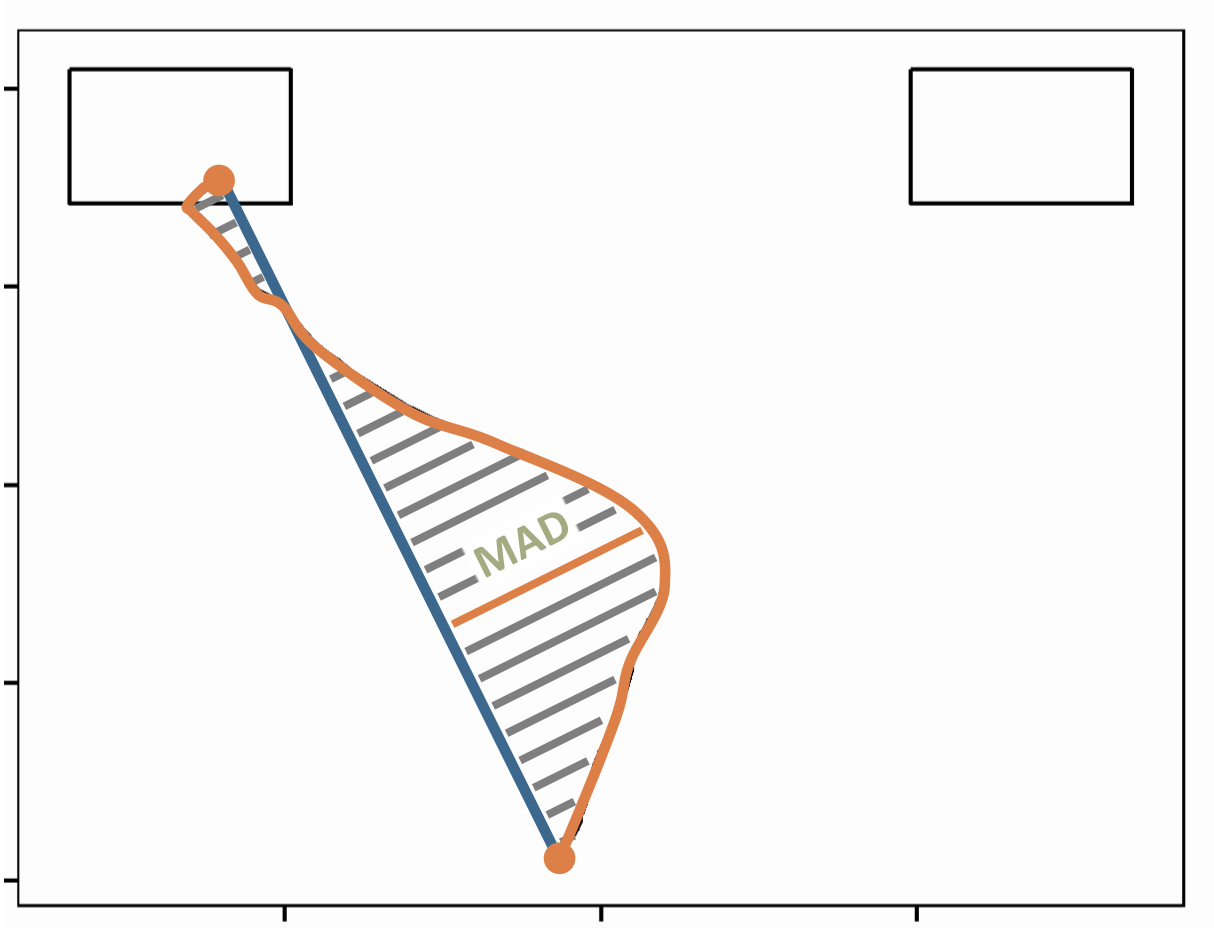
Implementation & analysis

Average time-normalized trajectories



Implementation & analysis

Selected measures for trajectory curvature



Measures of curvature quantify perpendicular distance between **observed trajectory** and an **idealized straight line**

- Maximum absolute deviation (**MAD**)
  McKinstry, Dale, & Spivey (2008)
- Average deviation (**AD**)
  Koop & Johnson (2011)
- Area under curve (**AUC**)
  Spivey, Grosjean, & Knoblich (2005)

Implementation & analysis

Typical analyses steps

- Filter trials
- Alignment of trajectories
- Remapping of trajectories
- Time (and space) normalization

- Trial-level indices (e.g., MAD for curvature)
- Dynamic measures (e.g., development of acceleration over time)

- Plot of individual trajectories
- Distribution of trial-level indices
- Identification of chaotic trajectories

- Potential aggregation of trajectories and trial measures
- Per participant and condition

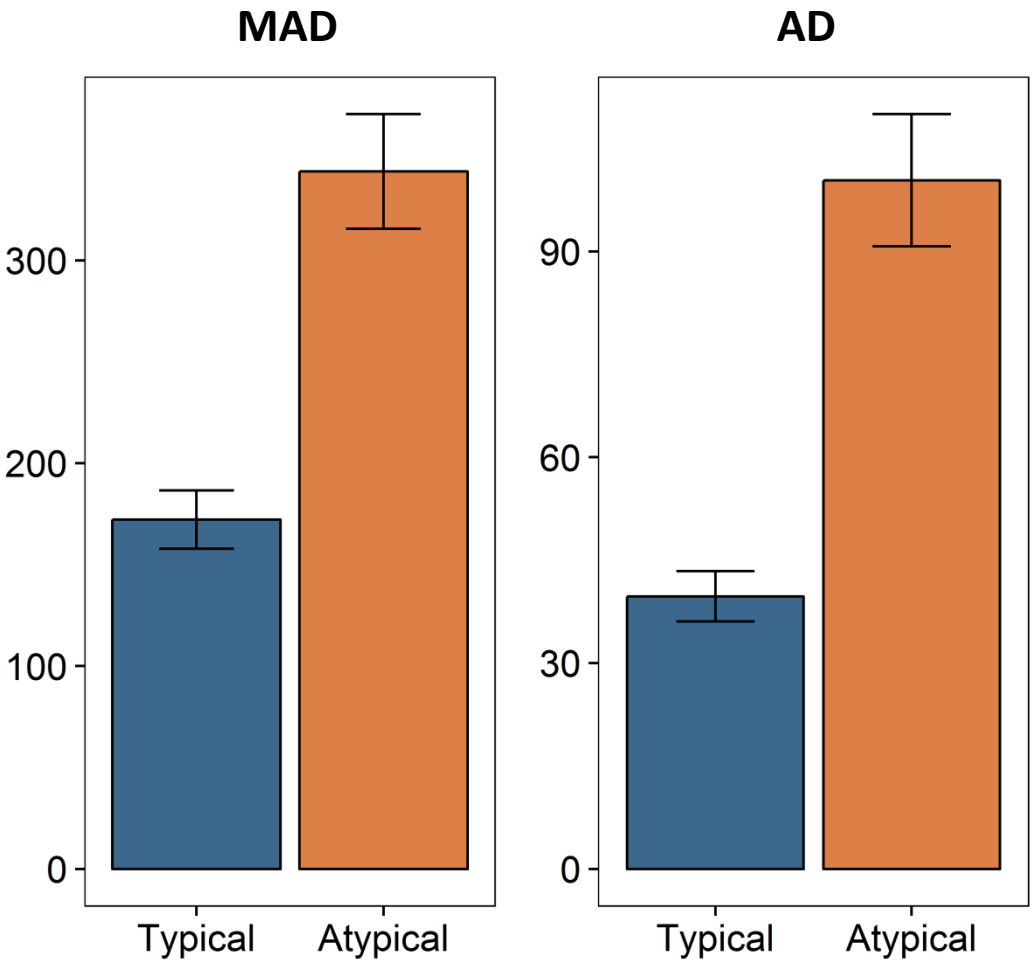
- Comparison of measures and trajectories
- Typically between experimental conditions in within designs

Analyses steps implemented in the mousetrap R package
More information: http://pascalkieslich.github.io/mousetrap/
Available from CRAN: `install.packages("mousetrap")`

Implementation & analysis

Comparison of (maximum) absolute deviations

- **MAD larger for atypical exemplars**
  - $d_z = 0.87$, $p < .001$
  - $BF_{10} = 1.57 * 10^6$
- **AD larger for atypical exemplars**
  - $d_z = 0.87$, $p < .001$
  - $BF_{10} = 1.78 * 10^6$

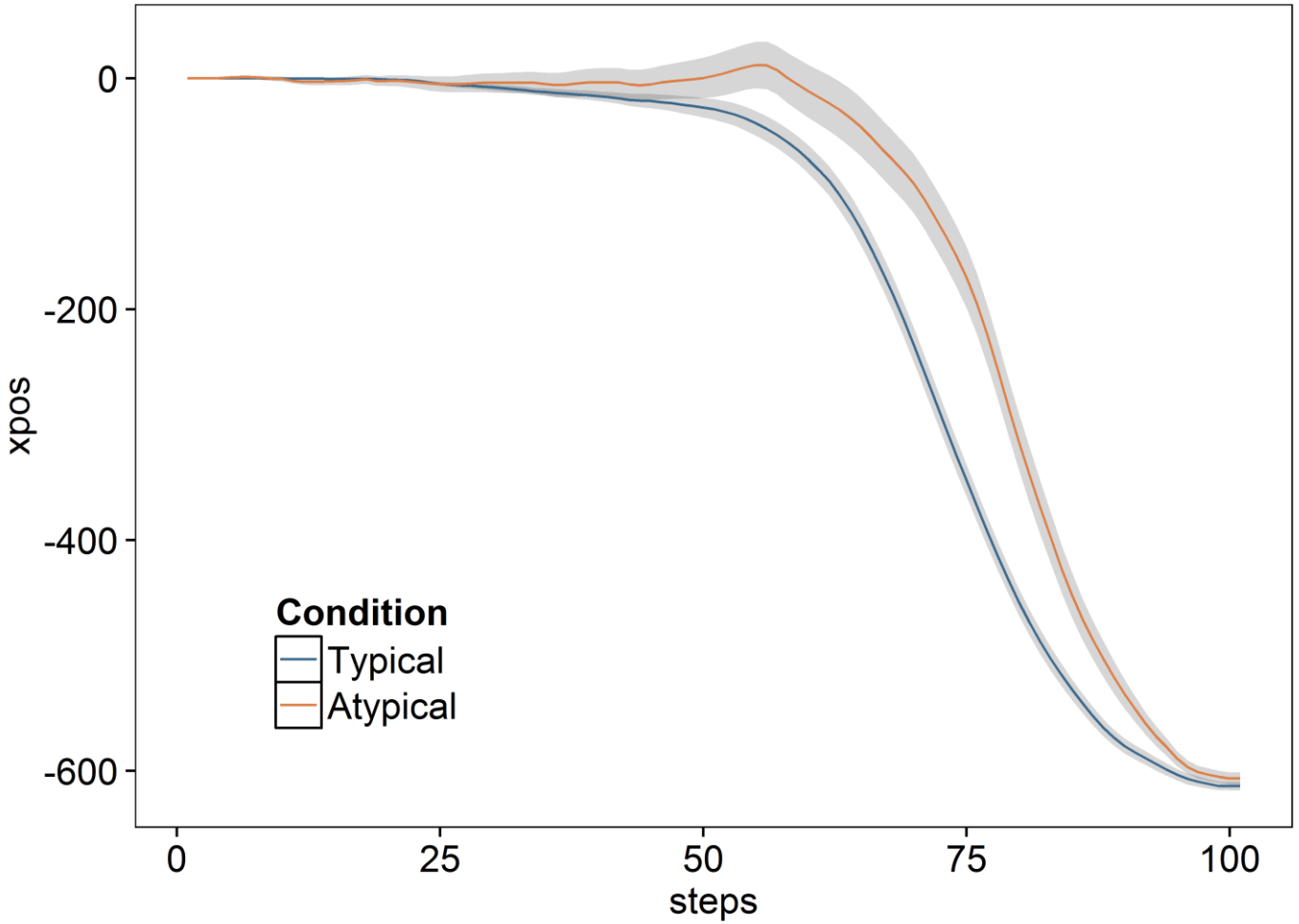


Error bars represent 1 *SEM*.

Implementation & analysis

Average x-positions per time step

- Similar analysis by Dale et al.: Paired *t*-test of average time-normalized x-position per per time step
- Significant differences ($p < .05$) between conditions from time steps 54 to 95



Error bars represent 1 *SEM*.

Implementation & analysis

Selected mouse-tracking measures

Measure	Definition	Possible interpretation	Example
Maximum absolute deviation (MAD)	Maximum deviation from idealized trajectory	Maximum attraction of non-chosen option	McKinstry et al. (2008)
Average Deviation (AD)	Mean deviation from idealized trajectory	Average attraction of non-chosen option	Koop & Johnson (2011)
Area under curve (AUC)	Geometric area between actual and idealized trajectory	Total attraction of non-chosen option	Spivey et al. (2005)
x-flips (xpos_flips)	Number of directional changes along x-axis	Instability, reversal of the momentary valence	Koop & Johnson (2013)
x-reversals (xpos_reversals)	Number of crossings of the y-axis	General reversal of preference	Koop & Johnson (2013)

Implementation & analysis
Analytical and theoretical challenges

- Interpretation of measures still needs to be validated

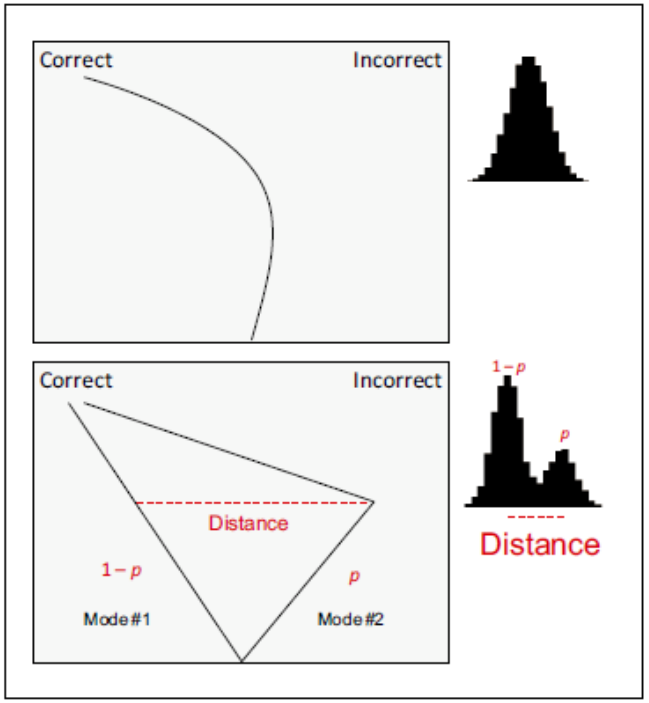
- Multitude of mouse-tracking measures available
  - Often highly correlated in practice
  - There is no standard yet which measure should be used
    → ensure that result does not depend on the specific measure used
    → decide which is the measure of interest a priori / conduct pre-registered replications of your findings

- Consider effects of aggregation by inspecting distribution of trajectories and indices on the trial level

Implementation & analysis

Smooth competition vs. abrupt shifts

- Different assumptions about response process (e.g., Hehman et al., 2015)
  - Single process
    - **"smooth graded competition"** in all trials
    - Continuous competition between response options
  - Dual process
    - **"abrupt shifts" / Change of Mind** in some trials: Initial movement towards one option, then reversal and choice of other option
    - Straight movements in other trials



- Statistical analysis of AUC or MAD distribution (Freeman & Dale, 2013)
  - "smooth graded competition" → unimodal
  - "abrupt shifts" → bimodal

Implementation & analysis

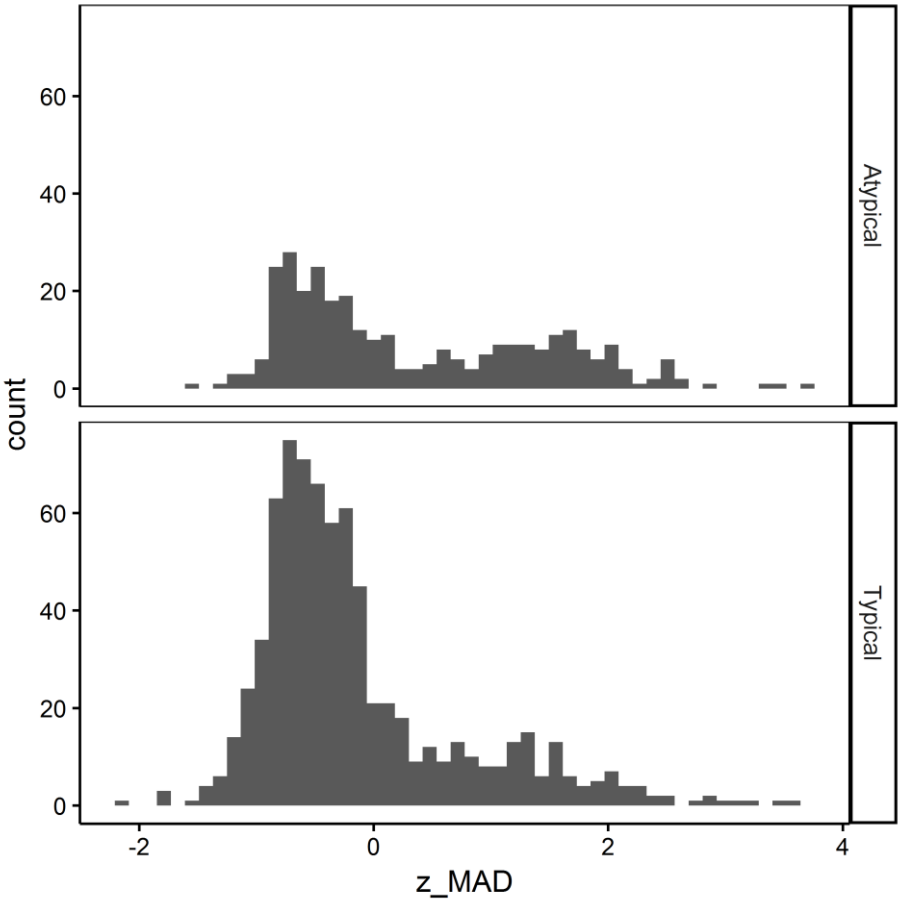
Methods for assessing bimodality and trajectory shapes

- Bimodality coefficient (**BC**, e.g., Pfister et al., 2013)
  - $$BC = \frac{m_3^2 + 1}{m_4 + 3 \cdot \frac{(n-1)^2}{(n-2)(n-3)}}$$
  - Bimodal, if **BC > 0.555**
- Hartigan's dip statistic (**HDS**, Hartigan & Hartigan, 1985)
  - Statistical test (H0: Distribution is unimodal)
  - If ***p* < .05**, distribution is multimodal (i.e., at least bimodal)

Implementation & analysis

Assessment of bimodality

- Distribution of standardized MAD
- **Bimodality coefficient** (*BC*)
  - $BC_{typical}$ = .61; $BC_{atypical}$ = .59
  - Indicates bimodality as *BC* >.555
- Also influenced by setup of study (cf. design factors)



Implementation & analysis

Assessing distribution of individual trajectory shapes

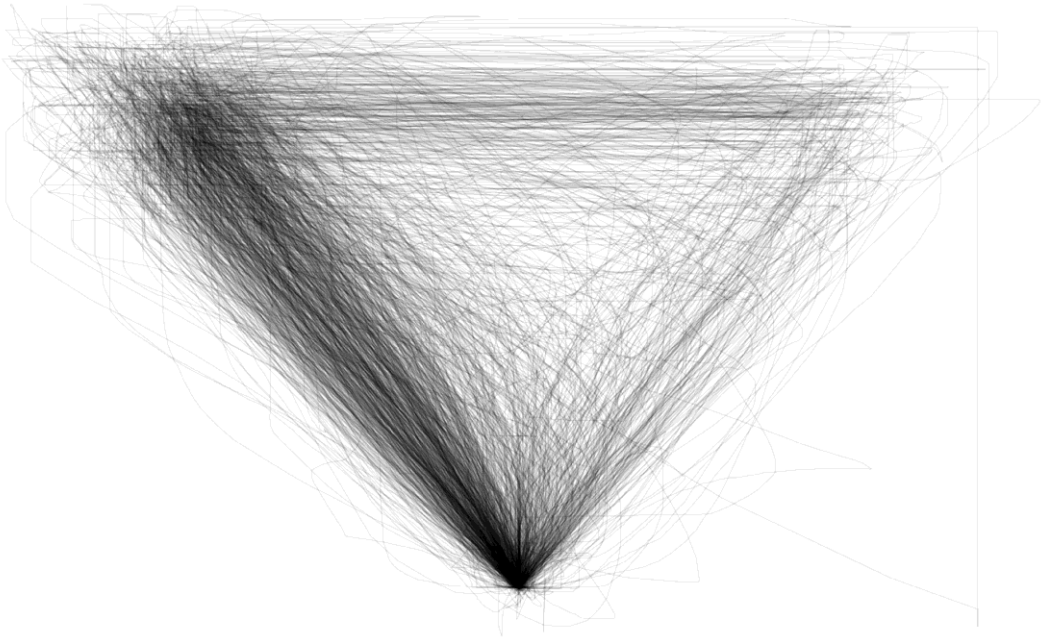
- Assess distribution of trajectory shapes (Wulff et al., in press)
  - **Bimodality analyses** so far focus on a **single measure** only
  - New analyses proposed taking complete **trajectory shape** into account
  - General question: is aggregate trajectory representative of individual trajectories – or are there **different types** of trajectories on the trial level?

- Visualization tools
  - Animations
  - Heatmaps and difference maps

- Analyses tools
  - Clustering
  - Prototype allocation

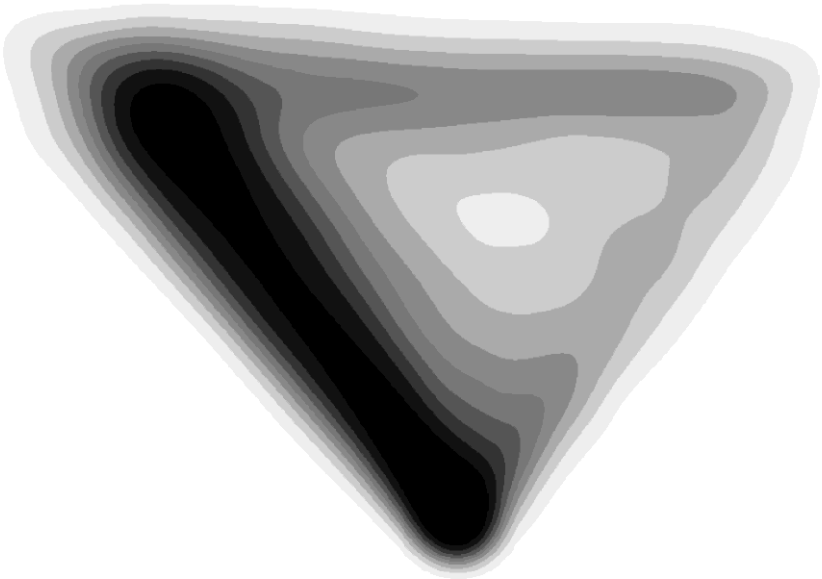
Implementation & analysis

Heatmap of raw trajectories



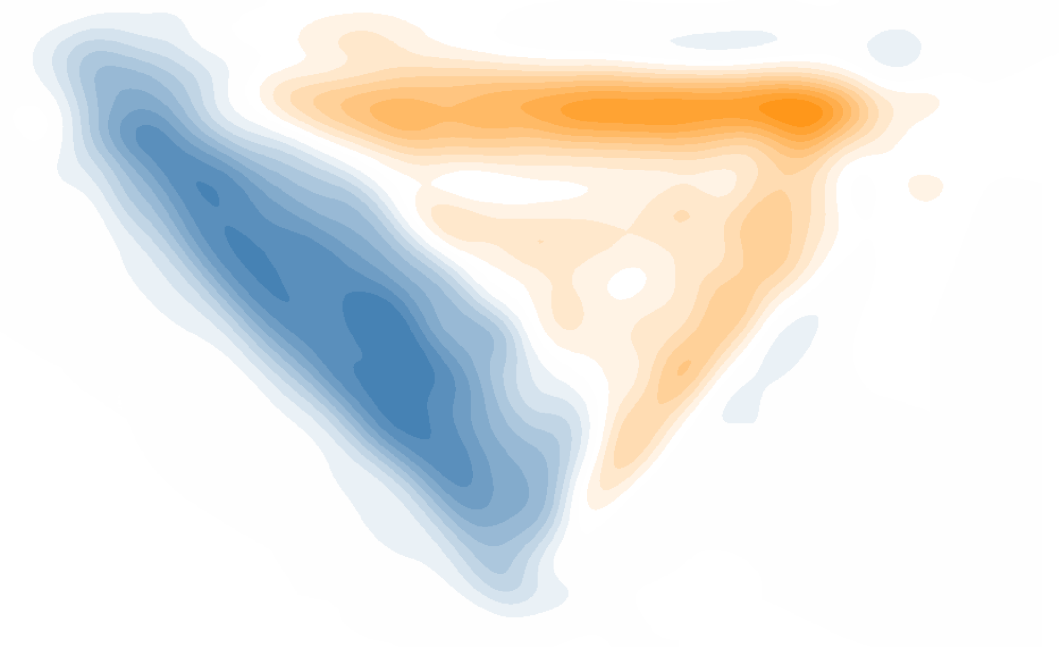
Implementation & analysis

Heatmap of raw trajectories (smoothed)



Implementation & analysis

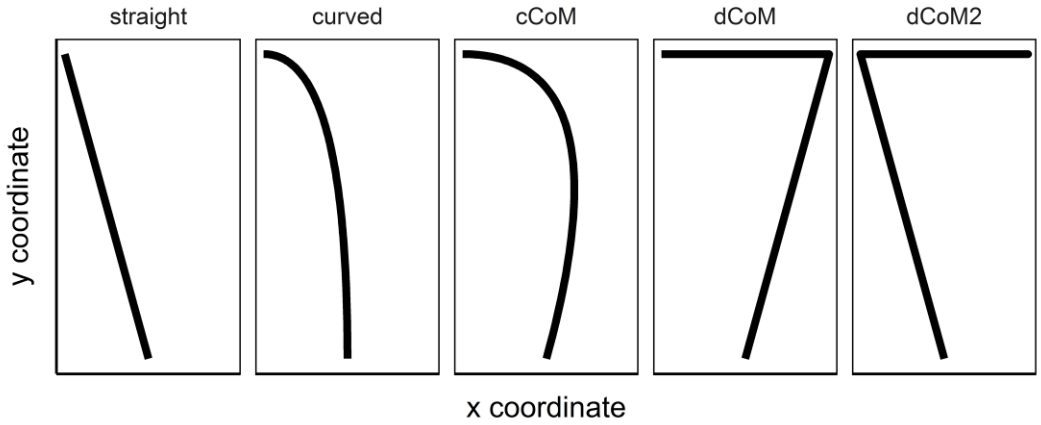
Difference map for typical vs. atypical condition



Implementation & analysis

Prototype recognition (Wulff et al., in press)

- **Specify set of prototypes**
  - Set of prototypes based on clustering results of the meta-analysis by Wulff et al. (in prep.)

