## Running the analysis

1. **Verify subjects’ data was recorded properly**
   1. Open ‘/experiment/RUN\_ME/code/’
   2. Open ‘/tests/runTests.m’ and set following params:
      1. Sub\_num
      2. Test\_type – ‘data’
      3. Test\_day – ‘Day2’ (except for exp 3 which also has day 1)
   3. Run runTests.m and verify:
      1. Event duration – stimuli was presented for the appropriate duration.
      2. Has values - all fields in output data contain values (not nan).
      3. Relations – Verifies prime/target/distractor category (natural/artificial) suites the condition (congruent/incongruent) and verifies prime+target or prime+distractor don’t share letters in common locations.
      4. Conditions – each condition appear the appropriate number of times.
      5. Target repetitions – don’t occur within a single block.
      6. Prime right/left alternations – primes appear the same number of times on the left side as on the right side.
      7. Count trials and blocks – matches planned experiment’s length.
   4. All tests should produce ‘1’, except for ‘has values’ in the keyboard session, where values will be missing if participant responded too late.
   5. Tests results are saves in ‘/tests/test\_results/’.
2. **Preprocess subjects’ data**
   1. Open folder ‘/analysis/analysis\_code/’ in MATLAB, then open ‘analysis.m’ , and run following sections:
   2. Parameters – define the following:
      1. – subjects to be analyzed.
   3. Simulate (optional, see ‘SIMULATE’ above)
   4. Create Proc Data File
   5. Add Fields – Adds to the output data of the first subjects three fields (late\_res, slow\_mvmnt, quit) that were recorded only at a later version of the experiment.
   6. Add trials – If subject quit before finishing, fills missing trials with nan.
   7. Preprocess and Normalize –

Taken from *Gallivan, J. P., & Chapman, C. S. (2014). Three-dimensional reach trajectories*.

For more information, see *‘/analysis\_code/imported\_code/craig\_code/Using Functional Data Analysis\_v1\_april2011.pdf*’

* + 1. Fill Missing Data –

Uses [inpaint\_nans](https://www.mathworks.com/matlabcentral/fileexchange/4551-inpaint_nans) to replace NaNs (“camera glitches” in trajectory) with interpolated values.

* + 1. Filter Trajectory – Apply low pass butter worth filter of 2nd order with cutoff at 8Hz on each axis separately to clean any noise in the recording. Since the noise is canceled, we can later over-fit the data without concern for capturing noise with our fit.
    2. Set Origin – Set the first sample in each trial as the axes origin by subtracting it from each sample.

Set the time of the first sample in each trial as t=0.

* + 1. Trim Onset Offset – Trims the trajectory.
       1. Smooth velocity to find onset – Compute velocity and apply low pass butter worth filter of 