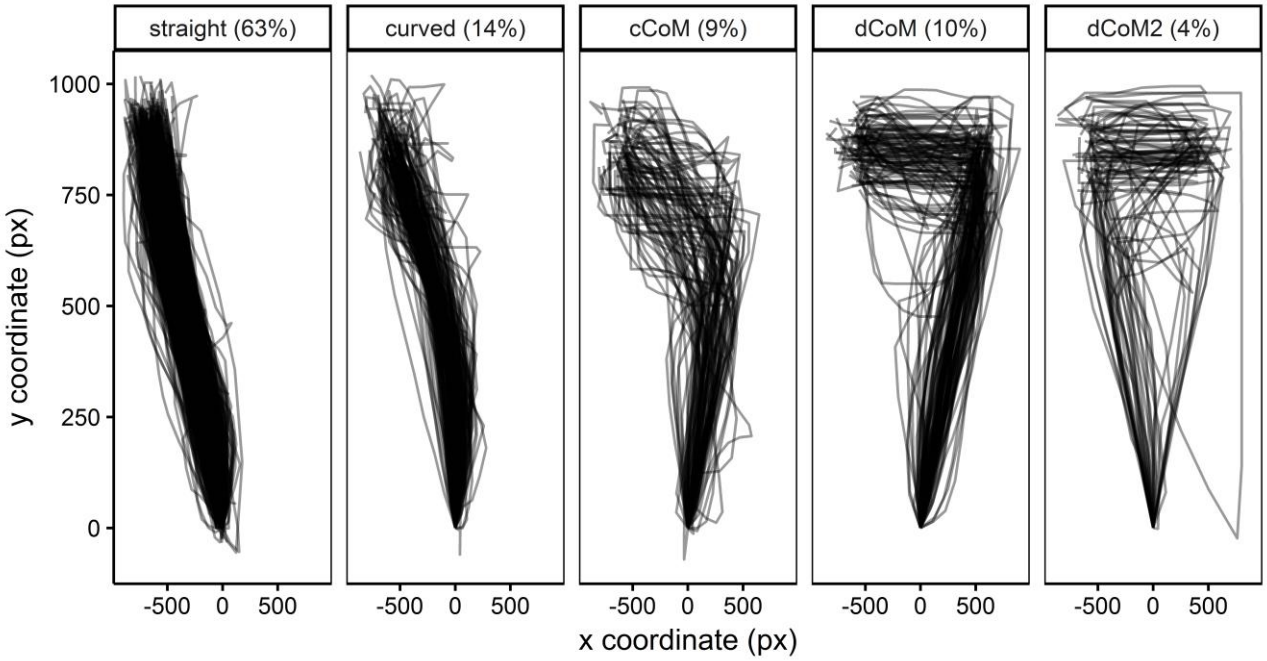


- **Spatialize trajectories**
  - Resample trajectories to small number of points distributed equally across space

- **Assign trajectories to prototypes**
  - Compute dissimilarity between every trajectory and prototype
  - Assign trajectory to prototype with smallest distance
  - (Potentially exclude trajectories where smallest distance is too large)

Implementation & analysis

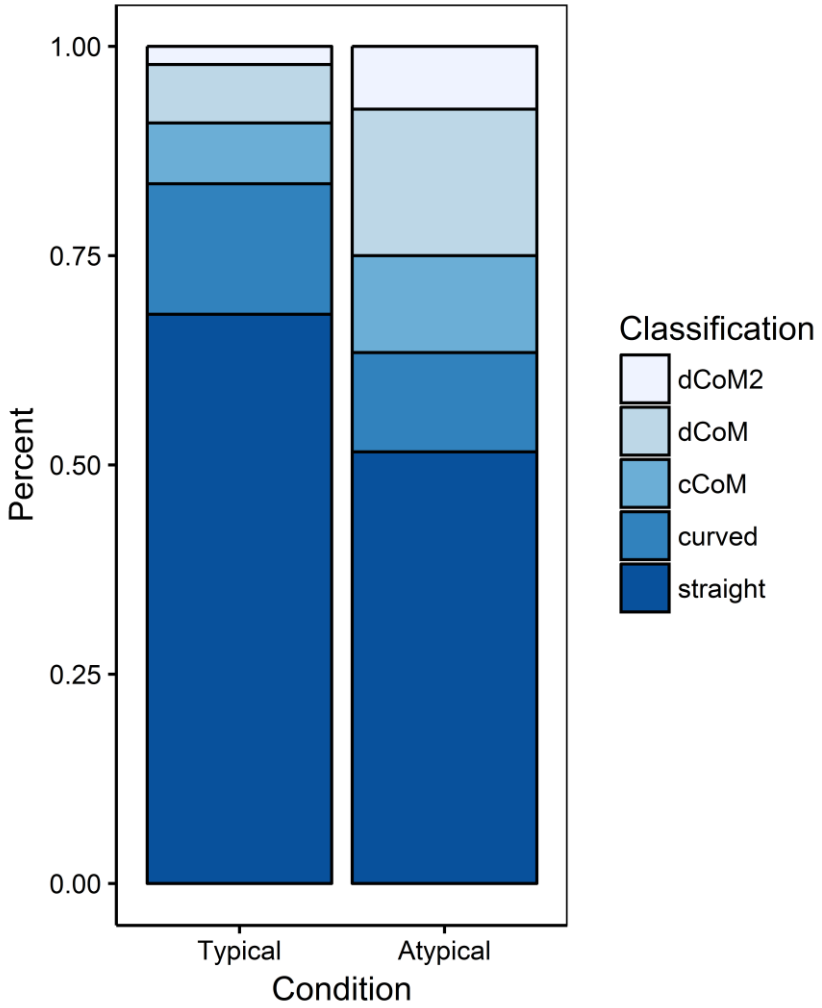
Prototype allocation for replication experiment



Implementation & analysis

Classification frequencies per condition

- Relative frequency of prototype classification differs for conditions
  - $\chi^2 = 57.9$, $p < .001$
- Atypical condition predicts occurrence of types that indicate more conflict
  - in ordinal mixed regression model on trial level
  - with random intercept per participant
  - $z = 6.74$, $p < .001$



Previous applications

Focusing on JDM research

Mouse-tracking
Application domains

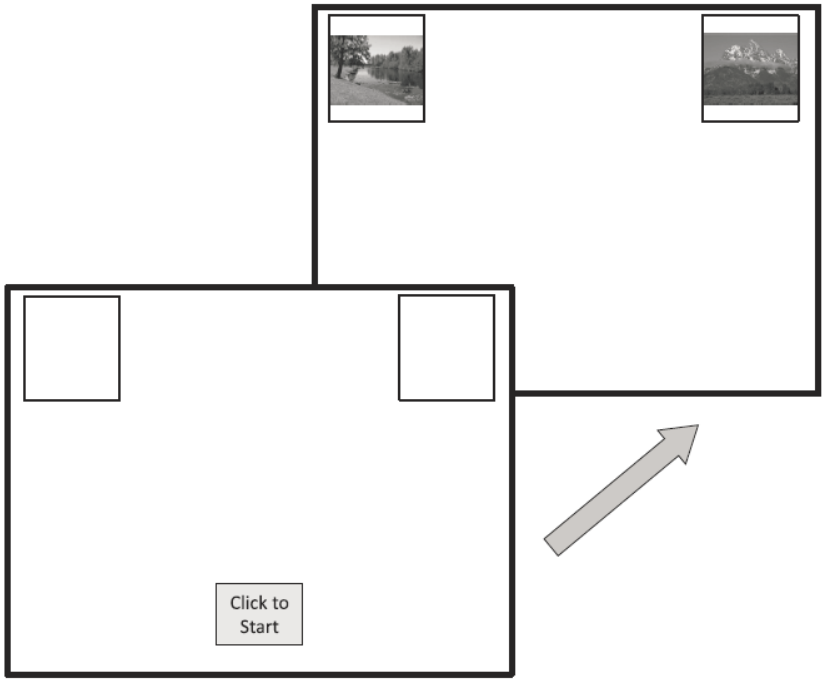
- Application of mouse-tracking in a **growing number of psychological domains** (Reviews by Freeman, in press; Stillman et al., 2018)
  - Semantic processing (e.g., Spivey et al., 2005; Dale & Duran, 2011)
  - Social cognition (e.g., Freeman et al., 2008; Freeman & Ambady, 2011)
  - Learning and memory (e.g., Dale et al., 2008; Koop & Criss, 2016)
  - Self-control (e.g., Sullivan et al., 2015; Stillman et al., 2017)

- In the last years also extended to **JDM research**
  - Intertemporal choice (Dshemuchadse et al., 2013)
  - Moral dilemmas (Koop, 2013)
  - Decisions under risk (Koop & Johnson, 2013)
  - Social dilemmas (Kieslich & Hilbig, 2014)
  - Judgmental biases (Szaszi et al., 2018; Travers et al., 2016)

Preferential decision making

Validation experiment (Koop & Johnson, 2013, Exp. 1)

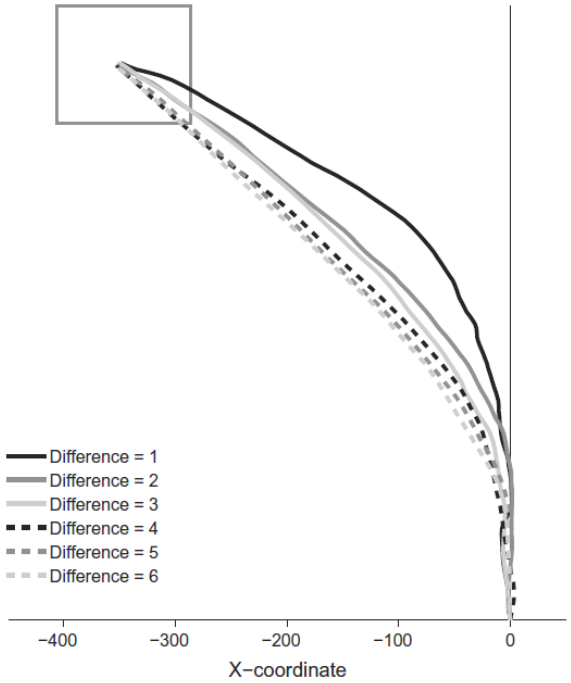
- Decisions between affective images
  - Task: Which of two images do you prefer?
  - Pictures from IAPS database: provides norms for pleasantness ratings
  - Creation of pairs where difference in preferences is systematically varied

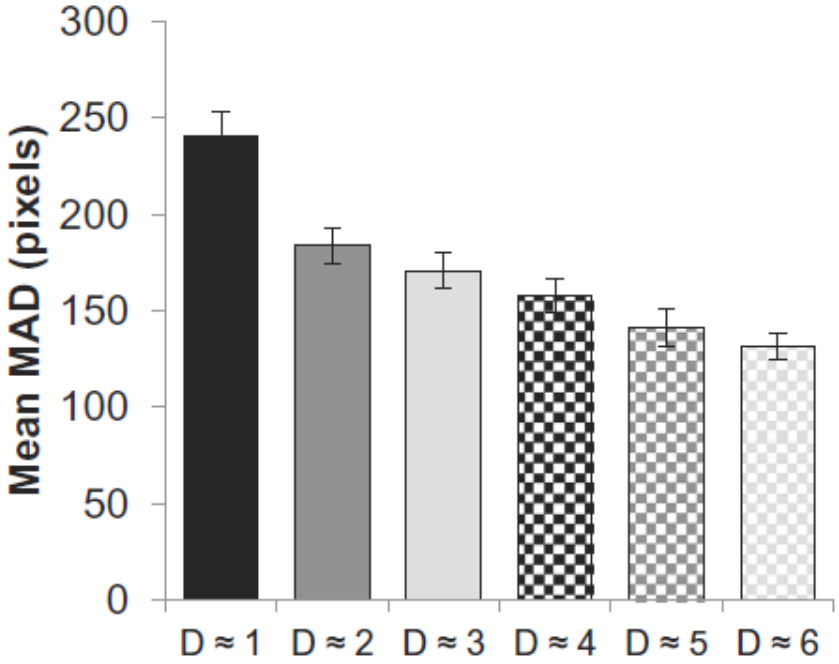


Preferential decision making

Validation experiment (Koop & Johnson, 2013, Exp. 1)

- Decisions between affective images
  - Increase in difference of a priori preference ratings leads to
    - Decrease in trajectory curvature
    - Decrease in maximum absolute deviation (MAD)





Social dilemmas
Basic structure

- Social dilemma (Dawes 1980; Van Lange et al., 2013)
  - Individuals can choose between two options
    - Defection
    - Cooperation

- Standard social dilemma: **Prisoner's dilemma game** (PDG; Rapoport & Chammah, 1965)

		Player 1	
		cooperates	defects
Player 2	cooperates	**100** | **100**	**200** | **0**
	defects	**0** | **200**	**50** | **50**

Social dilemmas

Spontaneous cooperation?

- **Theoretical proposition** (Rand et al., 2012, 2014)
  - People are **spontaneously** inclined to **cooperate**
  - **Defection** requires effortful **deliberation**

- **Empirical test using response times**
  - Idea: **spontaneous = fast**, **deliberative = slow**
  - Mixed results (e.g., Rand et al., 2014; meta-analysis by Rand, 2016; Registered replication report, 2017)
  - Other factors may influence speed (e.g., guessing, information search)

- **Experiment using mouse-tracking** (Kieslich & Hilbig, 2014)
  - When deciding to **defect**, mouse movements should be **more curved** towards non-chosen option (cooperation)
  - When deciding to **cooperate**, mouse movements should be **less curved** towards non-chosen option (defection)

Social dilemmas

Mouse-tracking experiment (Kieslich & Hilbig, 2014)

- Lab experiment (*N* = 115)
  - at the University of Mannheim
  - implementation in OpenSesame (Mathôt et al., 2012) in combination with
    - mousetrap plug-ins for mouse-tracking (Kieslich & Henninger, 2017)
    - psynteract plug-ins for interactive experiments (Henninger, Kieslich, & Hilbig, 2017)

- Participants play 15 two-person social dilemma games
  - without receiving feedback
  - random order
  - incentivized (5 interactions paid out, Ø payout: 2.56 €)

- Social dilemma games
  - 5 x prisoner‘s dilemma game (PDG)
  - 5 x chicken game
  - 5 x stag hunt game

Option A

Option B

Decision 9 of 15

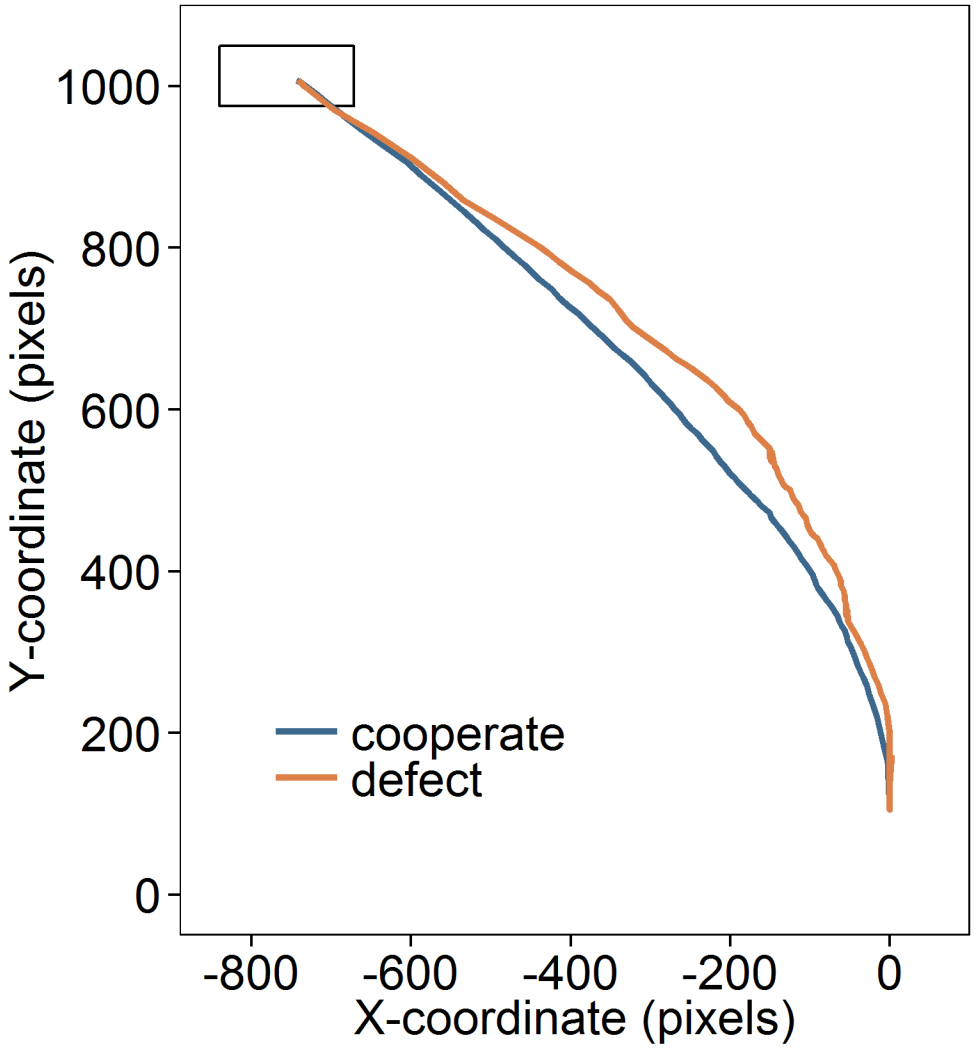
	You choose Option A	You choose Option B	
Person 2 chooses Option A	100 | 100	200 | 0	Person 2 chooses Option A
Person 2 chooses Option B	0 | 200	50 | 50	Person 2 chooses Option B
	You choose Option A	You choose Option B	

Please choose
between Option A and B.

Start

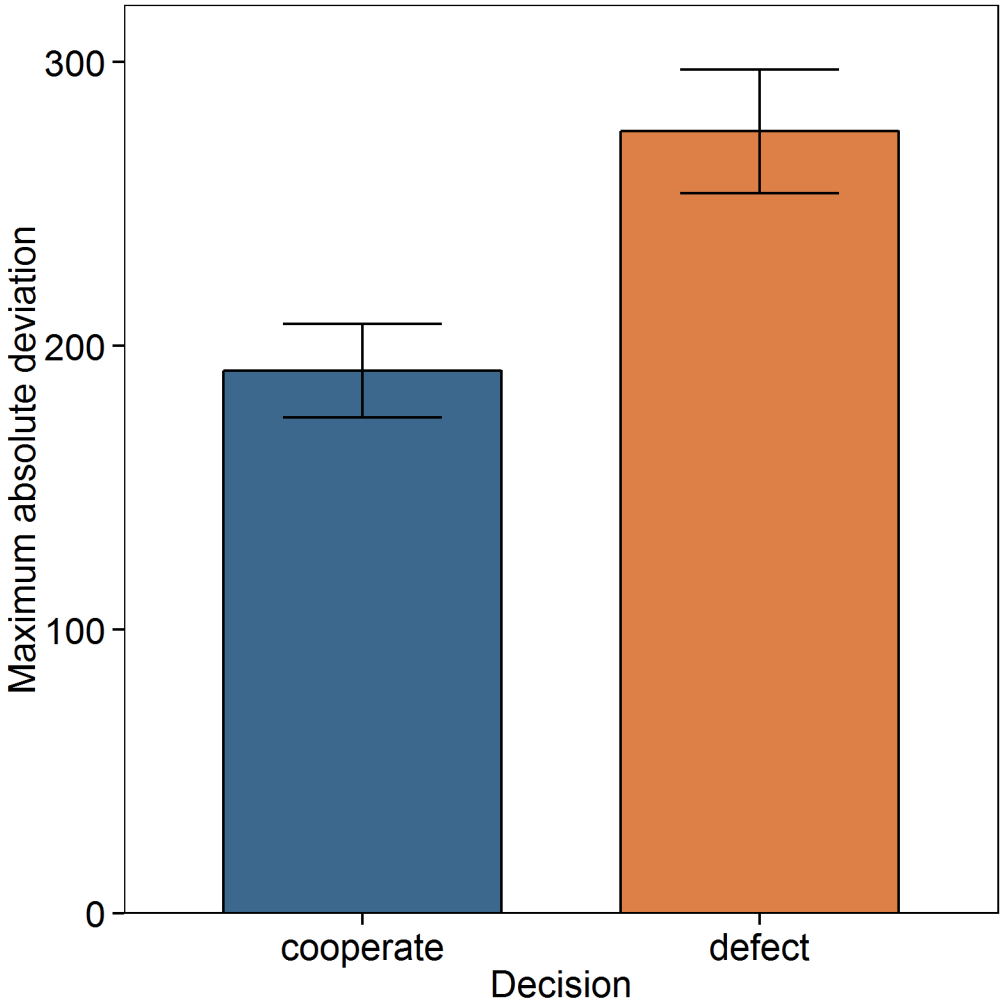
Social dilemmas

Average time-normalized response trajectories



Social dilemmas

Maximum deviation per decision

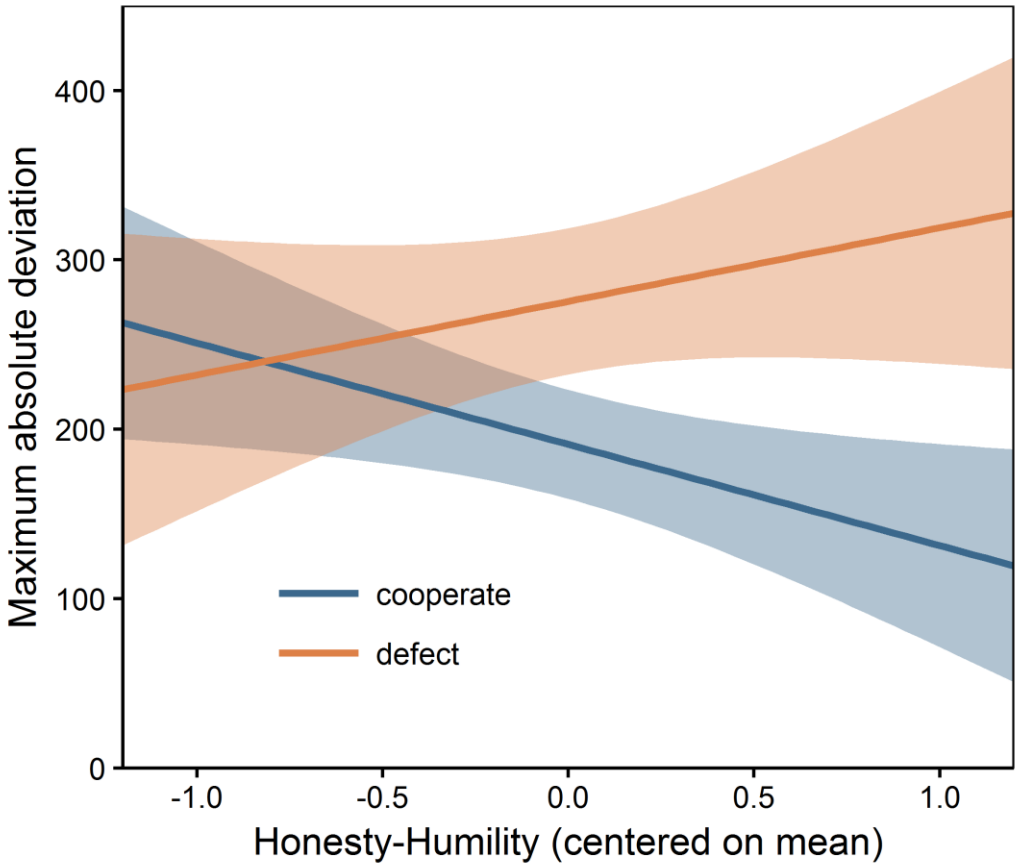


- Main effect of decision
  - MAD significantly higher for defection than for cooperation
- Effect replicated
  - With different measures
  - With filtered trials
  - With linear mixed model on trial level

Social dilemmas

Predicting individual differences in conflict

- Individual differences in conflict: Differences should be stronger for individuals high in Honesty-Humility
  - Dispositional cooperativeness
  - Basic personality factor in the HEXACO personality model (Ashton & Lee, 2007)
- Significant interaction between HH and decision



Confidence bands represent 95% CI.

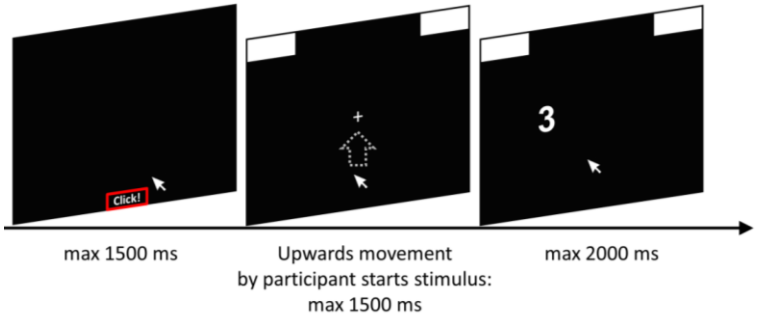
Social dilemmas
Mouse-tracking challenges

- **Experimental control over comparison dimension**
  - Mouse-tracking tasks usually involves “correct”/desired response option + comparison dimension is experimentally manipulated
  - Here **final choice** constitutes **comparison dimension of interest**
    - loss of experimental control
    - use of different games to achieve variation in cooperation rates
- **Complexity and amount of information**
  - **Amount of information** and **complexity of decision** considerably higher than in previous tasks
  - Mouse movements more noisy (e.g., reading movements in some trials)
    - Current solution: analyses replicated with and without problematic trials
    - Ideal solution: simpler task design with less information
      → working on conceptual replication in binary public goods game, also taking into account the newly proposed analytical approaches (prototype mapping)

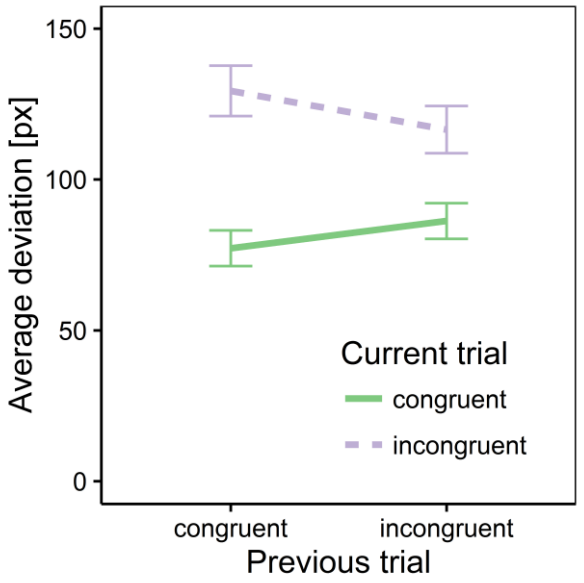
Action selection

Simon effect and conflict adaptation (Scherbaum et al., 2010)

- Mouse-tracking in Simon task
  - Participants have to **click on left vs. right** option depending on the stimulus (e.g., left if number < 5, otherwise right)
  - Position of stimulus varied (left vs. right) so that desired response and position are either **congruent** or **incongruent**



- Results
  - **Simon effect**: larger deviations in incongruent than in congruent trials
  - **Conflict adaptation**: Simon effect reduced if previous trial was incongruent



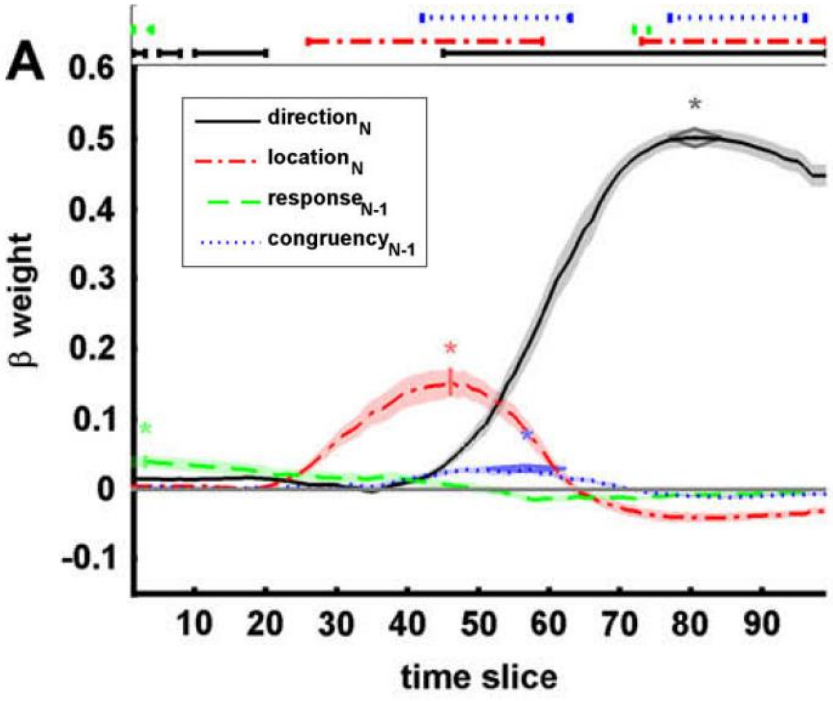
Reanalysis from Scherbaum & Kieslich (in press)

Action selection

Time continuous multiple regression (Scherbaum et al., 2010)

- **Time continuous multiple regression**
  - Criterion: mouse movement angles on the XY plane (≈ **movement direction**)
  - Separate regressions per time step and participant
  - Reveals **temporal order** and strength with which each predictor influences preference development

- **Predictors**
  - Task relevant
    - **Direction** (left/right)
  - Task irrelevant
    - **stimulus location** (left/right)
    - **previous response** (left/right)
    - **congruency sequence** (same/different)



Average β weights per time step and predictor.

Design factors
Overview

- Researchers face a number of **design choices** when creating mouse-tracking experiments
  - Starting procedure (static, restricted initiation time, dynamic)
  - Cursor speed settings (velocity & acceleration)
  - Indicate response via click vs. touch
- Some authors have given **recommendations** about designing mouse-tracking studies (Fischer & Hartmann, 2014; Hehman et al., 2015)
- Empirical **validation** studies are being conducted (Scherbaum & Kieslich, in press; Kieslich et al., in preparation)

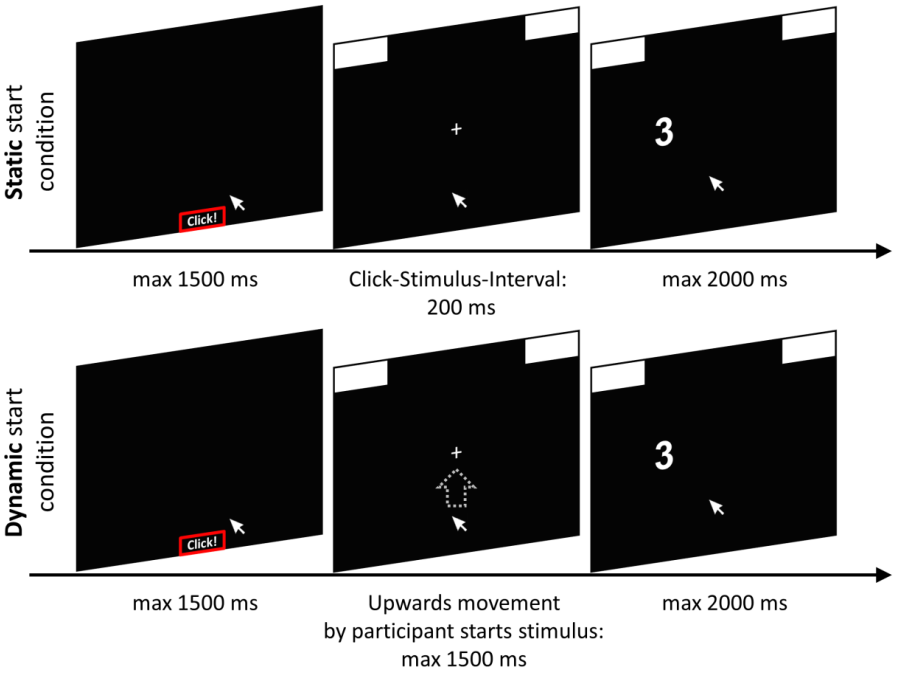
Design factors
Preliminary summary of findings

- **Response indication**
  - **Click** on button leads to larger effects than **touch** – effect related to higher proportion of trials with extreme movements to non-chosen option
- **Mouse sensitivity settings**
  - Did not significantly influence effect of interest in static setup – although **default** settings generally lead to more extreme curvature than **reduced mouse speed**
  - Reducing mouse speed becomes relevant for dynamic start condition to ensure stimulus information can be acquired during upwards movement
- **Starting procedure**
  - Restricting **maximum initiation time** led to larger effects – a **dynamic start** only influenced shape but not effect size
  - However, restricting initiation times also led to largest proportion of excluded trials (and seemed to be challenging for some participants)

Starting procedure: Static vs. dynamic start

Method (Scherbaum & Kieslich, in press)

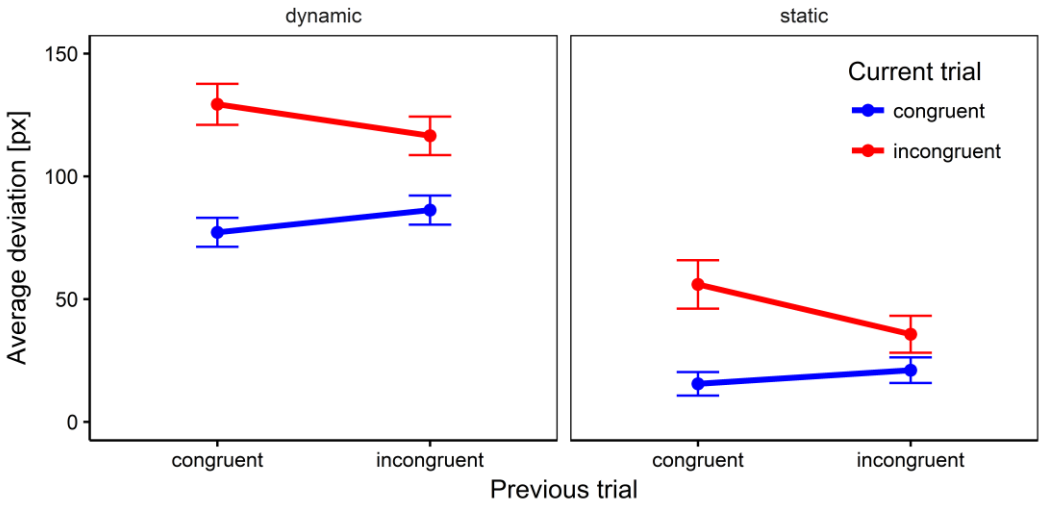
- Mouse-tracking in Simon task
  - Participants **click on left vs. right** option depending on stimulus (left if number < 5, otherwise right)
  - Position of stimulus varied (left vs. right) so that desired response and position are either **congruent** or **incongruent**
- Variation starting procedure
  - **Dynamic**: move upwards to display stimulus (data from Scherbaum et al., 2010)
  - **Static**: stimulus displayed after fixed interval of 200 ms (typical duration of movement initiation in dynamic condition) (new data)



Starting procedure: Static vs. dynamic start

Discrete effects: Results for average deviation

- Simon effect and congruency sequence effect replicated in both conditions
- No significant interaction of theoretically important effects with starting procedure

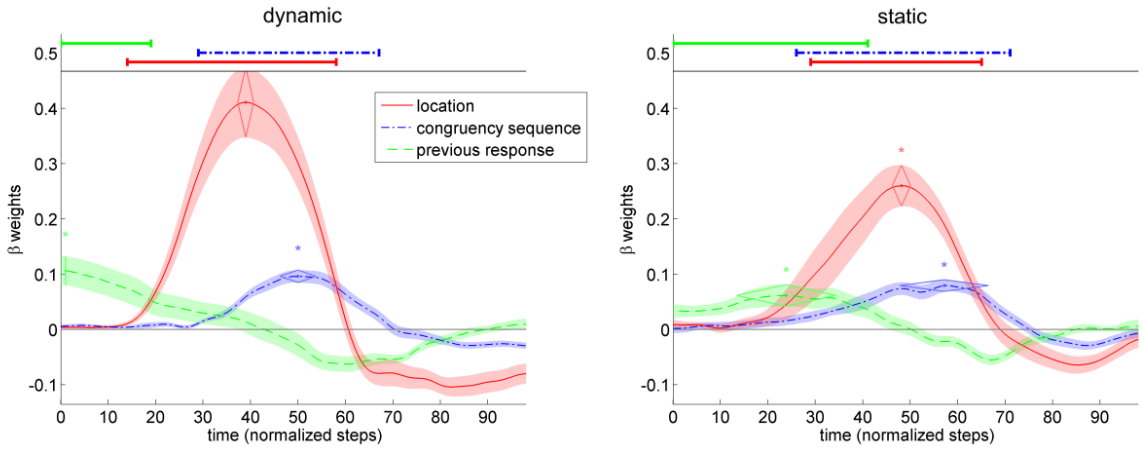


Error bars represent 1 *SEM*.

Starting procedure: Static vs. dynamic start

Dynamic effects: Time-continuous angle regression

- Time continuous multiple regression predicting vertical movement angle at each time point
- Predictors
  - location (congruency)
  - congruency sequence (same / different)
  - previous response (same / different)
- Effects stronger and more temporarily distinct in dynamic starting condition

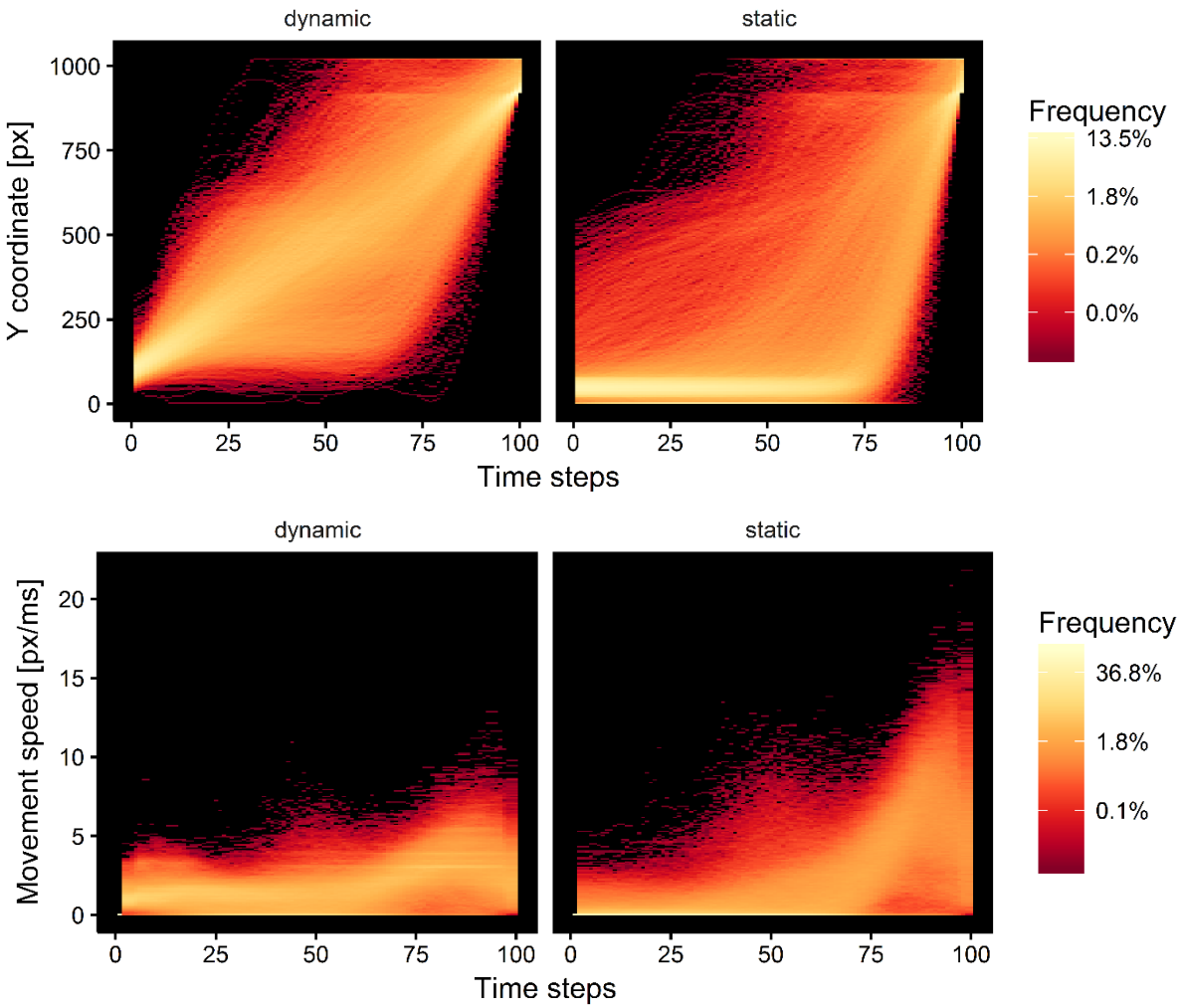


Average β weights per time step and predictor.
Lines indicate segments of β weights significantly > 0.

Starting procedure: Static vs. dynamic start

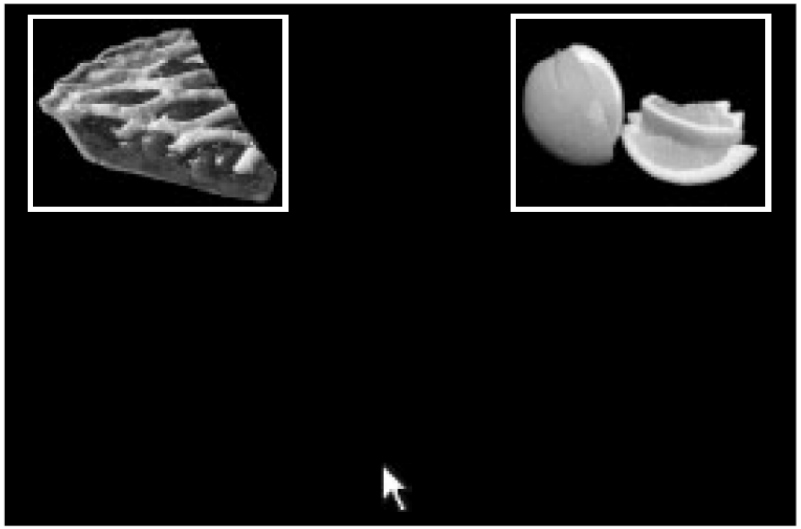
Movement consistency

- Smooth and consistent upwards movement in dynamic starting condition
- Participants in static starting condition often stay at bottom of screen for more than half of the trial before moving upwards quickly

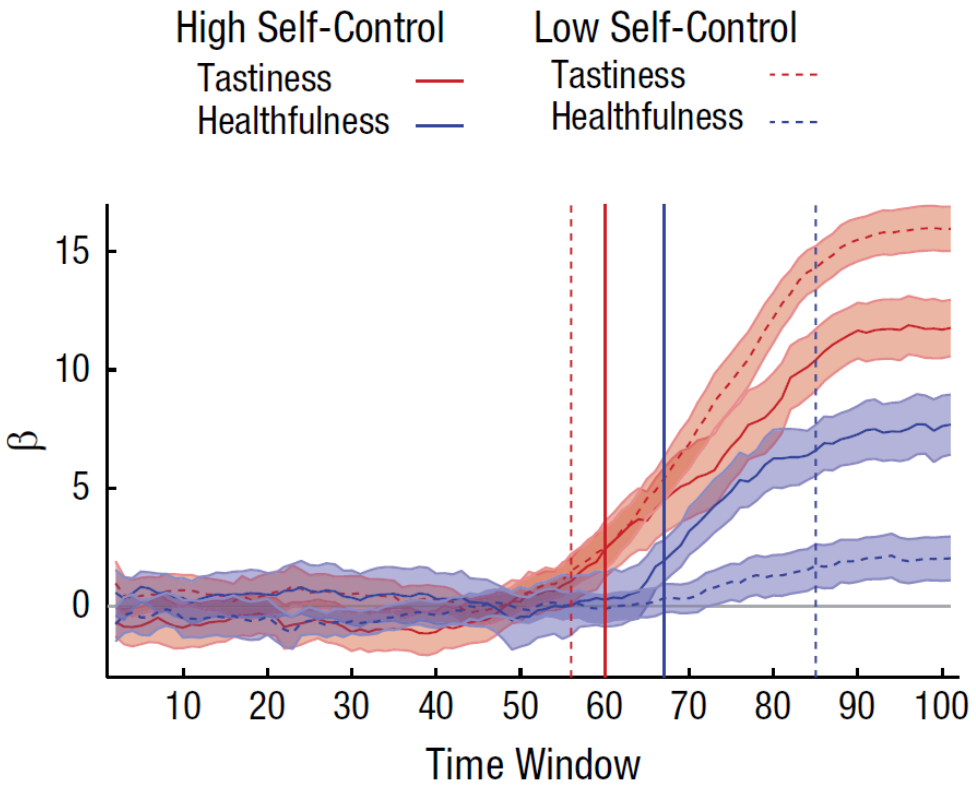


Self-control

Food choices (Sullivan et al., 2015)



Choice Screen



Decisions under risk

Basic paradigm

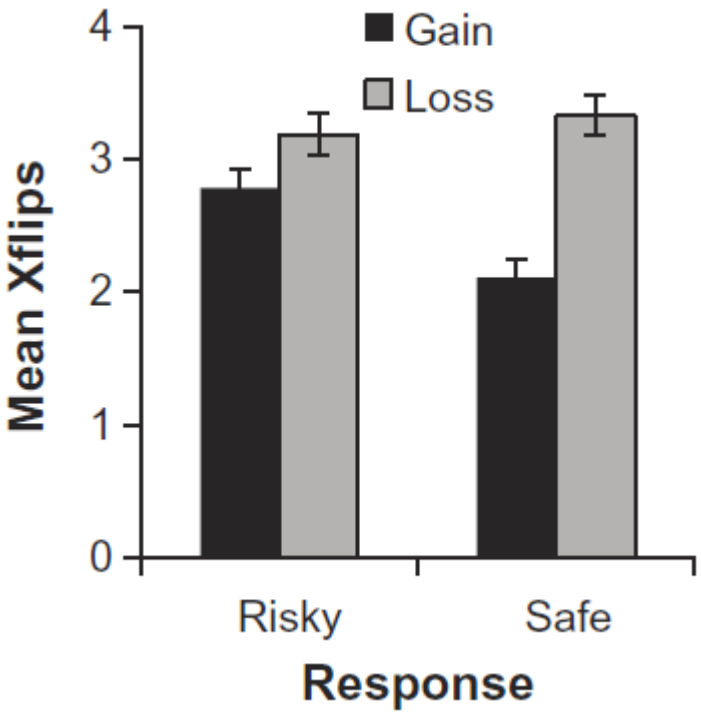
- Risky choice / decisions under risk
  - Which of the two gambles do you want to play?

Gamble A	Gamble B
You have a 50% chance of winning $90, otherwise nothing	You have a 90% chance of winning $50, otherwise nothing

- Gamble A: “risky”
  - Higher amount, lower probability of winning
- Gamble B: “safe”
  - Lower amount, higher probability of winning

Decisions under risk

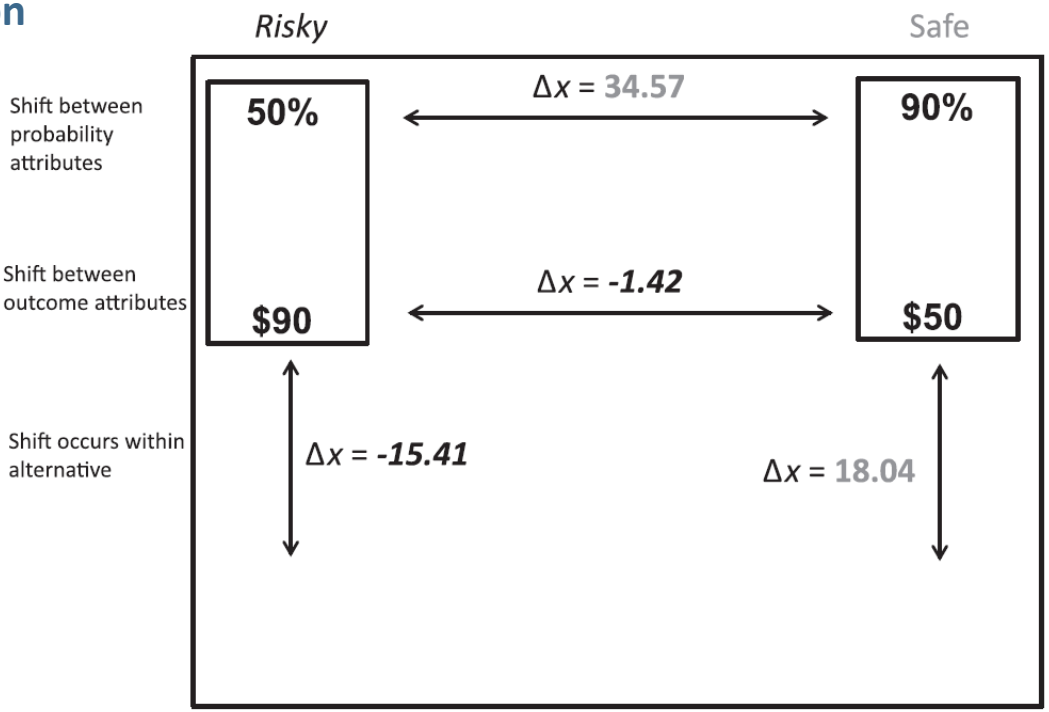
x-flips (Koop & Johnson, 2013, Exp. 2)



Decisions under risk

Combining mouse- and eye-tracking (Koop & Johnson, 2013, Exp. 3)

- Change in x-position (Δx) as function of **transitions of attention**
  - $\Delta x > 0$: movement towards safe gamble
  - $\Delta x < 0$: movement towards risky gamble
- **Evidence accumulation model**
  - Predict **momentary preference** based on **visual input**
  - Mean correlation between predicted preference and x-position is $r = .78$
- Conclusions
  - **Visual attention** to probability and outcome information **predicts mouse response**
  - Mouse movements largely reflect quality of acquired information



Your experiments

Mouse-tracking introduction (Monday)

- 13:00-14:30 General introduction to mouse-tracking
  - Paradigm and assumptions
  - Implementation and analysis
  - Previous applications
- 14:30-15:00 Introduction to task
  - Type of experiments considered
  - Your tasks during the workshop
- 15:00-17:00 Develop experimental design conceptually
- 17:00-18:00 Present experimental design in plenum

Your tasks during the workshop

- Goal of workshop
  - Design, build, pre-register, run, and analyze a mouse-tracking experiment
  - In small groups
- Monday
  - Develop experimental design (task, manipulation, hypotheses, measures)
  - Present experimental design in plenum
- Tuesday
  - Build experiment
  - Register experiment at OSF
  - Participate in experiments
- Wednesday
  - Analyze and visualize your data
  - Discuss your results
- Saturday
  - Present results

Type of experiments

- In the experiment, participants complete a number of trials that involve decisions of the same structure
- In each trial, participants have to decide between two options by clicking on the corresponding button (two-alternative forced choice task, 2AFC)
- Between trials, the stimulus to be decided upon varies (usually) and / or the two response categories
- The stimulus (and/or the response options - in case they vary) should be simple (e.g., a single word, a picture)

Implementation & analysis
Software

- **Custom extensions** for experimental software
  - Code based implementations, e.g., in E-Prime or MATLAB
  - Also need scripts for preprocessing the data
  - Require **programming** skills

- **MouseTracker** (Freeman & Ambady, 2010)
  - **Stand-alone** program
  - Relatively easy to use, but limited in features and flexibility
  - Free of charge but closed source, Windows only

- **Mousetrap** (Kieslich & Henninger, 2017; Kieslich, Wulff et al., in preparation)
  - Drag & drop plugins for experimental software **OpenSesame**
  - **R package mousetrap** for preprocessing and analysis
  - Open source, free of charge, cross-platform
  - Available from http://pascalkieslich.github.io/mousetrap/

Software for the workshop

- To create mouse-tracking experiments, first install OpenSesame. It is available from http://osdoc.cogsci.nl/3.2/download/.
- To install the mousetrap plugin for OpenSesame, follow the instructions at https://github.com/pascalkieslich/mousetrap-os#installation. Please make sure to install the latest version of OpenSesame (3.2.4) and the development version of the mousetrap-os plugin.

- To analyze mouse-tracking data install R (https://www.r-project.org/) and RStudio (https://www.rstudio.com/products/rstudio/download/).
- Afterwards, please run the following command in R to install the required packages:

```
install.packages(c("readbulk","mousetrap"))
```

- Screen resolution of experiment for lab computers: 1280 x 1024 px

Thank you!

Questions and comments are highly appreciated!

Now & via email: kieslich@psychologie.uni-mannheim.de
dirk.wulff@gmail.com

Mousetrap-os plugins: https://github.com/pascalkieslich/mousetrap-os

Mousetrap R package: http://pascalkieslich.github.io/mousetrap/

Thanks:

Felix Henninger, co-developer of mousetrap-os plugin and R package

Jonas Haslbeck & Michael Schulte-Mecklenbeck, co-developers of mousetrap R package

Mila Rüdiger and Monika Wiegelmann for data collection and testing

DAY 2:
CREATING MOUSE-TRACKING EXPERIMENTS

Pascal Kieslich (University of Mannheim)

Workshop at the EADM Summer School 2018 in Salzburg, Austria

Creating mouse-tracking experiments (Tuesday)

- 09:00-11:00 OpenSesame & mousetrap-os introduction
- 11:00-12:00 Build experiment
- 12:00-13:00 Lunch break
- 13:00-15:00 Build experiment
- 15:00-16:00 Register experiment at OSF (Michael)
- 16:00-17:00 Keynote Neil Stewart
- 17:00-18:00 Meet the Scientist
- 18:00-19:00 Participate in experiments (Lab computers)

Software for the workshop

- To create mouse-tracking experiments, first install OpenSesame. It is available from http://osdoc.cogsci.nl/3.2/download/.
- To install the mousetrap plugin for OpenSesame, follow the instructions at https://github.com/pascalkieslich/mousetrap-os#installation. Please make sure to install the latest version of OpenSesame (3.2.4) and the development version of the mousetrap-os plugin.

- To analyze mouse-tracking data install R (https://www.r-project.org/) and RStudio (https://www.rstudio.com/products/rstudio/download/).
- Afterwards, please run the following command in R to install the required packages:

```
install.packages(c("readbulk","mousetrap"))
```

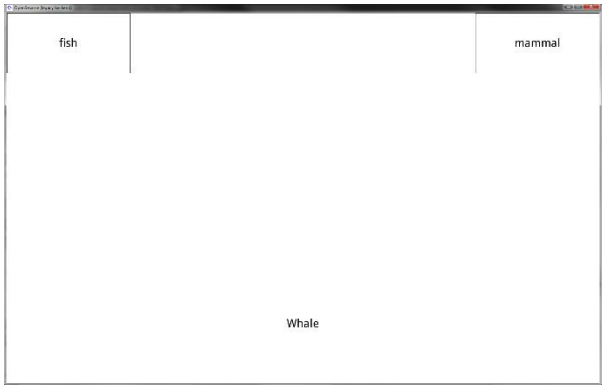
- Screen resolution of experiment for lab computers: 1280 x 1024 px

OpenSesame & mousetrap-os introduction

Creating mouse-tracking experiments

Replication study of Dale, Kehoe, & Spivey (2007)

- **Animal categorization task**
  - **Typical exemplars** only share features with correct category (e.g., cat as mammal)
  - **Atypical exemplars** share both features with correct and competing category (e.g., whale with mammal and fish)



- **Main hypothesis**
  - **Increased competition** when categorizing atypical exemplars
    - Mouse trajectories with deviation towards competing category


OpenSesame (legacy backend)
mammal
fish
Whale

OpenSesame
Overview

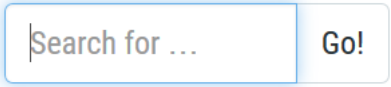
- Graphical experiment builder
  - Developed by Sebastiaan Mathôt (Mathôt, Schreij, & Theeuwes, 2012)
  - Create experiments by **drag & drop** via GUI
  - Implement complex tasks using **Python** scripts

- Open source & cross platform
  - Download and documentation: http://osdoc.cogsci.nl/
  - Available for Windows, Linux, Mac OS

- Allows for extensions via plugins
  - **PyGaze** plugin for eye-tracking (Dalmaijer, Mathôt, & Stigchel, 2014)
  - **Psynteract** plugin for interactive experiments (Henninger, Kieslich, & Hilbig, 2017)
  - **Mousetrap** plugin for mouse-tracking (Kieslich & Henninger, 2017)

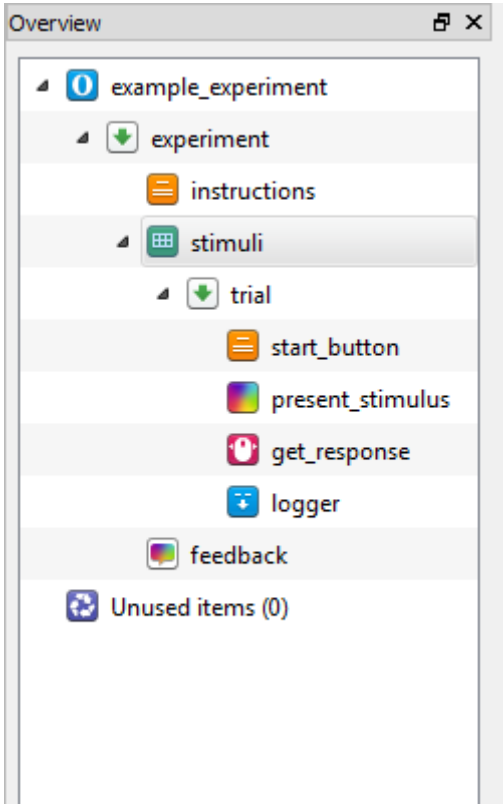
OpenSesame
Resources

- Documentation
  - http://osdoc.cogsci.nl
  - Extensive & comprehensible
- Forum
  - http://forum.cogsci.nl
  - Very supportive & fast responses
- Source code
  - https://github.com/smathot/OpenSesame
- Custom search
  - http://osdoc.cogsci.nl
  - searches documentation & forum



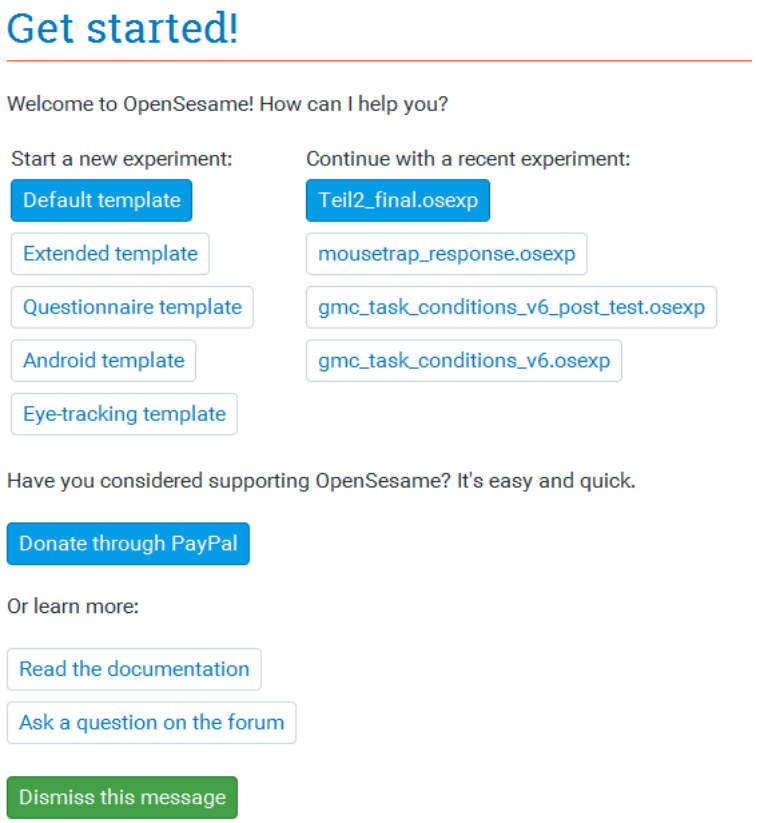
OpenSesame
Items

- **Items** as the building blocks of an experiment
- **Different types** of items serve basic purposes
  - Presentation of stimuli
  - Collection of responses
  - Logging of responses
  - …
- Organized in **hierarchical + sequential** structure
  - **Sequence** runs multiple items in succession
  - **Loop** repeats sequence of items multiple times (with variations)



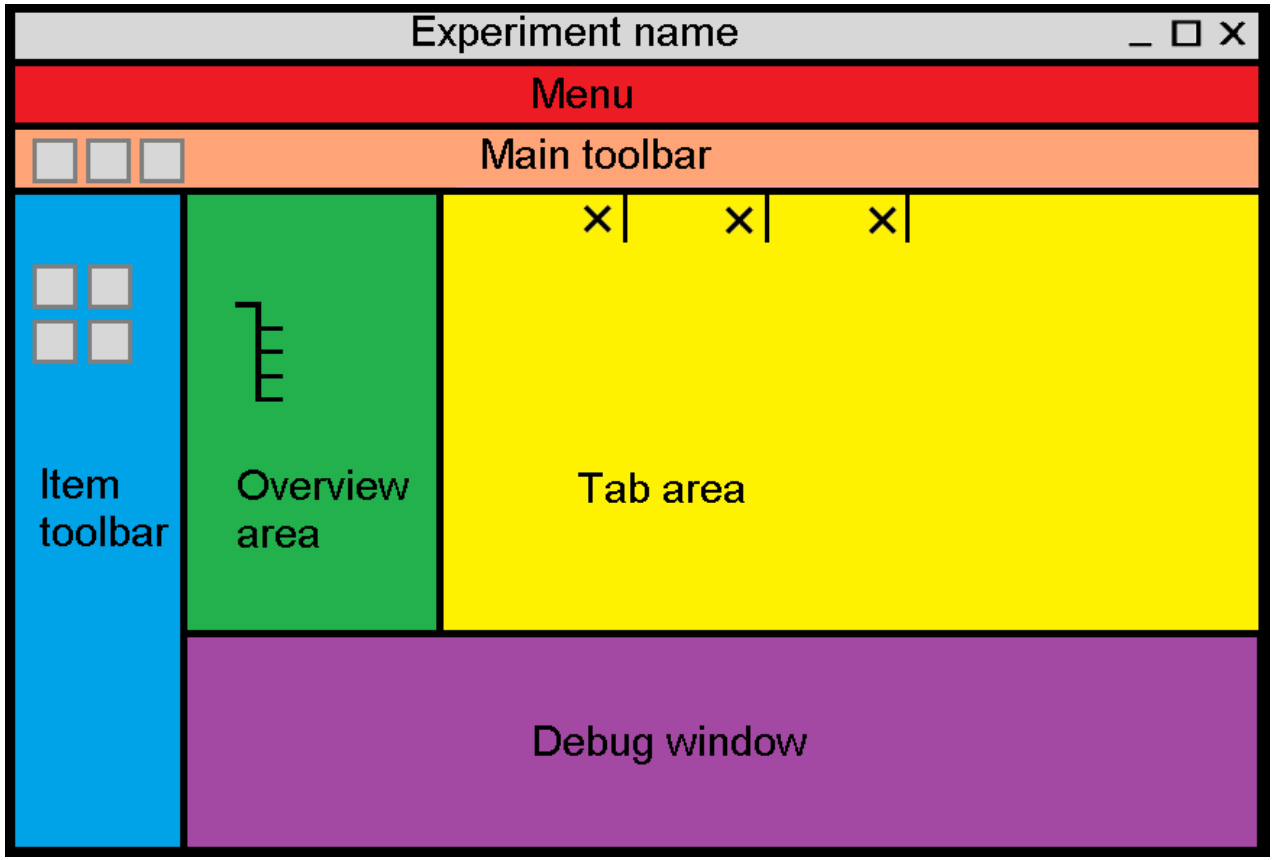
OpenSesame
Getting started

- **New**: List of available templates
  - **Default**: create an experiment from scratch
  - **Extended**: already includes a basic experimental structure
- **Recent**: List of recently opened experiments



OpenSesame

Graphical user interface (GUI)

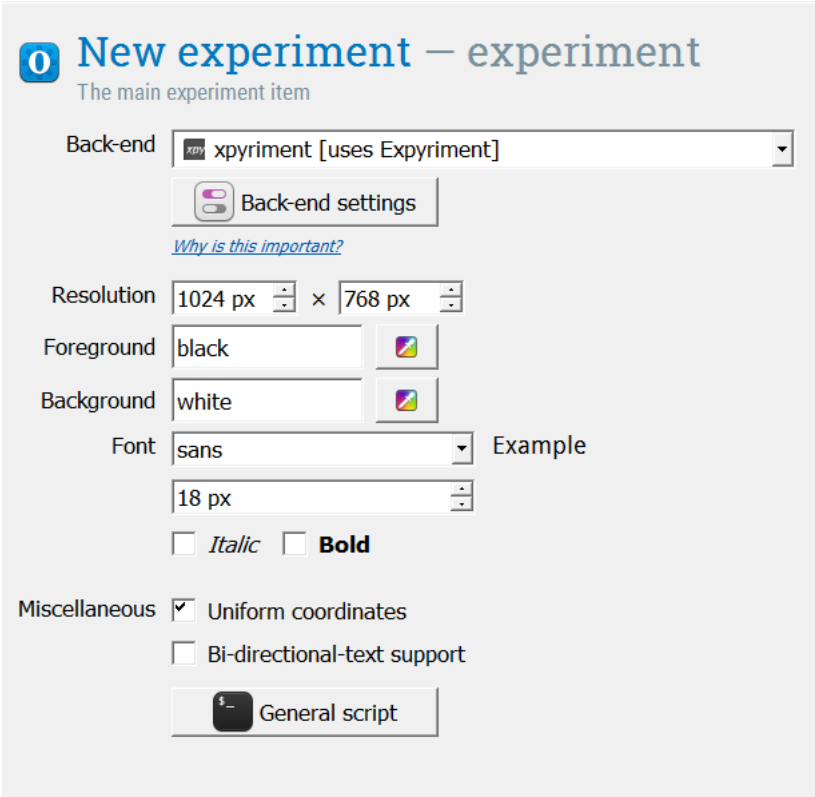


- http://osdoc.cogsci.nl/manual/interface/

OpenSesame

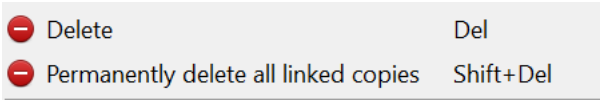
General experiment properties

- Set **general properties** for the experiment
  - Set the **experiment backend** to legacy or expyriment
  - Change the **name** of the experiment
  - **Foreground** color (= default font color) → black
  - **Background** → white
  - **Font** family → sans
  - **Resolution** → adjust to display resolution (mouse-tracking experiments are typically run full-screen) → **use: 1280 x 1024 px** (to match lab computers)

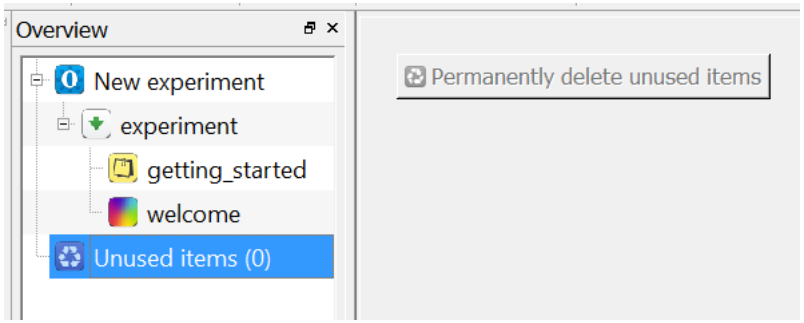


OpenSesame
Deleting items

- Delete items
  - By right clicking on them and selecting the corresponding option
  - By pressing delete (“Del”/“Entf”)



- **Task:** Delete starting items

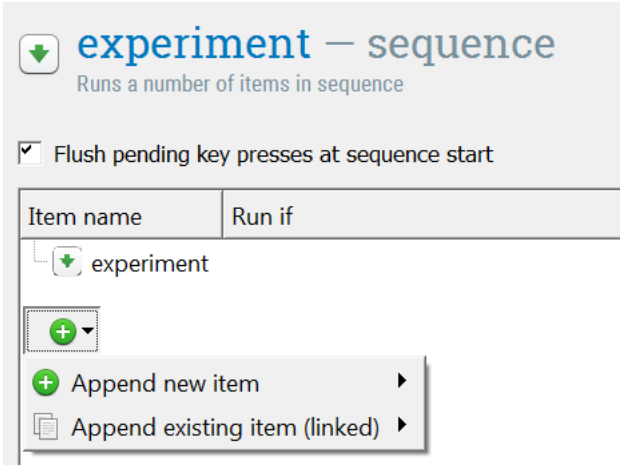


- Deleted items still available & might still affect experiment
  - Can be re-added (by dragging them back to the experiment)
  - Permanently delete them by selecting corresponding option in unused items tab → usually do this

OpenSesame
Adding new items

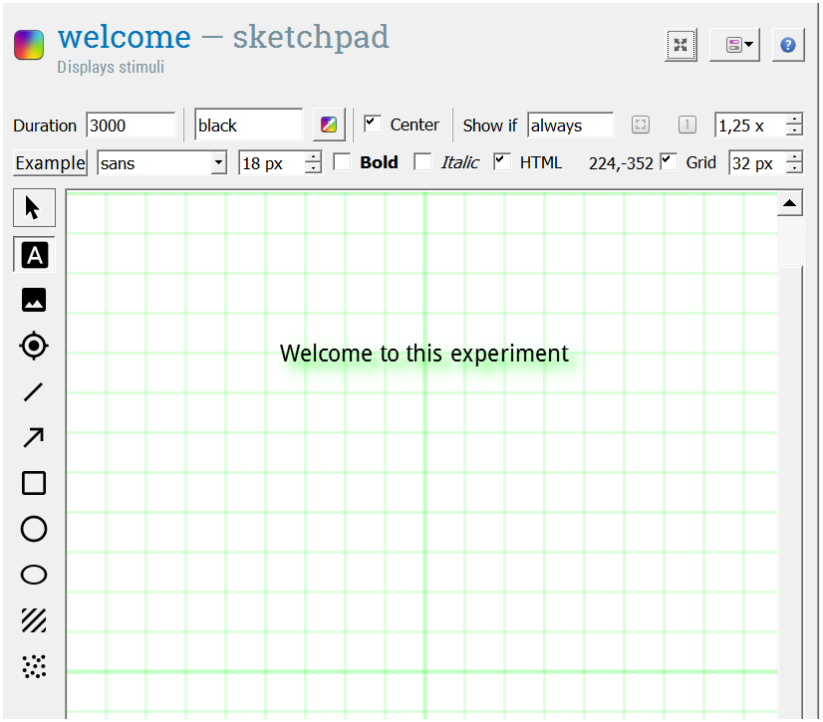
- Two ways to add items
  - Drag them from the item toolbar onto overview area
  - Click on the relevant sequence and select append new item (or existing item)

- **Task**
  - Create Welcome screen (sketchpad)
  - Should contain welcome message
  - Rename it *welcome*
  - Should be displayed for 3000 ms



OpenSesame
Sketchpads

- Used to present strings of text, geometric shapes or bitmap images
- Provides simple built-in drawing tools
- Uses a coordinate system (center coordinate 0,0 and pixel metric)



- http://osdoc.cogsci.nl/manual/stimuli/visual/

OpenSesame

Running your experiment

- Run fullscreen
- Run in window
- Test run (run in window without specifying subject number & file location)

- Abort running experiment: press escape key

- After experiment is finished
  - Log file can additionally be saved within the experiments file pool
  - Usually we don't want this, especially not for test runs

- Task
  - Give your experiment a test run

OpenSesame

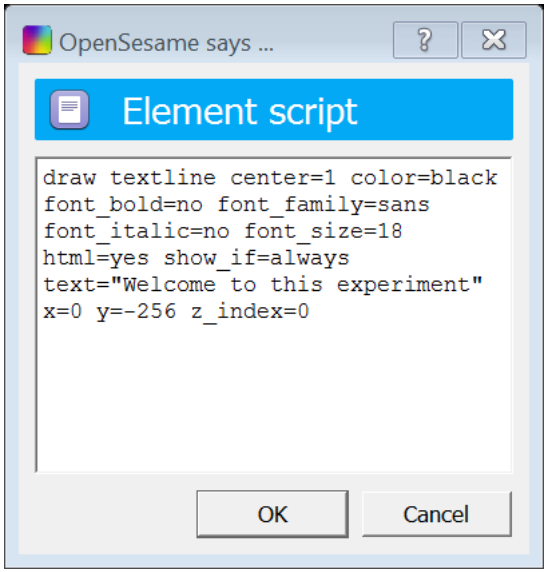
Sketchpads: Drawing tools

- Select and modify (modify & move existing elements around)
- Text element (line breaks are possible)
- Image element
- Fixation dot element
- Line element
- Arrow element
- Rectangle element
- Circle element
- Ellipse element
- Gabor patch element
- Noise patch element

OpenSesame

Sketchpad: Modifying elements on sketchpad

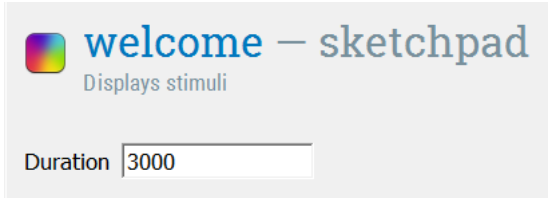
- Select modify
- Move elements: left click on them and drag them
- Modify elements: different options
  - Left click on them and change the options above
  - Double left click on them (especially to edit text)
  - Right click on them, select “edit script” and modify the underlying OpenSesame script syntax
- Delete elements: left click on them and press delete (“Entf”)
- Task
  - Change appearance of welcome message: it should be displayed in **blue and boldface**

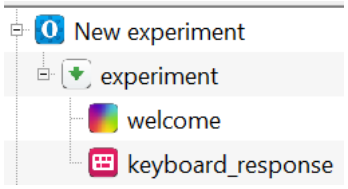


OpenSesame

Display duration & input of sketchpad/feedbacks

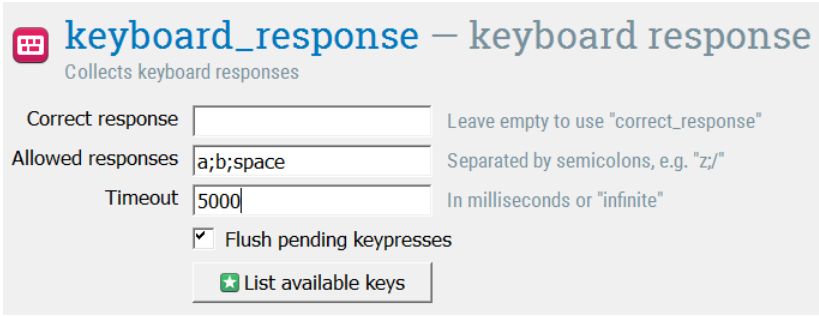
- Values for duration
  - Any positive integer: display duration in ms
  - *keypress*: display until any key is pressed
  - *mouseclick*: display until any mouse button is clicked
- For more specific keypress/ mouseclick conditions
  - Set duration to 0
  - Use keyboard_response/ mouse_response items





OpenSesame
Keyboard response

- Separate different keys by **;**
- **Flush** pending keypresses
  - always a good idea
  - otherwise unprocessed keypresses from previous stage of experiment may exert an unwanted influence
- **Special** key names
  - See list available keys
- Task
  - Set duration of sketchpad to 0
  - Participants can press space bar to continue (use keyboard_response item)

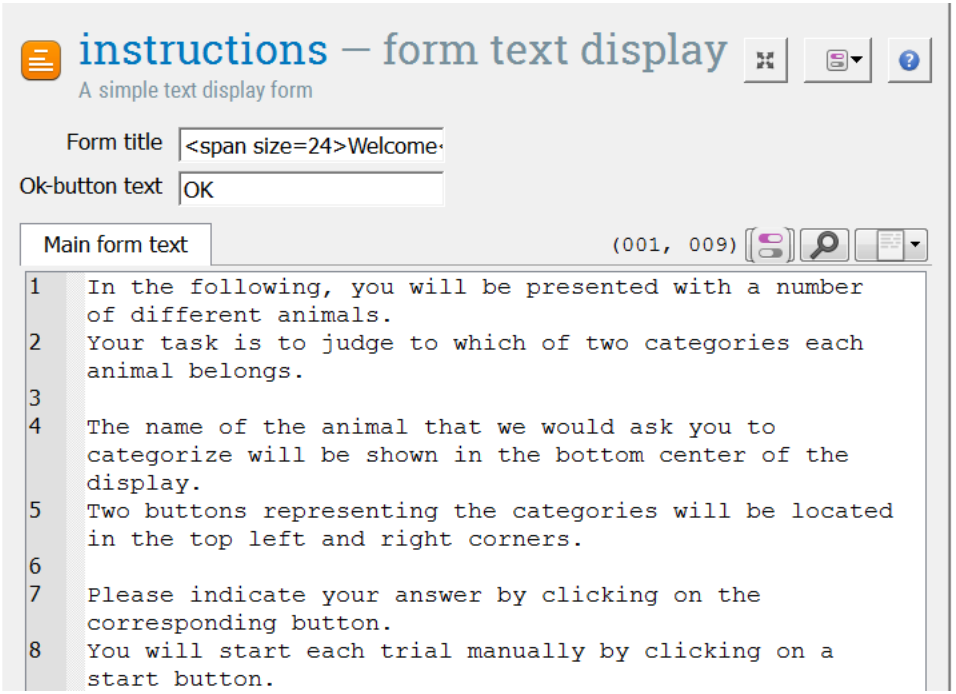


- http://osdoc.cogsci.nl/manual/response/keyboard/

Creating mouse-tracking experiments

Instruction screen

- Add **form_text_display** item to provide basic **instructions**
- Save experiment
- Give it a **test run**



Creating mouse-tracking experiments
Mousetrap plugin for OpenSesame

- Mousetrap plugin for OpenSesame
  - Enables users to implement mouse-tracking via **graphical user interface**
  - Also provides **Python classes** for mouse-tracking in Python code

- Installation (of current development version)
  - See https://github.com/pascalkieslich/mousetrap-os#installation
  - Execute following commands in OpenSesame's debug window

```
import pip
pip.main(['install', 'https://github.com/PascalKieslich/mousetrap-os/archive/master.zip'])
```

  - OpenSesame needs to be run in admin mode for this (under Windows)
  - Restart OpenSesame afterwards

Creating mouse-tracking experiments
Mousetrap plugin for OpenSesame

- Resources
  - Documentation and example experiments: https://github.com/pascalkieslich/mousetrap-os
  - Article (including tutorial and validation): Kieslich, P. J., & Henninger, F. (2017). Mousetrap: An integrated, open-source mouse-tracking package. *Behavior Research Methods, 49*(5), 1652-1667. https://doi.org/10.3758/s13428-017-0900-z

- Questions and updates
  - Forum for questions: http://forum.cogsci.nl/index.php?p=/categories/mousetrap
  - Mailing list for updates: http://eepurl.com/co1AqX

Creating mouse-tracking experiments

Mousetrap plugin for OpenSesame

- Two different options for implementing mouse-tracking

mousetrap_response item

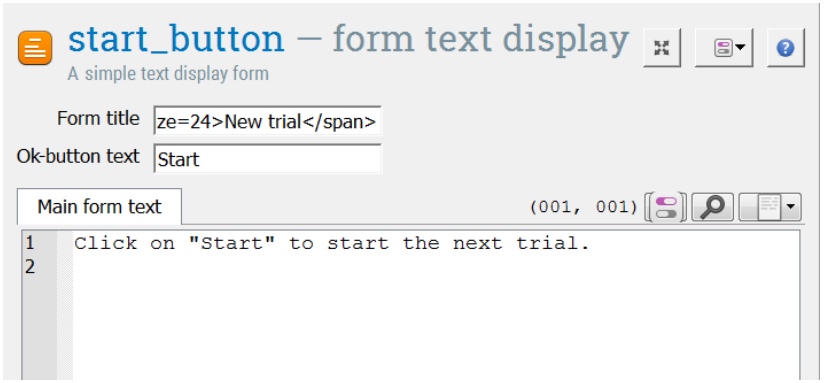
- Tracks mouse movements while stimulus display is provided by another item
- Can use graphical user interface for designing the stimulus display

mousetrap_form item

- Provides both stimulus display and mouse-tracking
- Stimulus designed using simple OpenSesame script syntax

Creating mouse-tracking experiments
Start button

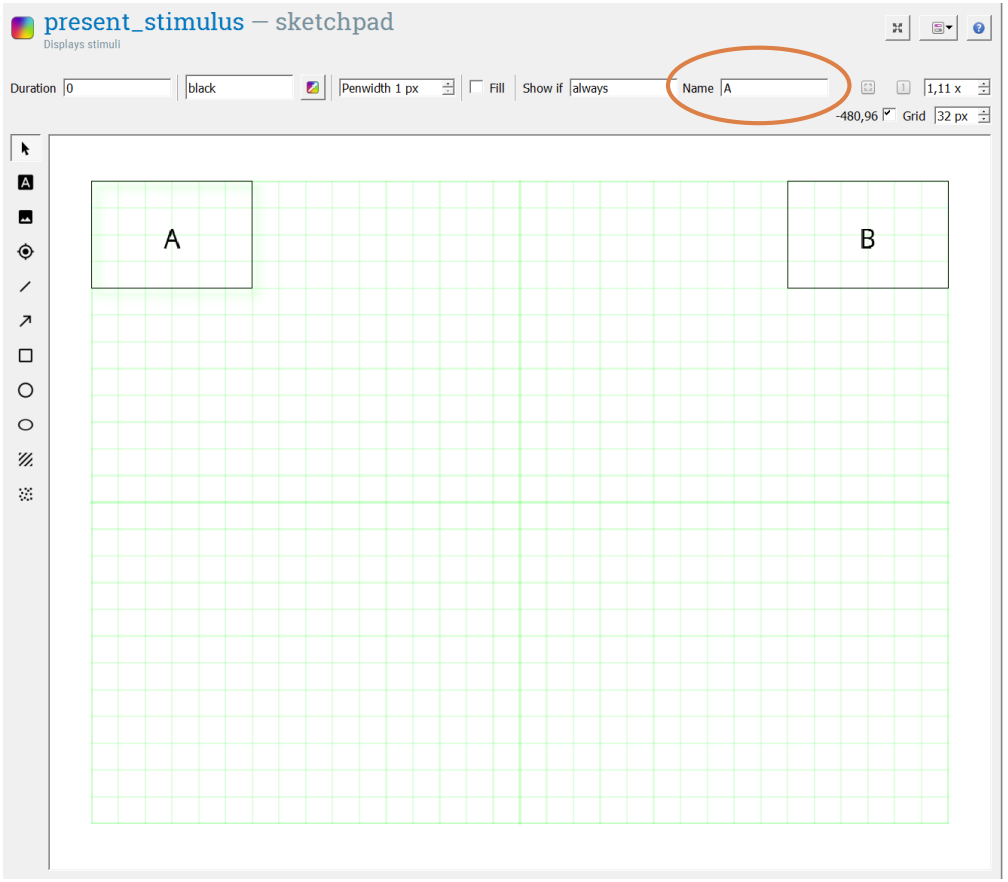
- Add **form_text_display** item that contains the start button
- Participants start stimulus presentation by clicking on this button
- This ensures that **start position** of cursor is **comparable** across trials
- This start position usually is in the **bottom center** of the screen



Creating mouse-tracking experiments

Stimulus display

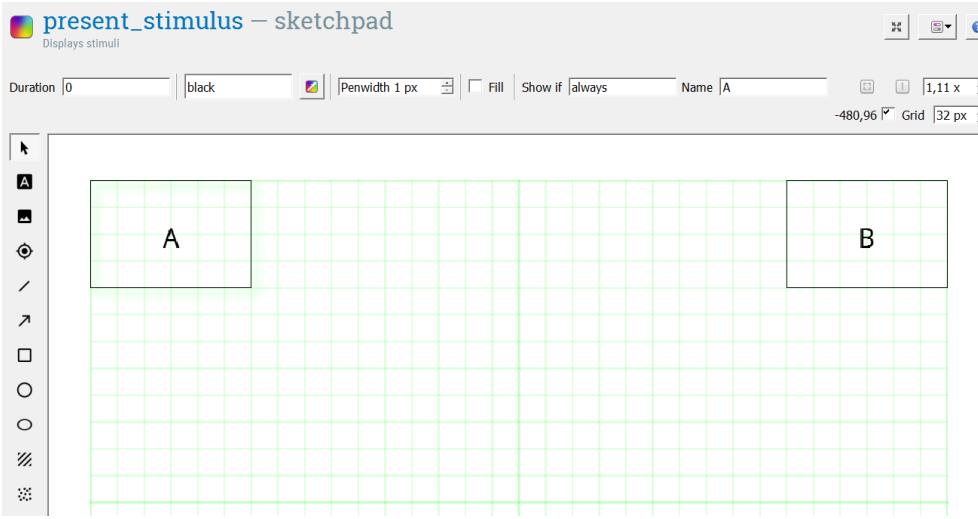
- Use a **sketchpad** item to create the stimulus display
- Create two **buttons**
  - Draw their borders using **rect elements** and give them a label via the **Name field**
  - Add the button text using **textline elements**
- Layout considerations
  - Place buttons in **screen corners** to avoid overshooting
  - Ensure **symmetric layout** and that buttons have same distance from start position (bottom center)

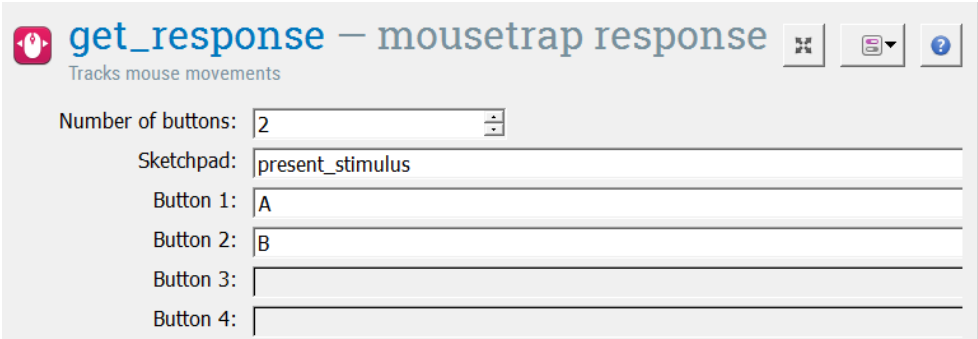


Creating mouse-tracking experiments

Track mouse movements

- Set duration of sketchpad to 0
- Add a **mousetrap_response** item to track mouse movements and specify **number** of buttons, their **sketchpad** and **name**
- As name we enter the **label text** – this will be saved when button is clicked
- Run experiment and check data using **variable inspector**





Creating mouse-tracking experiments

Trial structure

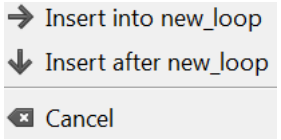
- Loops and sequences are used to structure the experiment

Sequences

- List of items that is executed sequentially
- Does not repeat automatically – need to combine it with a loop

Loops

- Repeatedly calls a single other item – usually a sequence (drag new sequence item on loop or select existing sequence)



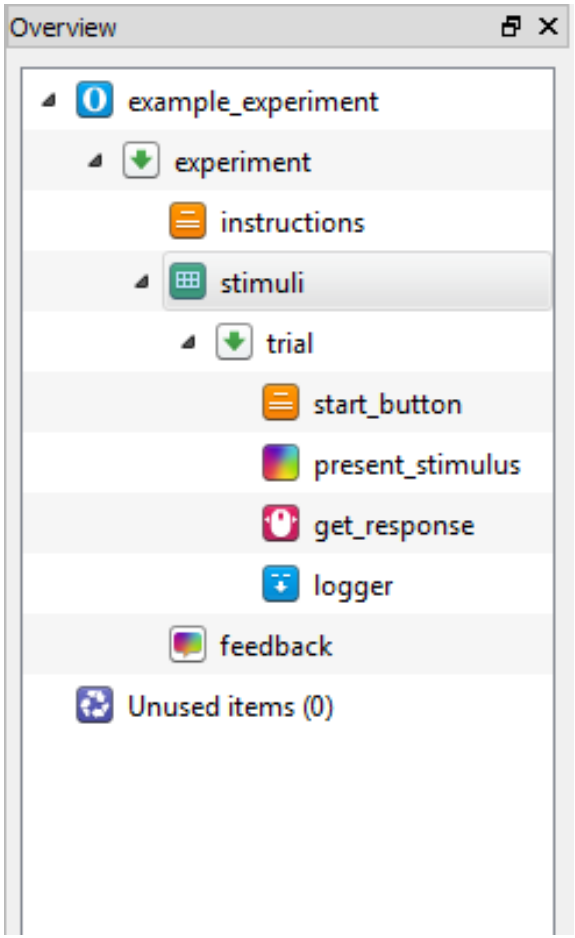
- In the loop item, variables can be stored that vary for each trial
- Thereby, independent variables can be varied and/or experimental material can be stored for the experiment

- http://osdoc.cogsci.nl/manual/structure/loop/

Creating mouse-tracking experiments

Trial structure

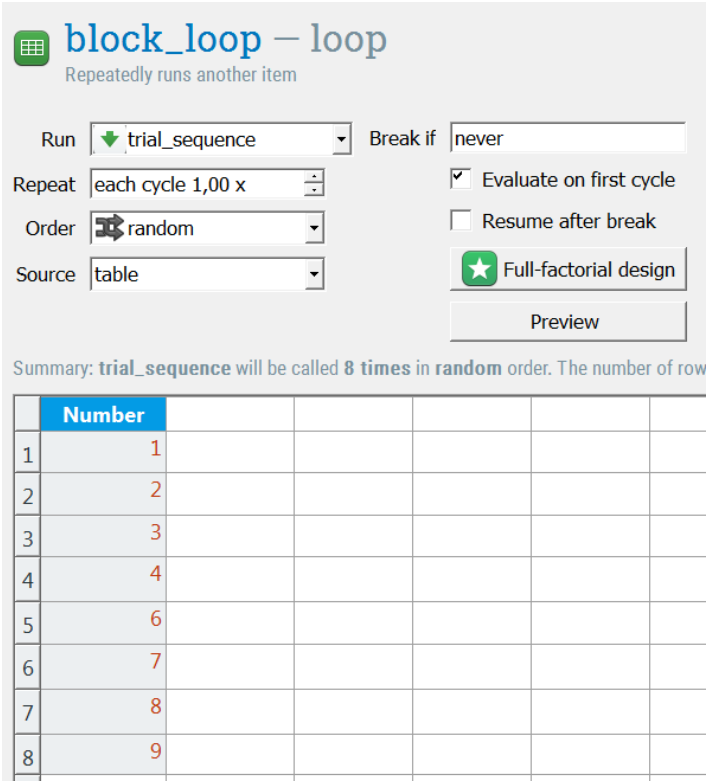
- Implement trial structure by including a **stimuli loop** and a **trial sequence**
- Include all items except the instructions in the trial sequence as they should be presented in each trial
- Add a **logger item** at the end of the trial sequence which writes the recorded data to the logfile



Creating mouse-tracking experiments

Loops

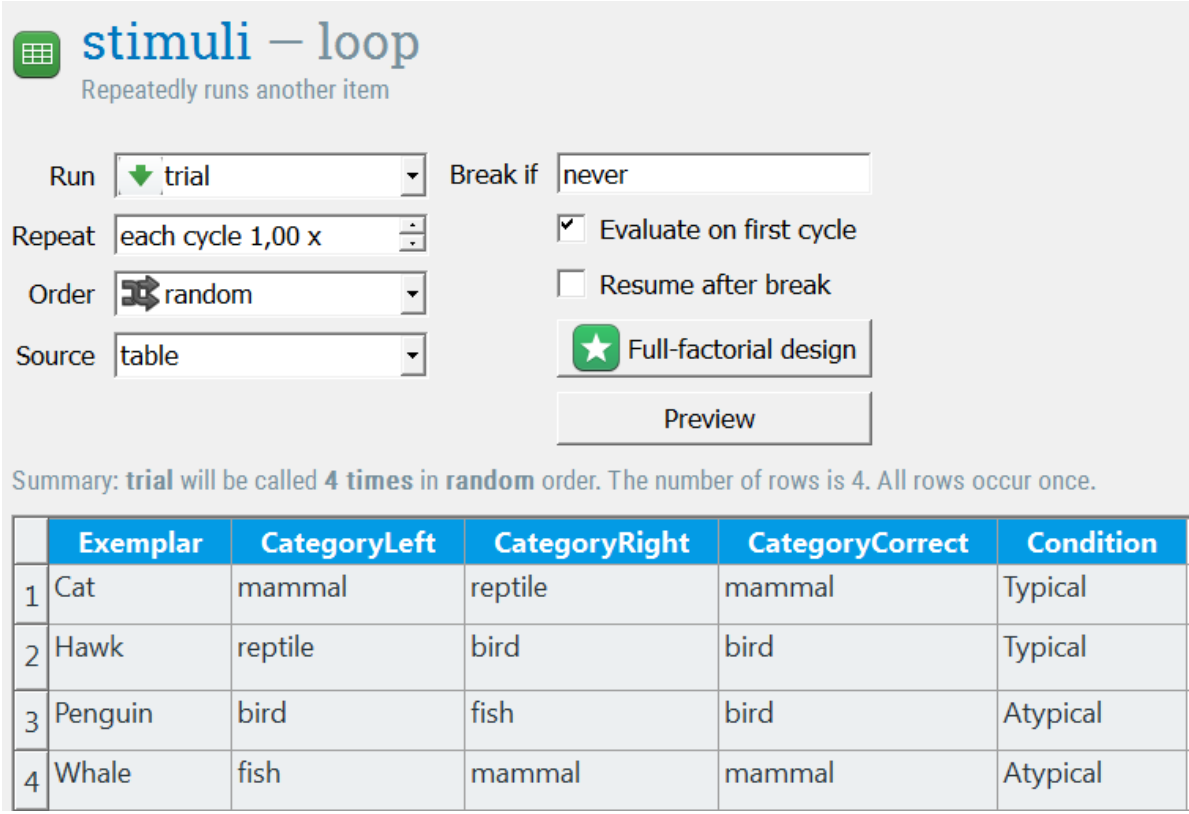
- Loops can be used to store material or to vary independent variables
  - **Variables** stored in columns
  - Distinct **trials/conditions** stored in rows (= cycles)
- **Repeat**: number of repetitions of each cycle
  - Total repetitions = cycles x repeat
  - If repeat is < 1, a randomly selected subset of trials is chosen (e.g., 50 % of trials for 0.5)
- **Order** refers to the order with which all trials are presented
  - **Random** = every trial is randomly drawn without replacement (ignoring cycles x repeat structure)
  - **Sequential** = every trial in sequential order



- http://osdoc.cogsci.nl/manual/structure/loop/

Creating mouse-tracking experiments
Material loop

- Design options of the loop are displayed at the top
- Material can be added in the bottom
  - each **column** corresponds to one experimental **variable**
  - each **row** corresponds to one unique **trial**
- Create the loop as displayed on the right



	Exemplar	CategoryLeft	CategoryRight	CategoryCorrect	Condition
1	Cat	mammal	reptile	mammal	Typical
2	Hawk	reptile	bird	bird	Typical
3	Penguin	bird	fish	bird	Atypical
4	Whale	fish	mammal	mammal	Atypical

Creating mouse-tracking experiments

Choosing variable names

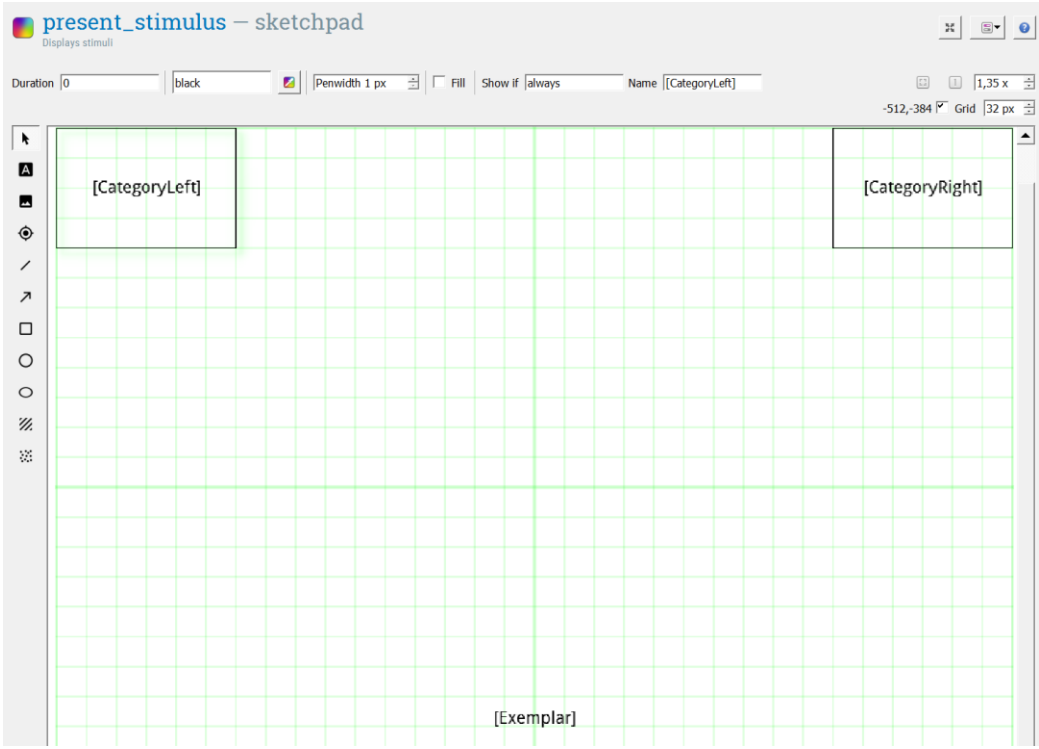
- OpenSesame (and Python) is **case sensitive**
  - "Condition" and "condition" are different variables
- Variable names **may not contain spaces**
  - use "_" instead, e.g., "my_variable"
- Variables may **not contain . or ,**

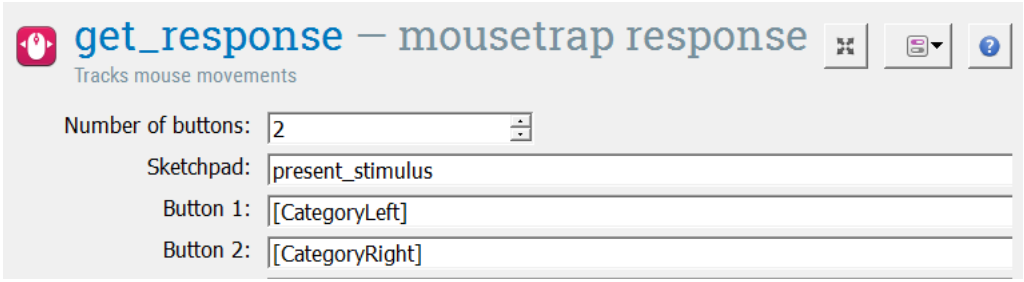
- http://osdoc.cogsci.nl/manual/variables/

Creating mouse-tracking experiments

Using variables

- Variables that are defined in a loop or recorded through an item can be used by including their name in **square brackets**
- **Adjust the button labels** in sketchpad and mousetrap_response item so that they vary in each trial
- Present **exemplar** name in bottom center of screen
- Give the experiment a test run





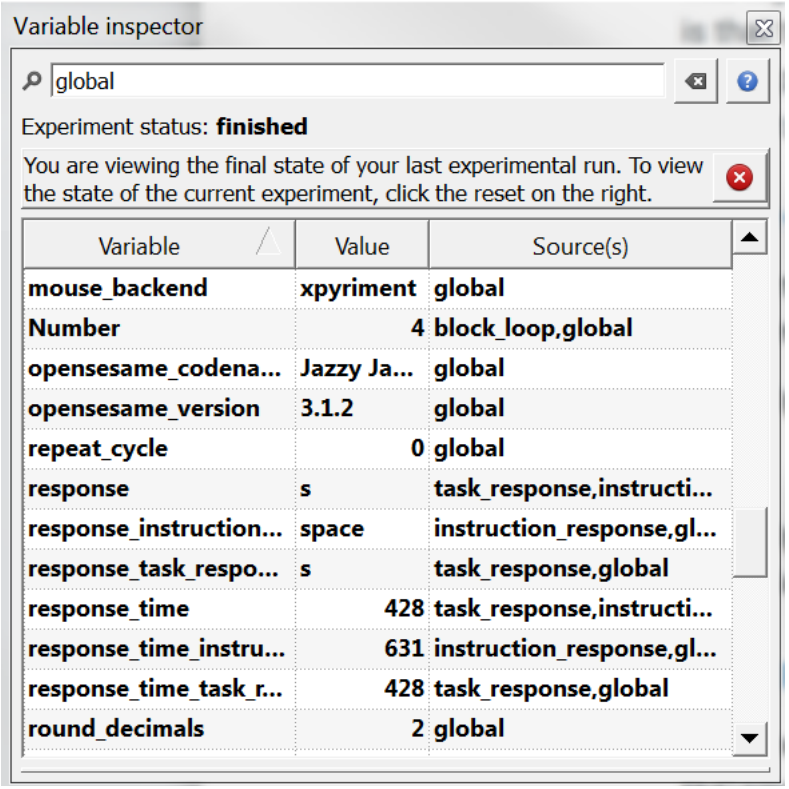
Creating mouse-tracking experiments
Built-in experimental variables (excerpt)

- **Experiment variables**
  - subject_nr (1, 2, …) & subject_parity (“odd” or “even”)
- **Item variables**
  - count_**item_name**: number of times – 1 **item** has been called (starting at 0)
  - time_**item_name**: timestamp of last time item was executed
- **Response variables**
  - response & response_**item_name** → last response overall and to specific **item**
  - response_time & response_time_**item_name** → response time in ms
  - correct & correct_**item_name** → =1 if response == correct_response, 0 otherwise
- **Feedback variables**
  - average_response_time/avg_rt: average response time
  - accuracy /acc: average percentage of correct responses
  - can be reset, e.g., are reset by default when using feedback item → take care!

- http://osdoc.cogsci.nl/manual/variables/#built-in-variables

Creating mouse-tracking experiments
Variable inspector

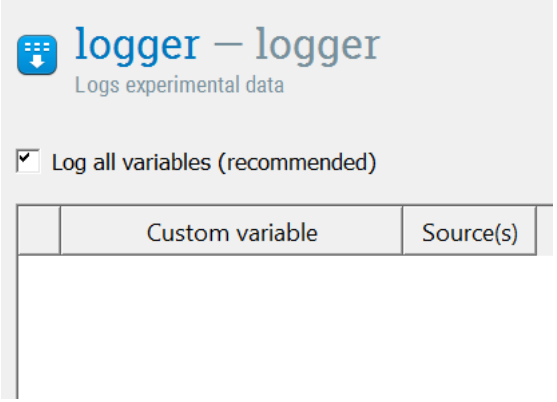
- Overview of experimental variables: **variable inspector**
  - User defined variables (e.g., via loop or inline_scripts)
  - Built-in variables
- Displays the current status of all experimental variables
  - Can be used to monitor changes while experiment is running
  - For this the Runner (specified in Tools/Preferences) cannot be set to opensesamerun



- http://osdoc.cogsci.nl/manual/variables/#the-variable-inspector

Creating mouse-tracking experiments
Response collection & logging

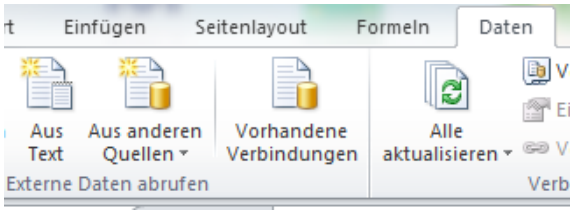
- Variables only saved in the log file **if there is a logger item!**
  - Every time a logger item is passed, it writes the current status of all experimental variables to the log file
  - Always check that you have a logger item included in your trial
  - Always use the **same logger** by using append existing item / linked copy
  - Always check your log file
- **Default** (log all variables) makes sense
  - In case a variable is not logged, add it as a custom variable but inspect experiment closely, as there might be a problem somewhere

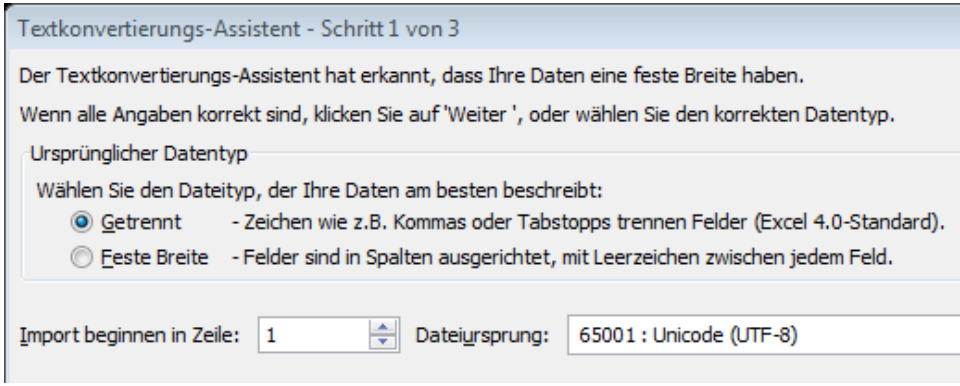


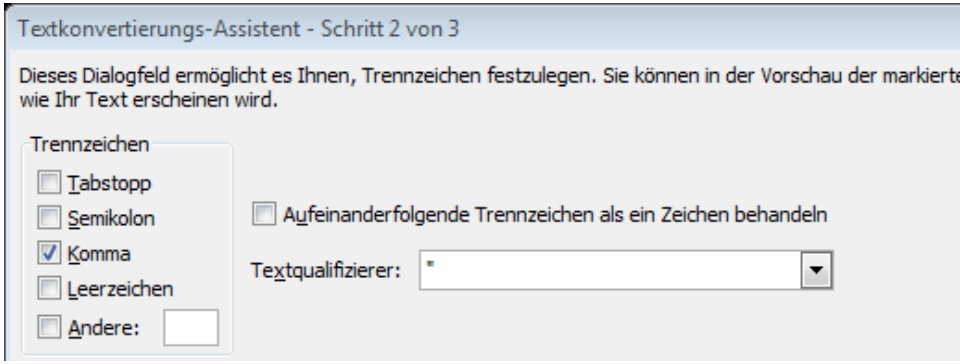
- http://osdoc.cogsci.nl/manual/logging/

Creating mouse-tracking experiments
Opening logfiles

- Logfiles are saved in csv format – specifications
  - Separator: “,”
  - Decimal point (not decimal comma)
  - Encoding: UTF-8
- Open csvs in Excel via text import assistant
- Task
  - Run experiment once and look at the logfile



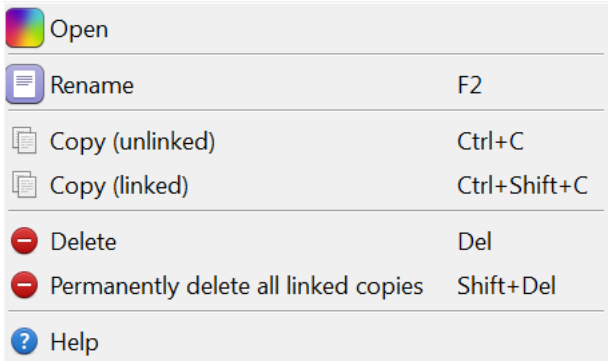




Creating mouse-tracking experiments

Copying items

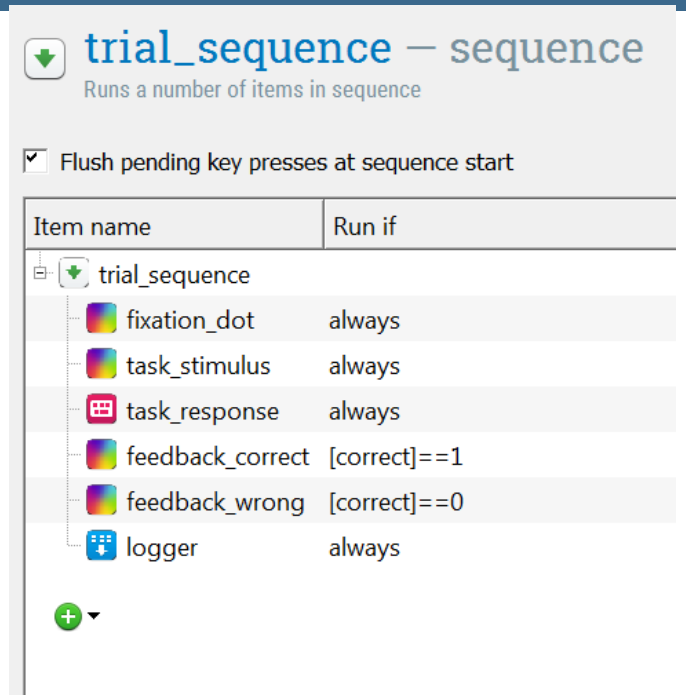
- Different options for copying an item
  - Unlinked copy
    - create a duplicate of the item that is not connected to the original item anymore
    - → if original item is changed, new item is not affected
  - Linked copy
    - create a duplicate of the item that is actually the very same item
    - → if either item is modified, the other changes as well



Creating mouse-tracking experiments

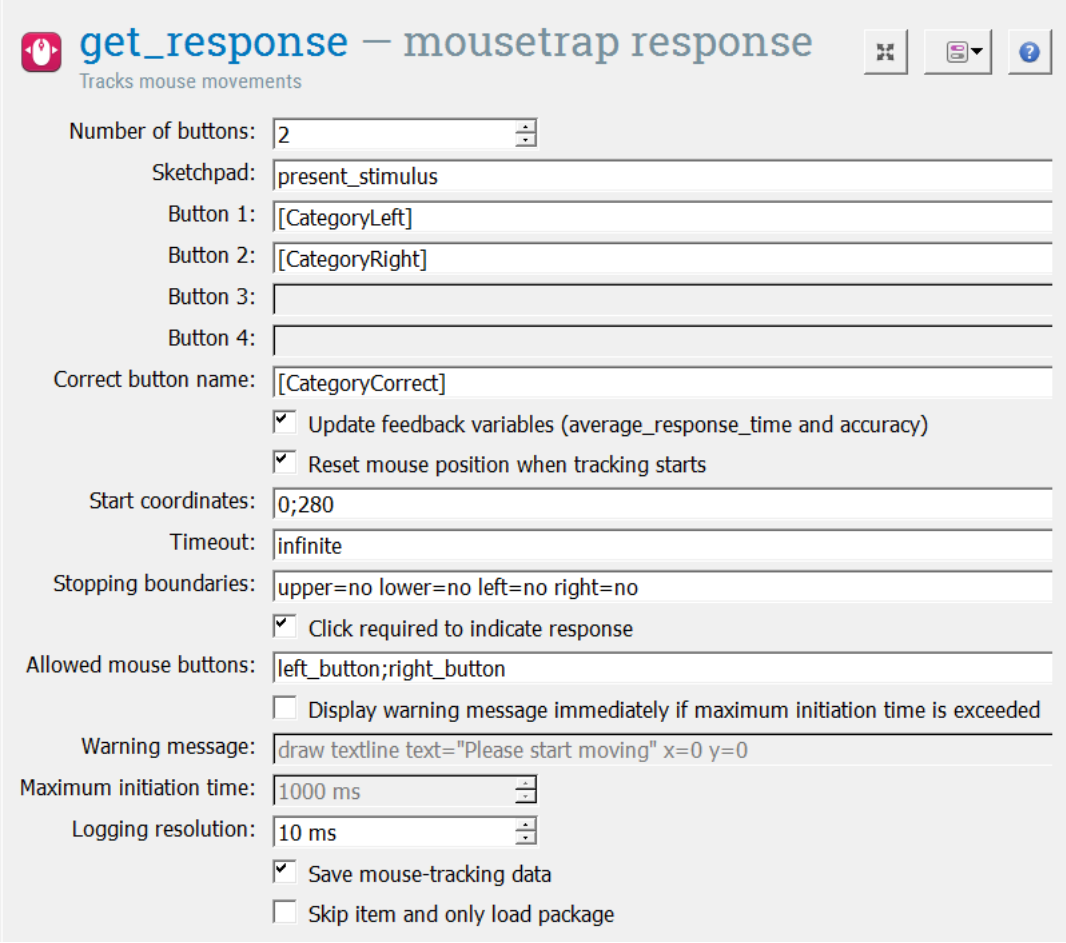
Conditional presentation of items and elements

- **Run if** in sequence and **show if** on sketchpad
  - Specify conditions under which an item is presented
  - Specify complex conditions by using **and**s and **or**s
- Can be used
  - … for implementing experimental conditions
  - … for giving feedback
  - … for adapting to a participant response
- Examples
  - [correct]==1
  - [correct]==1 and [response_time] >= 2000
  - [condition]==2 or [condition]==3
  - [condition]!=1
- Additional values are: always and never (for ignoring items)



- http://osdoc.cogsci.nl/manual/variables/#using-conditional-if-statements

Creating mouse-tracking experiments
Mouse-tracking settings



- **Define buttons**
- Automatically **code correctness** of response (correct = 0 or 1), e.g., to provide feedback
- **Reset cursor position** at tracking onset to exact coordinates
- **Limit** maximum **response time**
- Use **dynamic start** procedure (cf. Scherbaum & Kieslich, in press)
- End trial on **click** vs. on **touch**
- Issue immediate **warning** for long **initiation times**

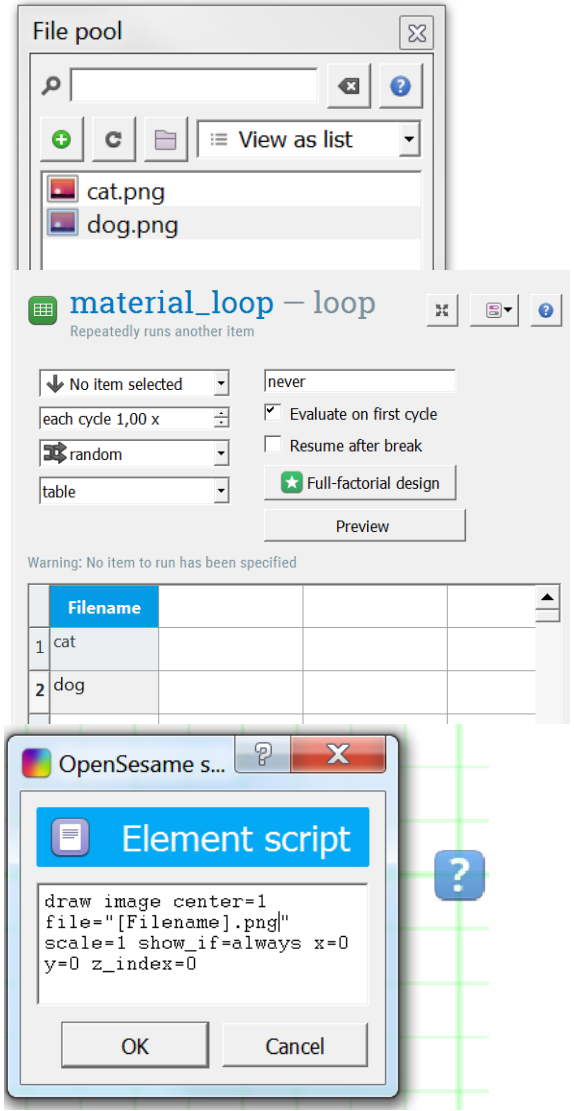
Creating mouse-tracking experiments
Mousetrap settings

- Task: Try out different mousetrap_response settings
  - Center the cursor position on exact coordinates when tracking starts
  - Record response and finish tracking as soon as the cursor touches one of the buttons (needs not to be clicked)
  - Only allow clicks on the left button of the physical mouse
  - Specify which variable contains the correct response and give feedback to the participant

Creating mouse-tracking experiments

File pool & pictures

- File pool
  - Is included in the experiment file itself
  - Can be used to include pictures (and other files) in the experiment
- Pictures
  - Can have various formats – however, especially the **.png** format seems recommendable
  - Rescaling should not be performed in OpenSesame but before, as it can lead to lags between trials
  - If pictures should vary between trials, set the file name in a loop and change the OpenSesame script of the image

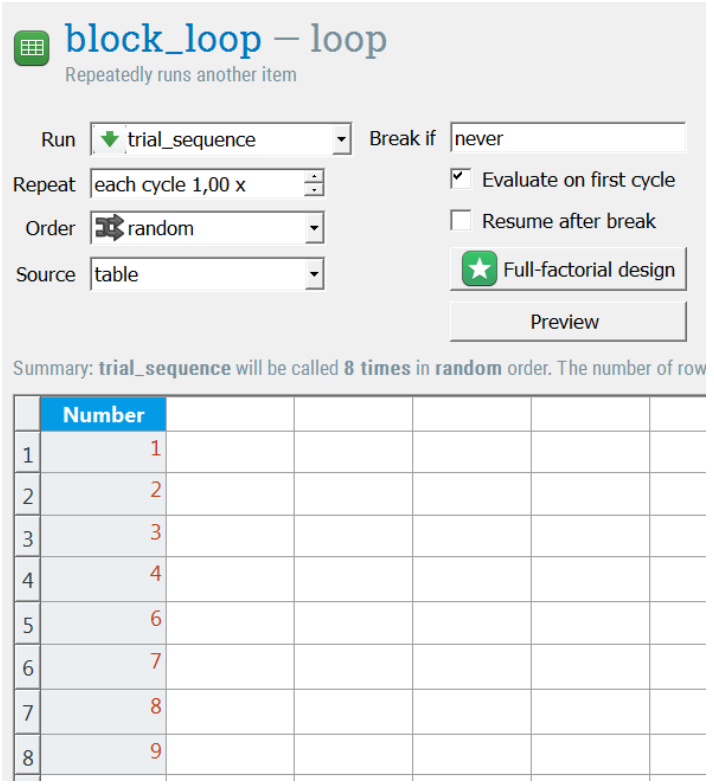


Randomization & manipulations

Randomizations & manipulations

Loops

- Loops can be used to store material or to vary independent variables
  - **Variables** stored in columns
  - Distinct **trials/conditions** stored in rows (= cycles)
- **Repeat**: number of repetitions of each cycle
  - Total repetitions = cycles x repeat
  - If repeat is < 1, a randomly selected subset of trials is chosen (e.g., 50 % of trials for 0.5)
- **Order** refers to the order with which all trials are presented
  - **Random** = every trial is randomly drawn without replacement (ignoring cycles x repeat structure)
  - **Sequential** = every trial in sequential order

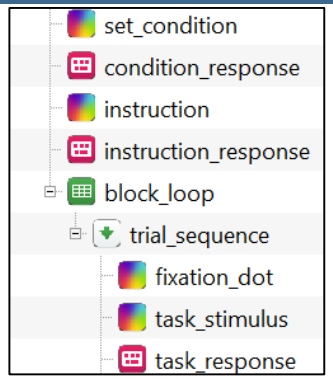


- http://osdoc.cogsci.nl/manual/structure/loop/

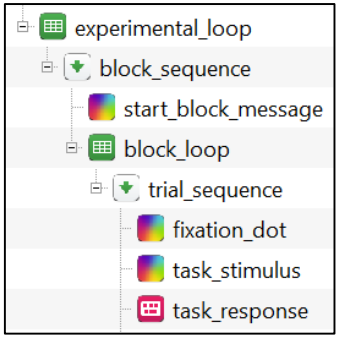
Randomizations & manipulations

Implementing randomizations in OpenSesame

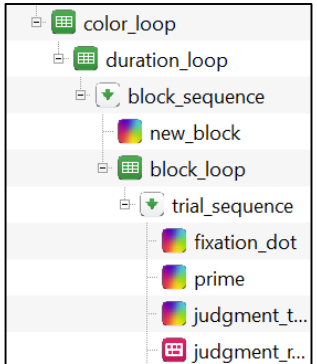
- **Between participants**
  - i.e., participant is in one condition set for whole experiment and every participant works on the same set of items
  - Assign condition at the beginning of the experiment (using a keyboard_response item)



- **Within participants**
  - a) **One factor** is manipulated and **same items are presented for each factor level (blocked)**
    - use one loop item for factor & nested within this a loop for material



  - b) **Several factors** are manipulated and same material is presented for each combination of factor levels (**blocked**)
    - i. If all combination of factor levels should be presented **in random order**, create combinations of all levels within one loop (see (a))
    - ii. If structure is **hierarchical**, i.e., for each level of one factor first all levels of the other factor are presented: use nested loops

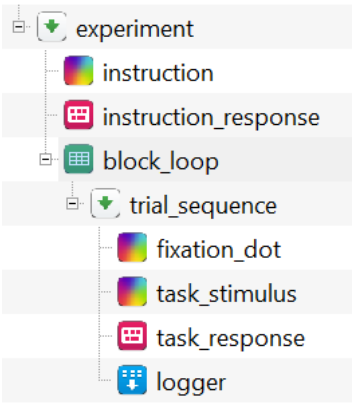


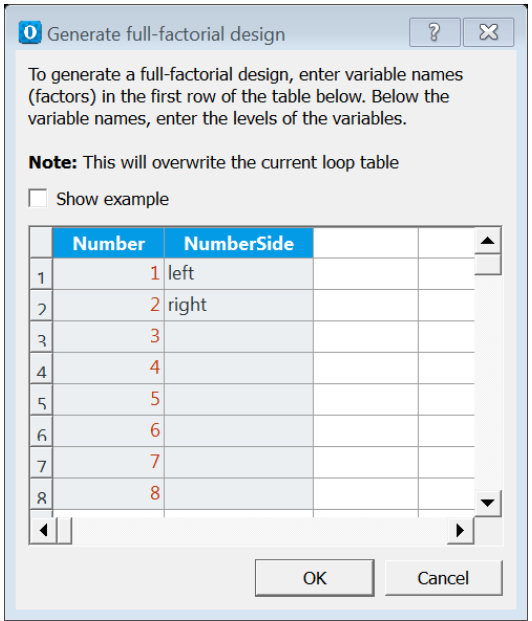
- Note: if items should repeatedly be presented: make use of **repeat**

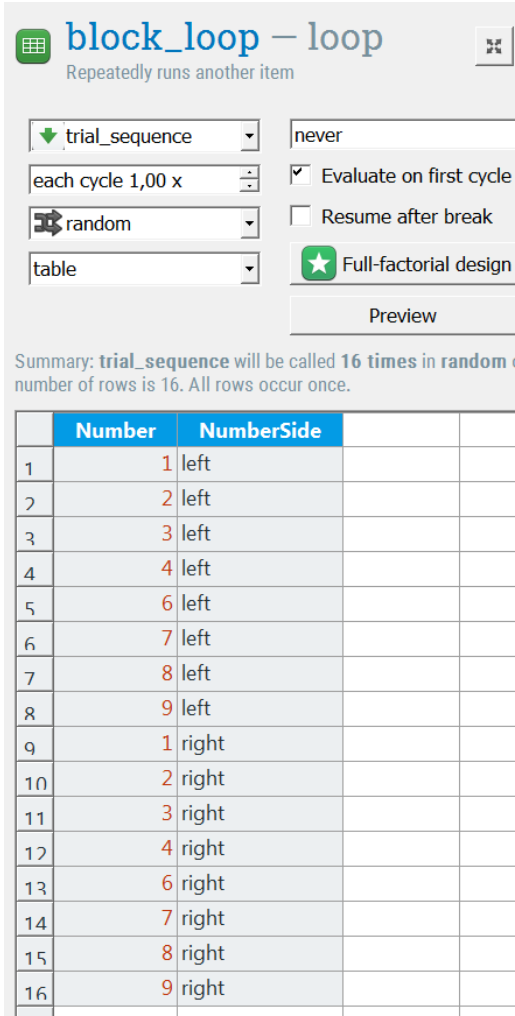
Randomizations & manipulations

Within participants: Full-factorial

- Case: **all items are presented for all factor levels**
  - Idea: One factor is manipulated within participants and same items are presented for each factor level – however, the presentation should not be blocked for factor levels but **everything is presented in random order**
  - Solution: Create one large loop where each factor level and item are combined (make use of **full-factorial design**)







Randomizations & manipulations

Crossed randomizations

- **Crossed randomization**
  - Idea: One factor is manipulated within participants and **items** are **randomly assigned to one of the factor levels** – importantly, each item is only presented **once**
  - Besides, presentation is not blocked but all items are presented in random order
- Solution with **nested loops**
  - One loop for items – within this loop: another loop for within factor where repeat is set to **1/number of factor levels**
  - Can also be used, e.g., to counterbalance presentation order when item pairs are presented
  - Problem: cannot ensure that number of items (i.e., number of trials) for each factor level are exactly the same
- Solution using **advanced loop operations**
  - Include items and within factor levels in the same loop in different columns
  - Use the advanced loop operation **shuffle** to only shuffle the column containing the within factor levels

Randomizations & manipulations

Advanced loop operations

- **Advanced loop operations**
  - Can be used for implementing advanced randomizations in loops
  - Have to be specified in the OpenSesame script of the loop item – after the setcycle commands (i.e. at the end of the script)
- Particularly useful **operations**
  - **shuffle cue:** Shuffle the column called cue → implement crossed randomizations
  - **shuffle_horiz word1 word2:** Shuffle the columns word1 and word2 horizontally → **useful for counterbalancing presentation order**
  - **weight w:** Repeat each rowby a weighting factor specified in column w → useful if each stimulus should be presented with a different frequency
- To test if a operation works as expected, use the **preview** feature

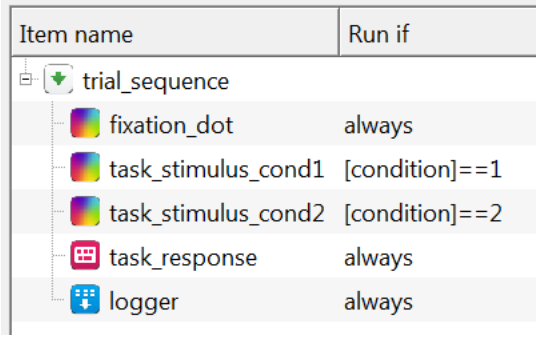
▶ http://osdoc.cogsci.nl/manual/structure/loop/#advanced-loop-operations

Randomizations & manipulations

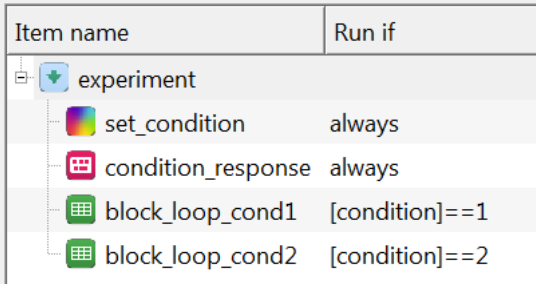
How to implement manipulations

- If manipulation affects specific task characteristic (e.g., color of font or different target letters)
  - Modify specific elements with experimental variables
  - Use **show if** condition

- If difference between conditions affects complex task changes
  - Complex layout changes
    - Create different sketchpads and use **run if** condition



  - Complex task changes
    - Use different loops/sequences and set run if condition for them
    - Identical elements within the different conditions can be realized by creating linked copies



Randomizations & manipulations

Block & trial structure

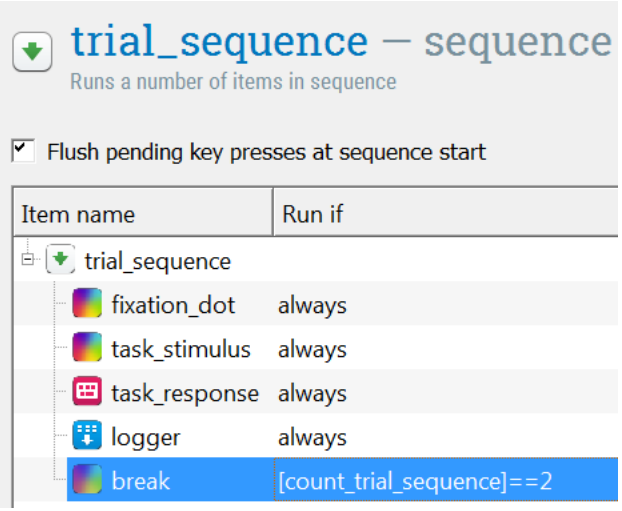
- Situation: Different blocks of trials are used with the same task
  - Typically: material (i.e., items) change but task remains the same
  - e.g., practice and task block
  - Solution: use different loops that run the same sequence

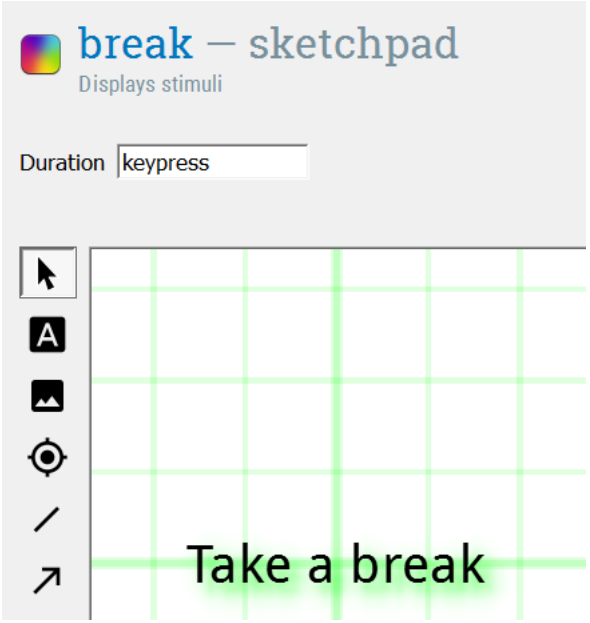
- Situation: block of trials has a specific structure
  - e.g., first 10 test items are presented, then the actual task items follow
  - Solution: the block actually consists of multiple blocks
    → use different loops that run the same sequence

Randomizations & manipulations

Events at specific trials

- Situation: Something happens at trial X
  - e.g., task has 6 trials and there should be a break after trial 3
  - Solution: use count_sequence to set run if condition
  - Note: trial counter starts at 0 (i.e., count_trial_sequence has value 0 in first trial, so if you want to make a break after 3 trials, the counter should be equal to 2)
  - More complex events can be realized using inline_scripts (also using the count_sequence variable)

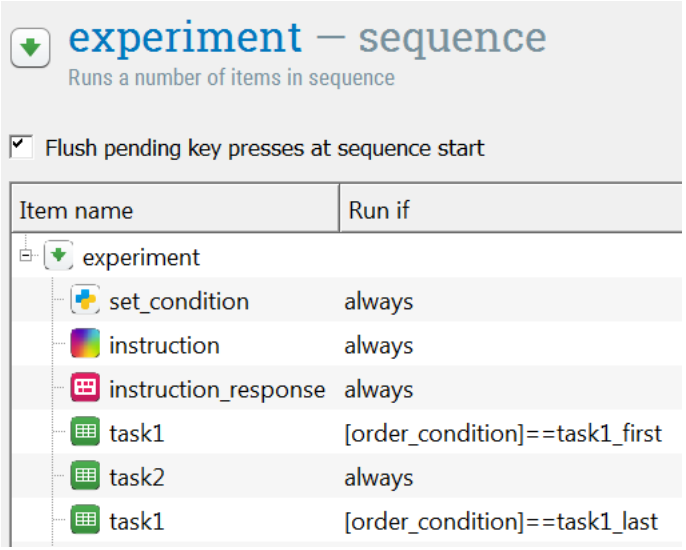




Randomizations & manipulations

Manipulation of task order

- Situation: The order of two tasks should be manipulated
  - Solution: Create linked copy of whole loop and set run if condition



Design factors

Creating mouse-tracking experiments
Methodological considerations

- General challenge when designing a mouse-tracking study
  - Movements should reflect developing **commitment** not information search → **minimize** amount of **new information** after tracking onset
  - **Preferences** should not develop **before** tracking starts → **critical information** should only be made available at the **last** moment

- Mouse **start positions** should be **comparable** across trials
  - Participants have to click on a **centered button** to start the trial
  - Exactly identical start positions across trials achieved by **resetting** mouse or by **computational alignment** during analysis

- **Counterbalancing positions** across trials / participants
  - Vary which option is presented on which side (left vs. right)
  - Can be done between trials or between participants (depending on study)

Design factors
Overview

- Researchers face a number of **design choices** when creating mouse-tracking experiments
  - Starting procedure (static, restricted initiation time, dynamic)
  - Cursor speed settings (velocity & acceleration)
  - Indicate response via click vs. touch
- Some authors have given **recommendations** about designing mouse-tracking studies (Fischer & Hartmann, 2014; Hehman et al., 2015)
- Empirical **validation** studies are being conducted (Scherbaum & Kieslich, in press; Kieslich et al., in preparation)

Design factors
Preliminary summary of findings

- **Response indication**
  - **Click** on button leads to larger effects than **touch** – effect related to higher proportion of trials with extreme movements to non-chosen option
- **Mouse sensitivity settings**
  - Did not significantly influence effect of interest in static setup – although **default** settings generally lead to more extreme curvature than **reduced mouse speed**
  - Reducing mouse speed becomes relevant for dynamic start condition to ensure stimulus information can be acquired during upwards movement
- **Starting procedure**
  - Restricting **maximum initiation time** led to larger effects – a **dynamic start** only influenced shape but not effect size
  - However, restricting initiation times also led to largest proportion of excluded trials (and seemed to be challenging for some participants)

Design factors
Starting procedure

- Simplest setup: static start
  - Stimulus presented immediately (or after a short delay) after click on start button
  - Participants can freely decide when to initiate their movement
  - Potential risk: participants make their decision first and then initiate mouse movement
  - Studies using this procedure often still find effects but in some trials (with extremely straight trajectories) this problem might occur
- Alternative: ensure early movement initiation
  - Alternative I: Participants explicitly told to initiate movement within certain time limit (around 400-600 ms) – if they take longer, warning message is displayed after the trial
  - Alternative II: dynamic start: Participants first have to move the mouse upwards for 50 px so the stimulus is presented
  - These procedures tend to lead to larger effects and more consistent movements – alternative I sometimes hard for participants to accomplish
  - Challenge: is the task simple enough that participants can make the complete decision during the upwards movement?

Design factors

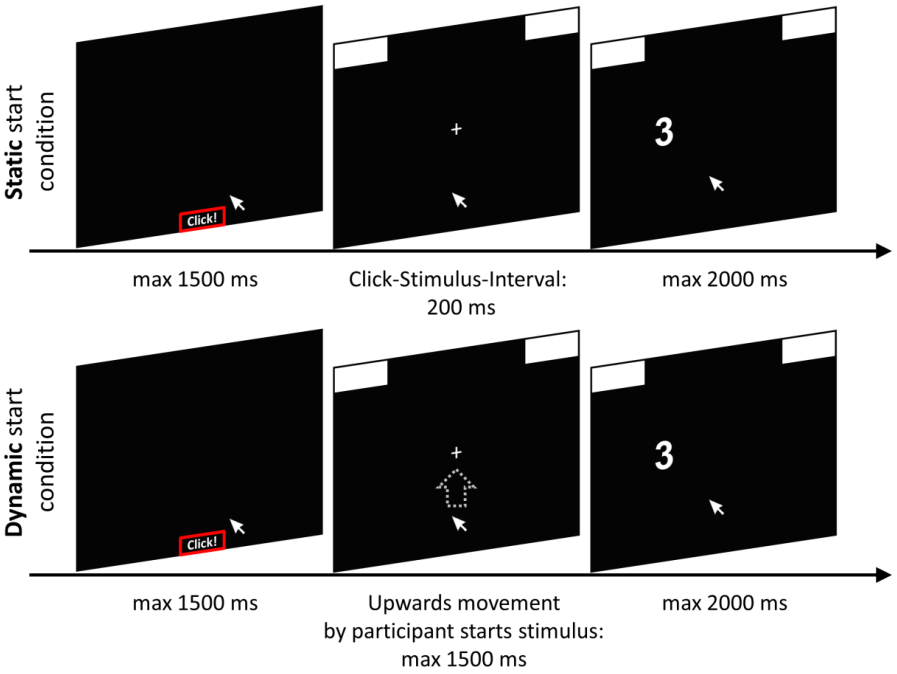
Cursor speed and acceleration

- Different settings have been employed
  - Default: medium speed, acceleration enabled
  - Slow: reduced speed, acceleration disabled
    → Reduction of speed and disabling of acceleration important for starting procedures that encourage early movement initiation → ensure that decision can be completed during upwards movement

Starting procedure: Static vs. dynamic start

Method (Scherbaum & Kieslich, in press)

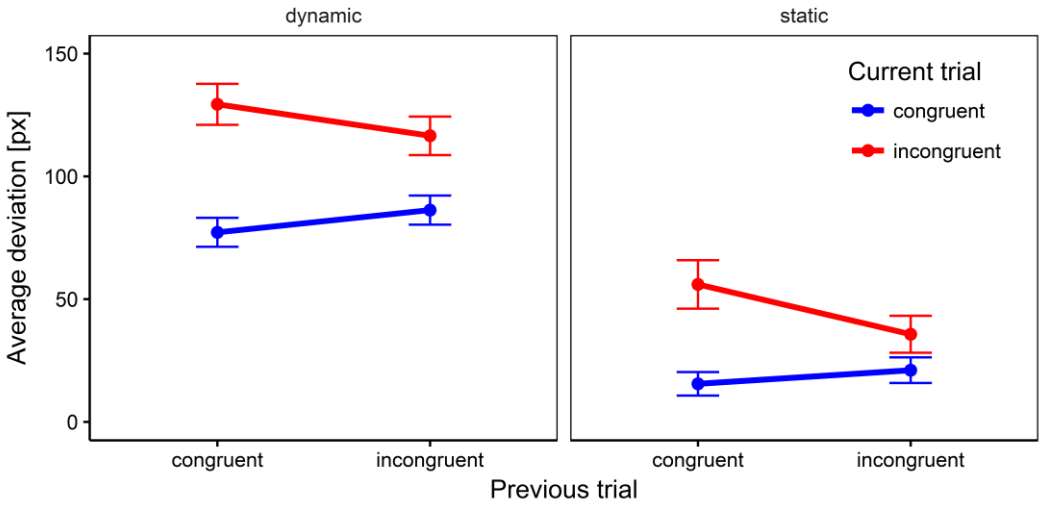
- Mouse-tracking in Simon task
  - Participants **click on left vs. right** option depending on stimulus (left if number < 5, otherwise right)
  - Position of stimulus varied (left vs. right) so that desired response and position are either **congruent** or **incongruent**
- Variation starting procedure
  - **Dynamic**: move upwards to display stimulus (data from Scherbaum et al., 2010)
  - **Static**: stimulus displayed after fixed interval of 200 ms (typical duration of movement initiation in dynamic condition) (new data)



Starting procedure: Static vs. dynamic start

Discrete effects: Results for average deviation

- Simon effect and congruency sequence effect replicated in both conditions
- No significant interaction of theoretically important effects with starting procedure

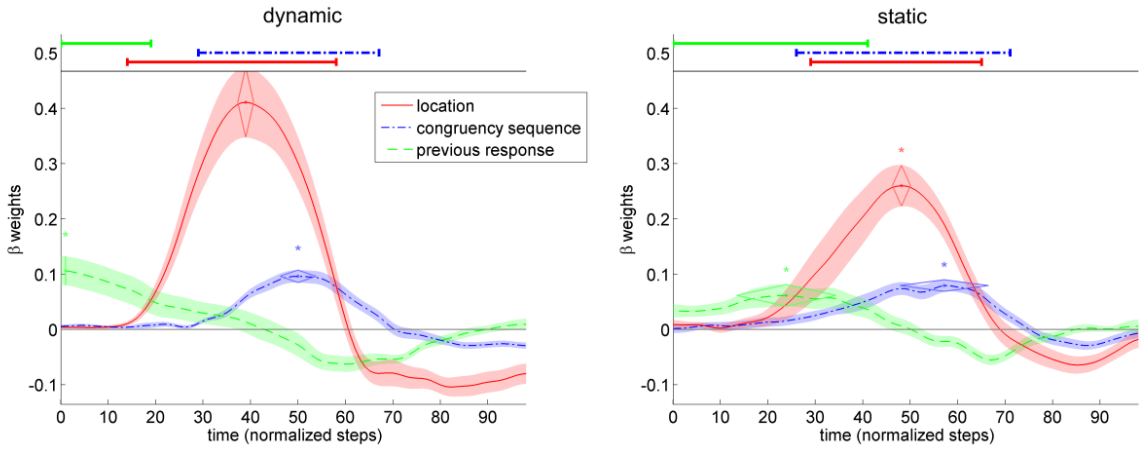


Error bars represent 1 *SEM*.

Starting procedure: Static vs. dynamic start

Dynamic effects: Time-continuous angle regression

- Time continuous multiple regression predicting vertical movement angle at each time point
- Predictors
  - location (congruency)
  - congruency sequence (same / different)
  - previous response (same / different)
- Effects stronger and more temporarily distinct in dynamic starting condition

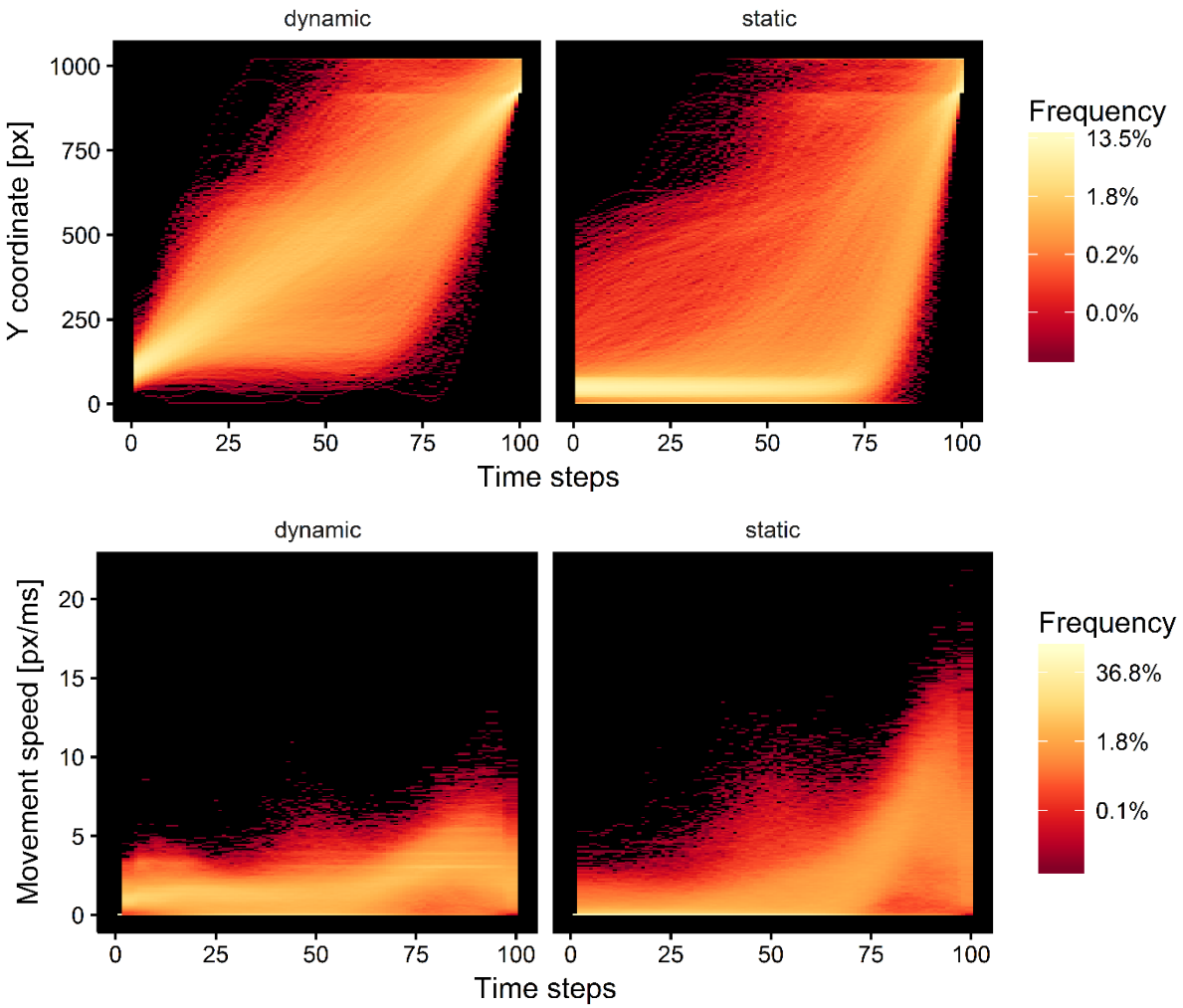


Average β weights per time step and predictor.
Lines indicate segments of β weights significantly > 0.

Starting procedure: Static vs. dynamic start

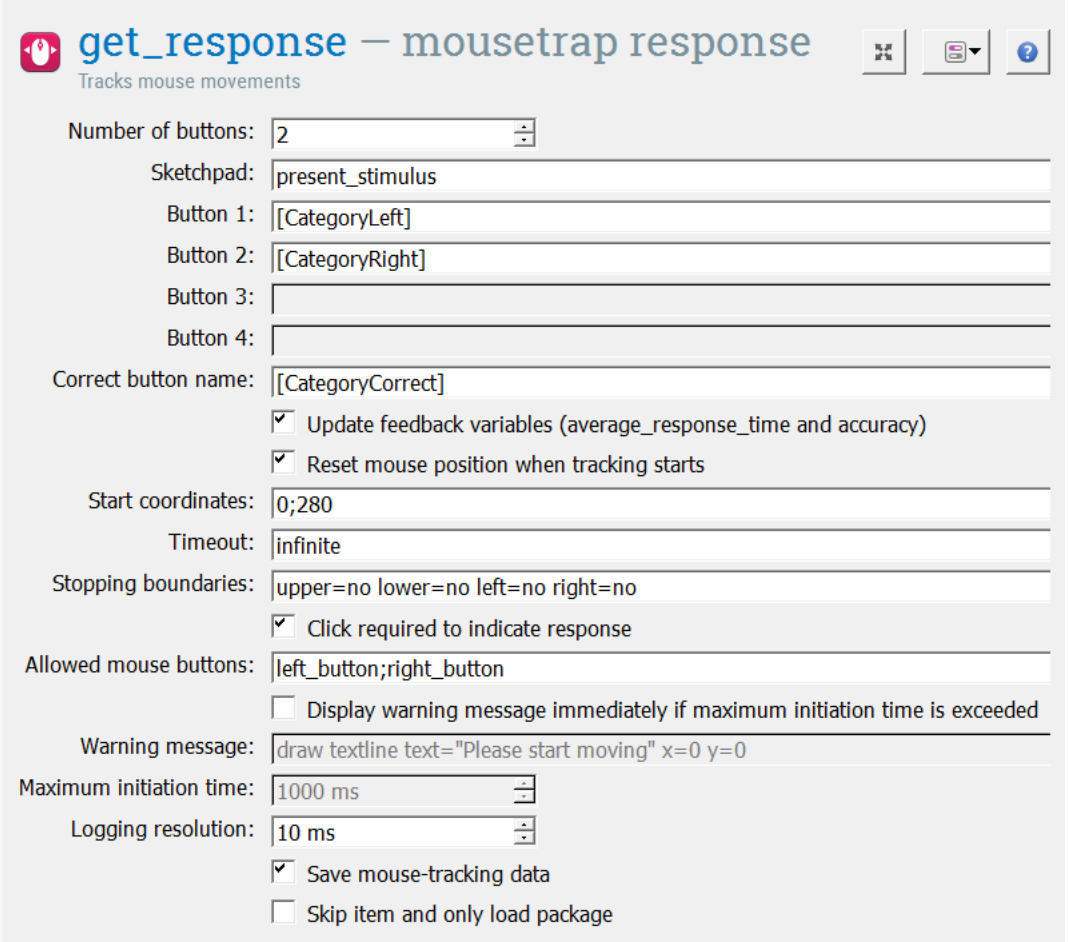
Movement consistency

- Smooth and consistent upwards movement in dynamic starting condition
- Participants in static starting condition often stay at bottom of screen for more than half of the trial before moving upwards quickly



Creating mouse-tracking experiments
Implementing a dynamic start procedure

- Task: Implement a dynamic start procedure
  - Include a new sketchpad without stimulus before decision screen as well as a new mousetrap_response item
  - Reset mouse cursor to center of the start button when tracking starts
  - Set number of buttons to 0 and specify an upper tracking boundary 50 px above start (→ subtract 50 px as px in OpenSesame increase towards the bottom)



Your experiments

Creating mouse-tracking experiments (Tuesday)

- 09:00-11:00 OpenSesame & mousetrap-os introduction
- 11:00-12:00 Build experiment
- 12:00-13:00 Lunch break
- 13:00-15:00 Build experiment
- 15:00-16:00 Register experiment at OSF (Michael)
- 16:00-17:00 Keynote Neil Stewart
- 17:00-18:00 Meet the Scientist
- 17:00-19:00 Participate in experiments (Lab computers)

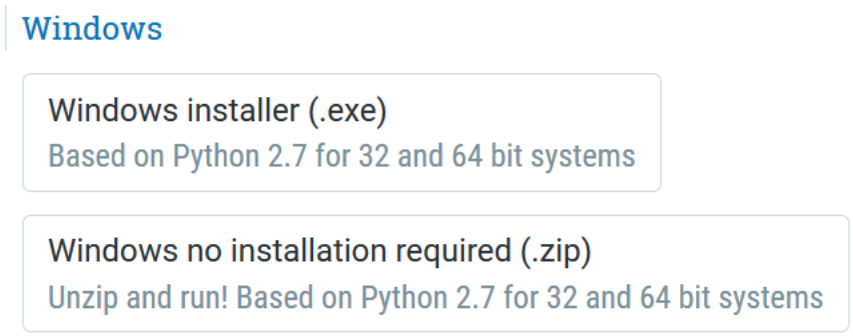
Running experiments

Collecting data: General concerns

- Is it easily possible to easily **identify the experimental block** of interest in each row of the log file?
  - OpenSesame does not directly log the sequence it is running
  - As soon as an experimental variable is set and not changed anymore, its value will be logged in all subsequent trials
  - → Introduce variable indicating the experimental block
- Are **items** stored conveniently for later analyses?
  - → Introduce ID variable for the items
- Do you log all relevant **independent** variables?
  - Especially important for those variables that you randomly define in inline_scripts
- Do you log all relevant **dependent** variables?
  - response, RT,…
- Do you have open text input fields in your experiment?
  - If quotation marks are used in responses, this might mess up the data import

Running experiments in the lab

- OpenSesame needs to be available on every computer
  - Can be installed using the installer
  - Alternative version available that can directly be copied to computer → does not require admin rights



- http://osdoc.cogsci.nl/download/

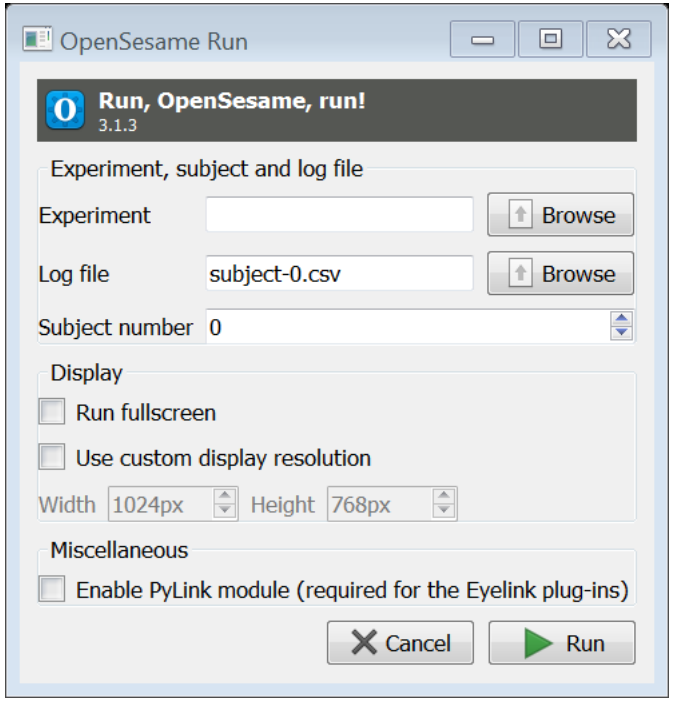
Running experiments: Potential problems

- On some older computers, OpenGL might not be supported
  - Disable OpenGL in expyriment back-end
  - Or switch to legacy back-end
  - Note that this might slightly reduce temporal accuracy
  - http://osdoc.cogsci.nl/manual/backends/

- On Mac computers the default runner might cause problems
  - External runner is not working on Mac
  - Go to preferences and change runner to inprocess
  - http:///osdoc.cogsci.nl/manual/runners/

Running experiments in the lab

- OpenSesame (runtime) / opensesamerun
  - Simple solution if we just want to run our experiment
  - Automatically installed with OpenSesame
- At start
  - Specify experiment file
  - Specify logfile (in folder)
  - Specify subject number



- http://osdoc.cogsci.nl/manual/opensesamerun/

Running mouse-tracking experiments

- Run experiments full screen
- Ensure cursor speed and acceleration settings are identical across computers
- Ensure participants have enough space for moving the mouse
  - move keyboard out of the way
  - design experiment so that participants can complete the entire experiment by using only the mouse
- Assess handedness and the hand that participants used for moving the mouse

Thank you!

Questions and comments are highly appreciated!

Now & via email: kieslich@psychologie.uni-mannheim.de
dirk.wulff@gmail.com

Mousetrap-os plugins: https://github.com/pascalkieslich/mousetrap-os

Mousetrap R package: http://pascalkieslich.github.io/mousetrap/

Thanks:

Felix Henninger, co-developer of mousetrap-os plugin and R package

Jonas Haslbeck & Michael Schulte-Mecklenbeck, co-developers of mousetrap R package

Mila Rüdiger and Monika Wiegelmann for data collection and testing

DAY 3:
ANALYZING MOUSE-TRACKING DATA

Pascal Kieslich (University of Mannheim) & Dirk Wulff (University of Basel)

Workshop at the EADM Summer School 2018 in Salzburg, Austria

Analyzing mouse-tracking data (Wednesday)

- 09:00-10:00 General R introduction (Dirk)
- 10:00-12:00 Preprocessing and curvature analyses (Pascal)
  - Data import
  - Preprocessing
  - Trajectory plots
  - Curvature indices
- 12:00-13:00 Lunch break
- 13:00-14:30 Additional indices and analyses (Pascal)
  - Trial-level indices
  - Temporal analyses
- 14:30-16:00 Trajectory types (Dirk)
  - Heatmaps
  - Clustering
  - Prototypes

Software for the workshop

- To create mouse-tracking experiments, first install OpenSesame. It is available from http://osdoc.cogsci.nl/3.2/download/.
- To install the mousetrap plugin for OpenSesame, follow the instructions at https://github.com/pascalkieslich/mousetrap-os#installation. Please make sure to install the latest version of OpenSesame (3.2.4) and the development version of the mousetrap-os plugin.

- To analyze mouse-tracking data install R (https://www.r-project.org/) and RStudio (https://www.rstudio.com/products/rstudio/download/).
- Afterwards, please run the following command in R to install the required packages:

```
install.packages(c("readbulk","mousetrap"))
```

- Screen resolution of experiment for lab computers: 1280 x 1024 px

General R introduction

Analyzing mouse-tracking data

R

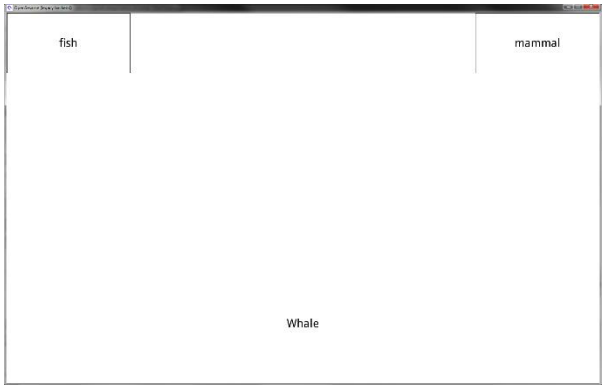
- R
  - Statistical programming language
  - Open-source and cross-platform
  - https://www.r-project.org/
- RStudio
  - Integrated development environment for R
  - Provides graphical user interface (GUI) for working with R
  - http://www.rstudio.com/
- R packages
  - Users can develop new packages / libraries to extend R's functionality
  - Can be uploaded to CRAN (reviewed by CRAN Team)
  - Can be installed using `install.packages()` or via GUI in RStudio
  - Load required packages at the beginning of each session using `library()`

Preprocessing & curvature analyses

Analyzing mouse-tracking data

Replication study of Dale, Kehoe, & Spivey (2007)

- **Animal categorization task**
  - **Typical exemplars** only share features with correct category (e.g., cat as mammal)
  - **Atypical exemplars** share both features with correct and competing category (e.g., whale with mammal and fish)



- **Main hypothesis**
  - **Increased competition** when categorizing atypical exemplars
    - ➢ Mouse trajectories with deviation towards competing category

- **Replication study** (Kieslich & Henninger, 2017)
  - Same material (translated into German) and procedure, but higher resolution and different aspect ratio
  - *N* = 60 students from the University of Mannheim
  - Material, data, and analyses at https://github.com/pascalkieslich/mousetrap-resources

Analyzing mouse-tracking data

Typical mouse-tracking analyses steps

- Filter trials
- Alignment of trajectories
- Remapping of trajectories
- Time (and space) normalization

- Trial-level indices (e.g., MAD for curvature)
- Dynamic measures (e.g., development of acceleration over time)

- Plot of individual trajectories
- Distribution of trial-level indices
- Identification of chaotic trajectories

- Potential aggregation of trajectories and trial measures
- Per participant and condition

- Comparison of measures and trajectories
- Typically between experimental conditions in within designs

Analyzing mouse-tracking data
Reading and merging raw data

- OpenSesame raw data
  - OpenSesame stores raw data in **csv files**
  - In our experiment, each line represents a new trial
  - Data for each participant stored in separate csv files

- Reading data into R
  - R provides many different functions and packages for reading raw data of various formats
  - If data were collected using mousetrap in OpenSesame, the `read_opensesame` function of the **readbulk** package (Kieslich & Henninger, 2016) can be used to read and merge all individual participant files
  - If data were collected using MouseTracker (Freeman & Ambady, 2010), the `read_mt` function of the **mousetrap** package can be used

Analyzing mouse-tracking data

Getting started

- General preparation in R / RStudio
  - Install packages: `install.packages(c("readbulk","mousetrap"))`
  - Create a new project
  - Create and save a new R script
- Loading package and data
  - Load the mousetrap library: `library(mousetrap)`
  - The raw data from Kieslich & Henninger (2017) are already provided in the package: `raw_data <- KH2017_raw`
- Explore raw data using different functions
  - Click on the raw_data in RStudio‘s Environment pane
  - Try out `str(raw_data)`
  - Check number of correctly answered trials using `table(raw_data$correct)`
- Subset data
  - Use `subset` function to only keep correctly answered trials: `raw_data <- subset(raw_data, correct==1)`

Analyzing mouse-tracking data
Mousetrap R package

- **Mousetrap** (Kieslich, Wulff, Henninger, Haslbeck & Schulte-Mecklenbeck, in preparation)
  - **R package** for processing, analyzing and visualizing mouse-tracking data
  - Available on **CRAN** – can be installed using `install.packages("mousetrap")`
  - **Actively developed** – new versions published regularly
  - Sign up to our mailing list to receive updates (http://eepurl.com/co1AqX)

- Package overview and documentation
  - Package overview: http://pascalkieslich.github.io/mousetrap/
  - Overview of functions within R: `package?mousetrap`
  - Function specific help: `?mt_import_mousetrap`
  - Demonstration of analyses using the replication data: https://github.com/pascalkieslich/mousetrap-resources

Analyzing mouse-tracking data

Mousetrap R package

- Automatic **merging** of individual data (with package `readbulk`)
- Built-in support for **different** mouse-tracking data **formats,** including MouseTracker (.mt) files

- **Alignment** of trajectories
- **Remapping** of trajectories
- **Time** and space **normalization**
- Resampling
- Averaging

- **Velocity** and **acceleration**
- Movement **angles**
- Trial-level **indices**, e.g., maximum deviation, sample entropy
- **Bimodality** analyses
- **Clustering**

- Simple **aggregation** at multiple levels
- Both for trajectory data and trial-level indices
- Export in long or wide format

- Preparation of data for **ggplot2**
- **Plot** individual and aggregate **trajectories**
- **Riverbed** plots and **heatmaps**
- **Animation** of trajectories

Installation from CRAN: `install.packages("mousetrap")`
More information: http://pascalkieslich.github.io/mousetrap/

Analyzing mouse-tracking data
Data import

- To perform analyses on mouse-tracking data they have to be transformed into a **mousetrap** data object first

- Depending on the structure of the raw data, use one of the following import functions
  - `mt_import_mousetrap`: data from mousetrap plugin for OpenSesame
  - `mt_import_wide`: data in wide format, e.g., from MouseTracker
  - `mt_import_long`: data in long format, i.e., one row per recorded position

- Task
  - Import filtered raw data into mousetrap using the following command:
    ```
    mt_data <- mt_import_mousetrap(raw_data)
    ```
  - Explore imported data object using `str()` or RStudio's Environment pane

Analyzing mouse-tracking data

Mousetrap data object


```
> str(mt_data)
List of 4
 $ data           :'data.frame':      1064 obs. of  11 variables:
  ..$ subject_nr     : int [1:1064] 1 1 1 1 1 1 1 1 ...
  ..$ count_trial    : int [1:1064] 2 3 4 5 6 7 8 9 ...
  ..$ Condition      : Factor w/ 2 levels "Atypical","Typical": 2 2 ...
  ..$ Exemplar       : Factor w/ 19 levels "Aal","Alligator",..: 13 ...
  ..$ CategoryLeft   : Factor w/ 6 levels "Amphibie","Fisch",..: 2 4 ...
  ..$ CategoryRight  : Factor w/ 6 levels "Amphibie","Fisch",..: 5 5 ...
  ..$ CategoryCorrect: Factor w/ 5 levels "Fisch","Insekt",..: 4 4 4 ...
  ..$ response       : Factor w/ 6 levels "Amphibie","Fisch",..: 5 5 ...
  ..$ response_time  : int [1:1064] 1000 1387 2211 1606 2230 1219  ...
  ..$ correct        : int [1:1064] 1 1 1 1 1 1 1 1 ...
  ..$ mt_id          : chr [1:1064] "id0001" "id0002" "id0003" ...
 $ trajectories   : num [1:1064, 1:2159, 1:3] 0 0 0 0 0 0 0 0 ...
  ..- attr(*, "dimnames")=List of 3
  .. ..$ : chr [1:1064] "id0001" "id0002" "id0003" ...
  .. ..$ : NULL
  .. ..$ : chr [1:3] "timestamps" "xpos" "ypos"
 $ tn_trajectories: num [1:1064, 1:101, 1:4] 0 0 0 0 0 0 0 0 ...
  ..- attr(*, "dimnames")=List of 3
  .. ..$ : chr [1:1064] "id0001" "id0002" "id0003" ...
  .. ..$ : NULL
  .. ..$ : chr [1:4] "timestamps" "xpos" "ypos" ...
$ measures       :'data.frame':      1064 obs. of  20 variables:
  ..$ mt_id          : chr [1:1064] "id0001" "id0002" "id0003" ...
  ..$ xpos_max       : num [1:1064] 1 1 2 0 259 170 593 1 ...
  ..$ xpos_min       : num [1:1064] -609 -615 -577 -559 -822 -655 ...
  ..$ ypos_max       : num [1:1064] 831 780 861 826 755 769 878 908 ...
  ..$ ypos_min       : num [1:1064] 0 0 0 -4 0 0 0 0 ...
  ..$ MAD            : num [1:1064] -85.1 -65.4 99.2 -64.3 ...
  ..$ MAD_time       : num [1:1064] 671 811 1054 1130 ...
  ..$ MD_above       : num [1:1064] 0.807 8.562 99.184 0 ...
  ..$ MD_below       : num [1:1064] -85.1 -65.4 -32.3 -64.3 ...
  ..$ AD             : num [1:1064] -5.9 -14.1 5.5 -20.7 ...
  ..$ AUC            : num [1:1064] -46527 -13329 51842 -32054 ...
  ..$ xpos_flips     : num [1:1064] 3 3 3 3 2 1 3 2 ...
  ..$ ypos_flips     : num [1:1064] 0 0 0 1 0 2 1 1 ...
  ..$ xpos_reversals : num [1:1064] 1 1 1 0 1 1 3 1 ...
  ..$ ypos_reversals : num [1:1064] 0 0 0 1 0 0 0 0 ...
  ..$ RT             : num [1:1064] 1000 1387 2211 1606 ...
  ..$ initiation_time: num [1:1064] 511 441 774 260 552 640 30 20 ...
  ..$ idle_time      : num [1:1064] 670 807 1311 936 ...
- attr(*, "class")= chr "mousetrap"
```

data: data.frame containing general trial information (≈ all non-mouse-tracking data)

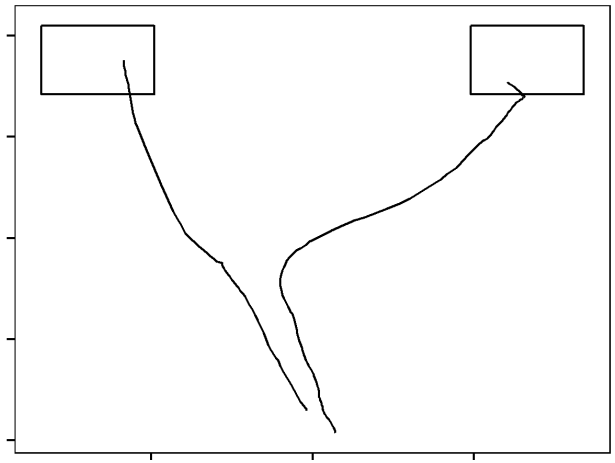
trajectories: 3-dimensional array of raw trajectories

tn_trajectories: 3-dim. array of time-normalized trajectories (added later)

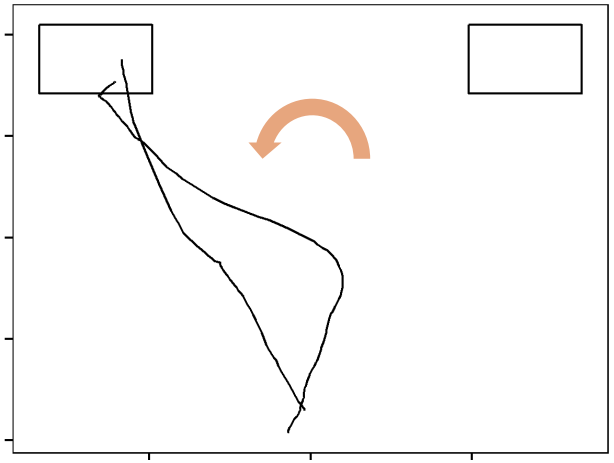
measures: data.frame containing mouse-tracking measures (will be calculated later)

Analyzing mouse-tracking data

Data preparation: Remapping and alignment

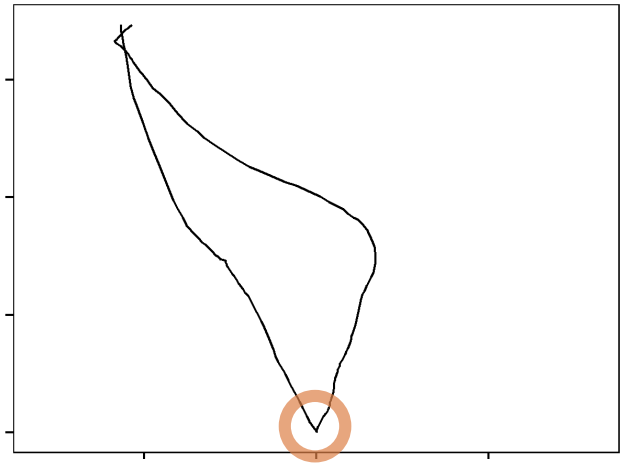


Raw data



Remapping

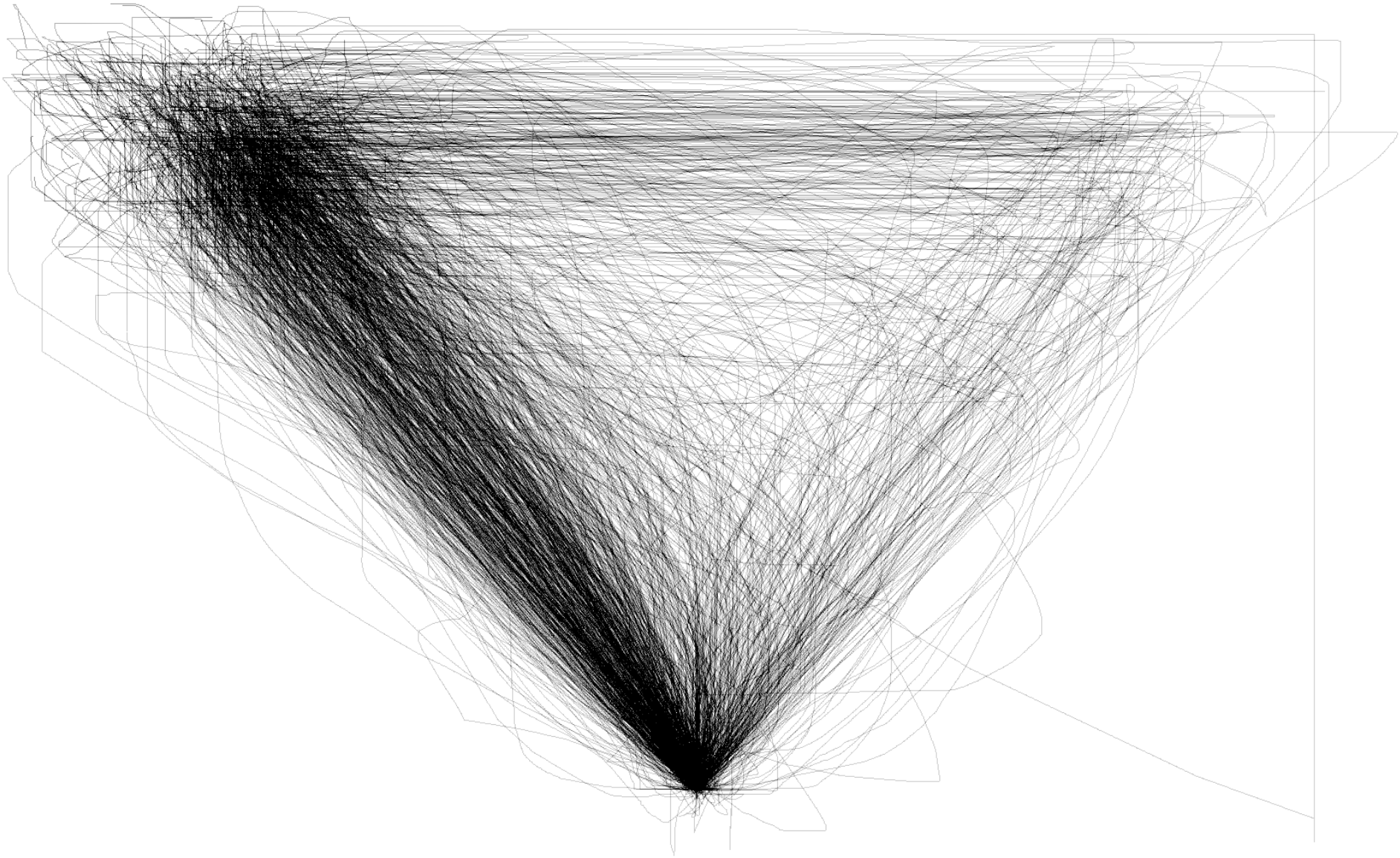
Equal direction



Alignment

Equal coordinate system

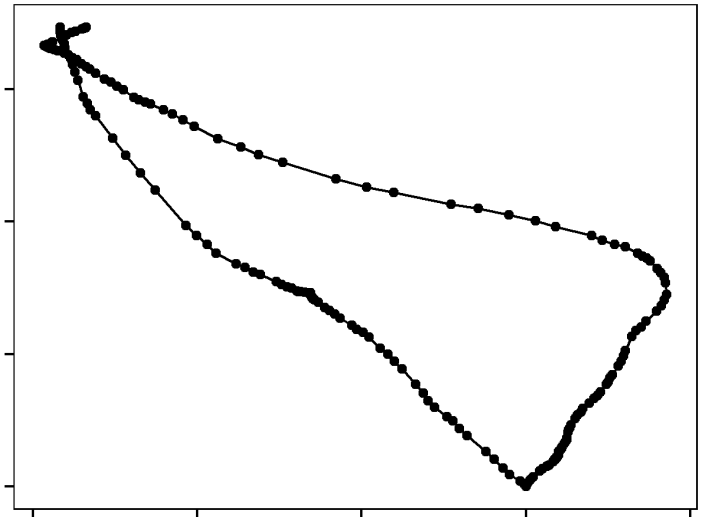
```
mt_data <- mt_remap_symmetric(mt_data)
mt_data <- mt_align_start(mt_data, start=c(0,0))
```



Analyzing mouse-tracking data

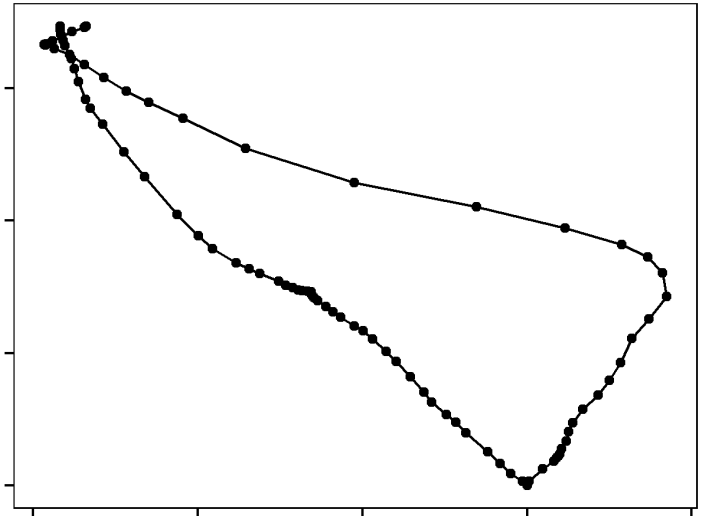
Data preparation: Time normalization

- Trials with differing response time vary regarding number of recorded coordinates
- To permit averaging across trials: time-normalization (cf. Spivey et al., 2005)
- Each trajectory divided into 101 equally spaced time steps using linear interpolation



Raw data

Constant sampling rate → Absolute time



Time normalization

Relative time steps

```
mt_data <- mt_time_normalize(mt_data, nsteps=101)
```

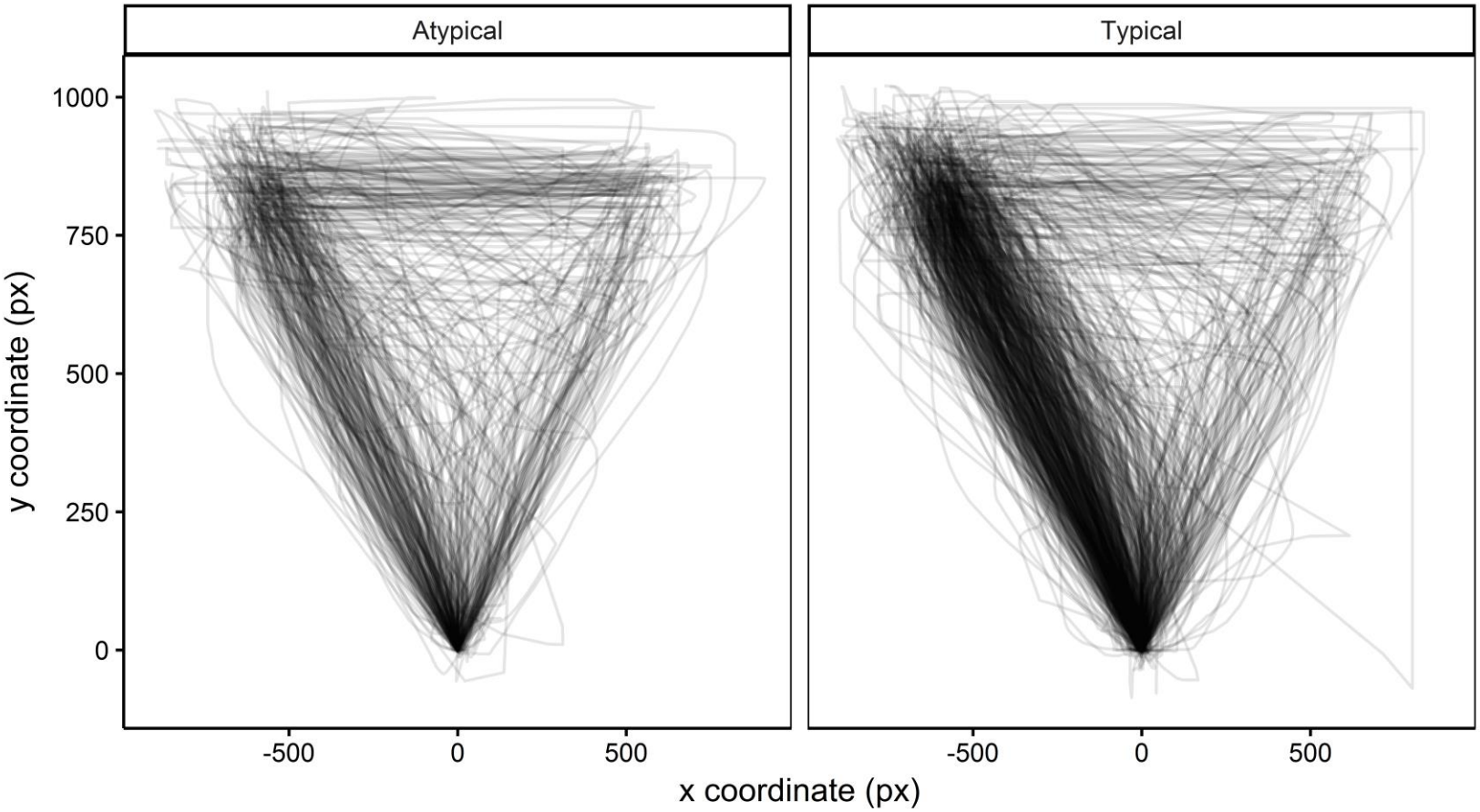
Analyzing mouse-tracking data

Data preprocessing

- Task
  - Remap and align the imported trajectories using
    ```
    mt_data <- mt_remap_symmetric(mt_data)
    mt_data <- mt_align_start(mt_data, start=c(0,0))
    ```
    (note that the resulting object is provided in the mousetrap package as KH2017)
  - Time normalize trajectories
    ```
    mt_data <- mt_time_normalize(mt_data, nsteps=101)
    ```
  - Explore mousetrap data object using `str()` or RStudio's Environment pane: Time-normalized trajectories have been added as an additional trajectory array called `tn_trajectories`
  - Most mousetrap functions allow the user to specify which trajectories should be used for a specific operation via the use argument, e.g.:
    ```
    mt_plot(mt_data, use="tn_trajectories")
    ```

Analyzing mouse-tracking data

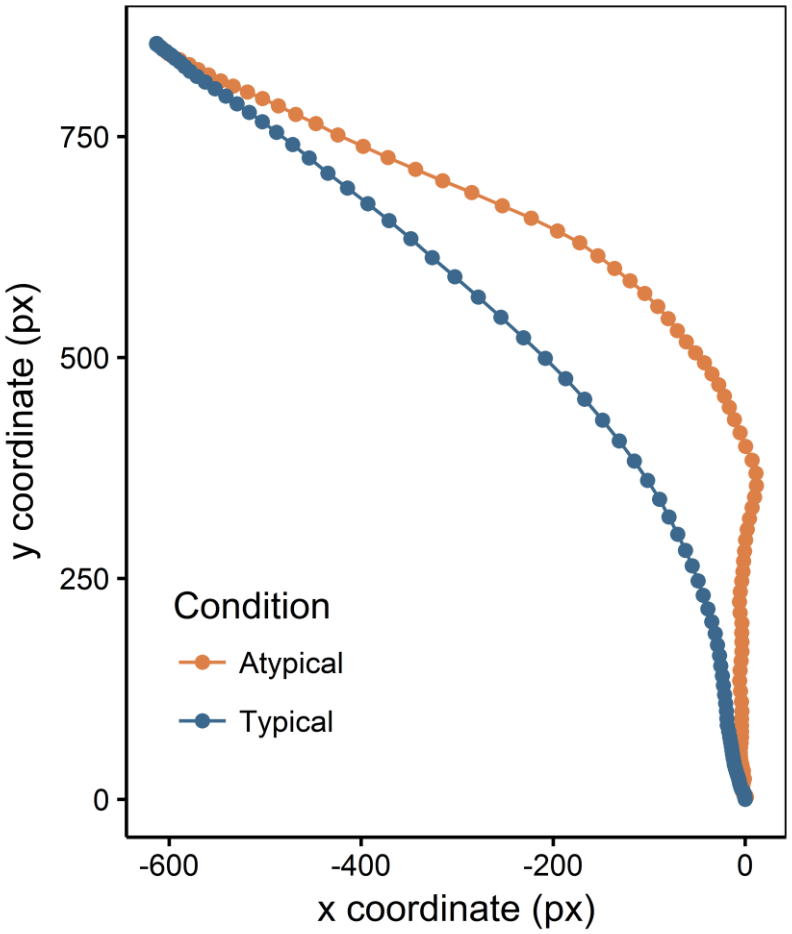
Time-normalized trajectories per condition




```
mt_plot(mt_data, use="tn_trajectories",
        facet_col="Condition", alpha=0.1)
```

Analyzing mouse-tracking data

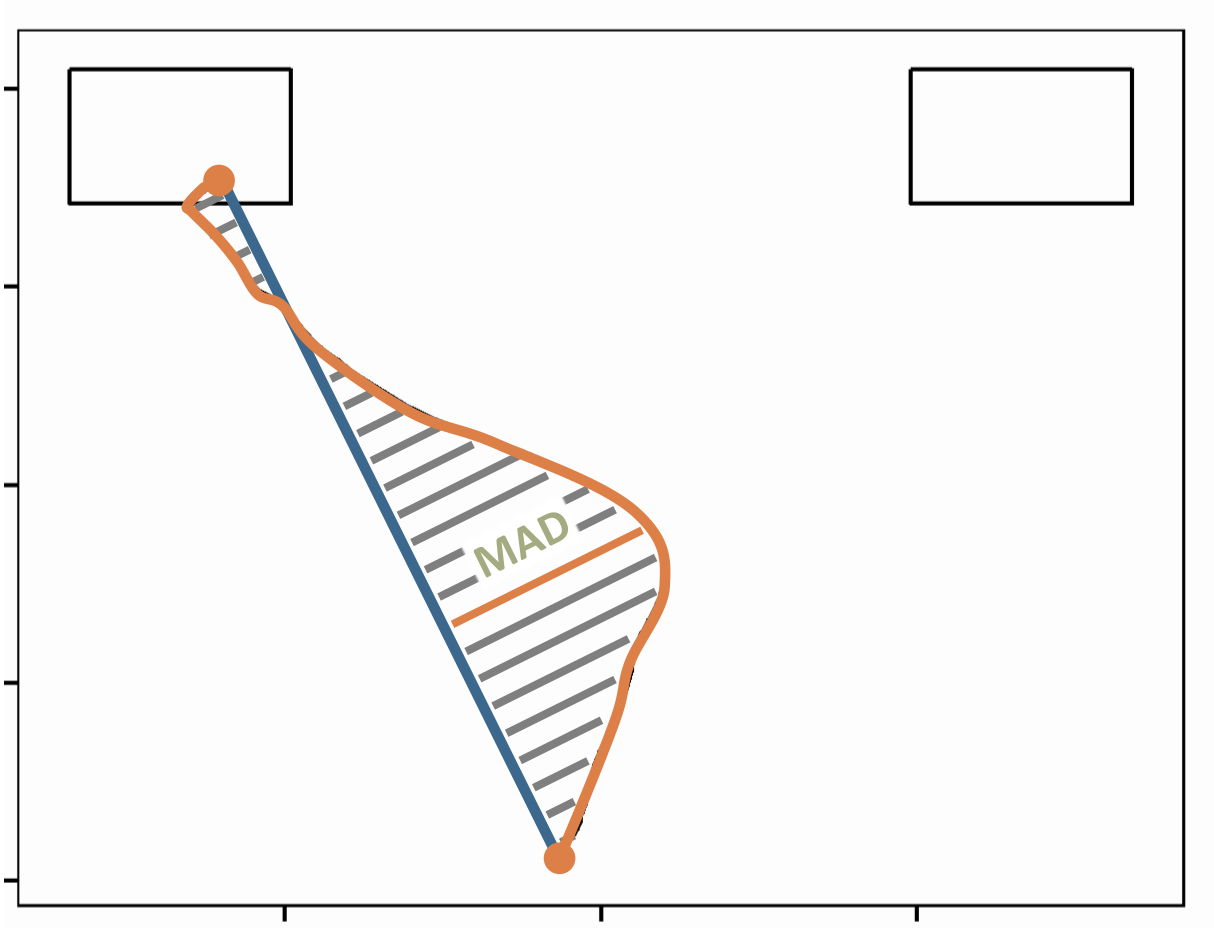
Average time-normalized trajectories




```
mt_plot_aggregate(mt_data, use="tn_trajectories", color="Condition")
```

Analyzing mouse-tracking data

Selected measures for trajectory curvature



Measures of curvature quantify perpendicular distance between **observed trajectory** and an **idealized straight line**

- Maximum absolute deviation (**MAD**)
  McKinstry, Dale, & Spivey (2008)
- Average deviation (**AD**)
  Koop & Johnson (2011)
- Area under curve (**AUC**)
  Spivey, Grosjean, & Knoblich (2005)

Statistical analysis

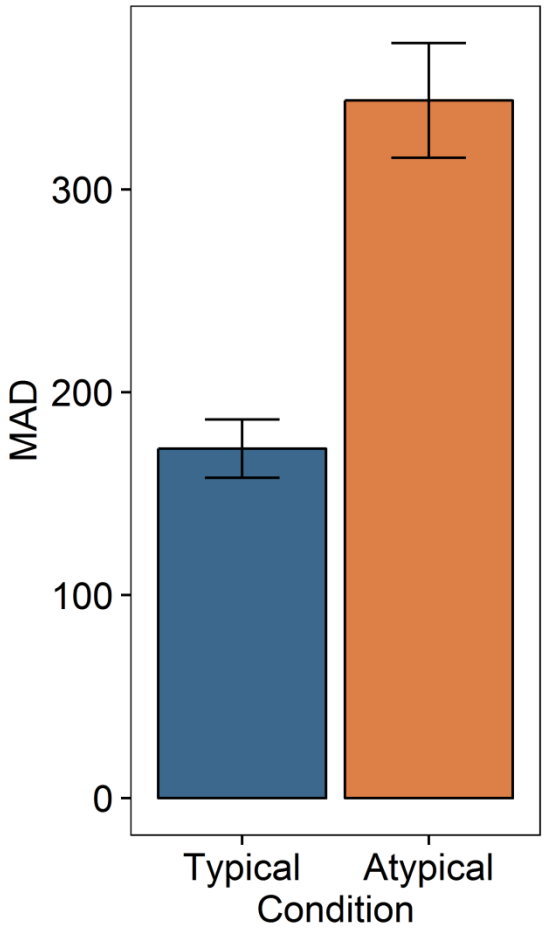
Comparison of maximum absolute deviation

- **MAD larger for atypical exemplars**
  - $t(59) = 7.73$, $p < .001$

```
mt_data <- mt_measures(mt_data)

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

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



Error bars represent 1 *SEM*.

Analyzing mouse-tracking data

R code for main analyses


```r
# library(readbulk)
# raw_data <- read_opensesame("raw_data")

library(mousetrap)
raw_data <- KH2017_raw
raw_data <- subset(raw_data, correct==1)

mt_data <- mt_import_mousetrap(raw_data)
mt_data <- mt_remap_symmetric(mt_data)
mt_data <- mt_align_start(mt_data)
mt_data <- mt_measures(mt_data)

agg_measures <- mt_aggregate_per_subject(mt_data, subject_id = "subject_nr",
  use_variables = "MAD", use2_variables = "Condition")
t.test(MAD~Condition, data = agg_measures, paired = TRUE)

mt_data <- mt_time_normalize(mt_data)
mt_plot_aggregate(mt_data, use = "tn_trajectories", points = TRUE,
  x = "xpos", y = "ypos", color = "Condition", subject_id = "subject_nr")
```

Analyzing mouse-tracking data

Utilizing R's analysis and visualization capabilities

- **Mousetrap** mainly designed for **processing of trajectories** and **computation of measures** (+ provides mouse-tracking specific visualization functions)
  - Once mouse-tracking data are preprocessed and measures have been computed, the resulting data can be merged into a data.frame

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

  - Mouse trajectories (raw, averaged, time-normalized, ..) can be transformed in a format required for the statistical analysis using `mt_export_long` or `mt_export_wide`

- Data can be analyzed and visualized using **general purpose R packages**
  - Basic functions such as `lm` and `glm` for linear and generalized linear models
  - Linear mixed regressions using `lmer` function from **lme4** package
  - ANOVAs with within & between factors using `aov_ez` function from **afex** package
  - Visualizations using **ggplot2** (which is internally used by mousetrap)
  - …

Your data

Analyzing mouse-tracking data

Reading and merging raw data

- General preparation in R / RStudio
  - Create a new project
  - Copy all data into a folder called "raw_data"
  - Create and save a new R script
- Loading package and data
  - Load the readbulk library: `library(readbulk)`
  - Use the read_opensesame function to read in your raw data
    `raw_data <- read_opensesame("raw_data")`
- Explore raw data using different functions
  - Click on the raw_data in RStudio‘s Environment pane
  - Try out `str(raw_data)`
- If you want to filter your data, make use of the subset function
  - ```
    raw_data <- subset(raw_data, Condition!="practice",
      select=c(subject_nr, Condition, response, correct))
    ```
    ```

Analyzing mouse-tracking data
Data import

- To perform analyses on mouse-tracking data they have to be transformed into a **mousetrap** data object first

- As data were collected in OpenSesame via the mousetrap plugin for OpenSesame, you can use the `mt_import_mousetrap` function
  - If only one mousetrap item collected mouse-tracking data in the experiment, it will automatically detect the corresponding variables
  - If more than one mousetrap item collected mouse-tracking data, the mouse-tracking variables need to ne specified via the corresponding variables
  - These allow also that tracking data from several items can be combined

```
mt_data <- mt_import_mousetrap(raw_data,
    xpos_label = c("xpos_initial_phase","xpos_get_response"),
    ypos_label = c("ypos_initial_phase","ypos_get_response"),
    timestamps_label = c("timestamps_initial_phase","timestamps_get_response"))
```

- Task
  - Import your raw data into mousetrap:
    `mt_data <- mt_import_mousetrap(raw_data)`
  - Explore imported data object using `str()` or RStudio's Environment pane

Analyzing mouse-tracking data
Preprocessing and analysis

- Preprocess your data as previously presented
  - Remap and align trajectories
  - Time-normalize trajectories
  - Compute trial-level indices
- Visualize your trajectories using mt_plot and mt_plot_aggregate
- Perform the curvature related analyses that you preregistered for your dataset

Analyzing mouse-tracking data

R code for main analyses


```r
# library(readbulk)
# raw_data <- read_opensesame("raw_data")

library(mousetrap)
raw_data <- KH2017_raw
raw_data <- subset(raw_data, correct==1)

mt_data <- mt_import_mousetrap(raw_data)
mt_data <- mt_remap_symmetric(mt_data)
mt_data <- mt_align_start(mt_data)
mt_data <- mt_measures(mt_data)

agg_measures <- mt_aggregate_per_subject(mt_data, subject_id = "subject_nr",
  use_variables = "MAD", use2_variables = "Condition")
t.test(MAD~Condition, data = agg_measures, paired = TRUE)

mt_data <- mt_time_normalize(mt_data)
mt_plot_aggregate(mt_data, use = "tn_trajectories", points = TRUE,
  x = "xpos", y = "ypos", color = "Condition", subject_id = "subject_nr")
```

Analyzing mouse-tracking data

Custom ggplot2 theme


```
theme_set(theme_classic()+
  theme(
    axis.line = element_line(colour = "black"),
    axis.ticks = element_line(colour = "black"),
    axis.text = element_text(colour = "black"),
    panel.border = element_rect(colour = "black", fill=NA)
  ))
```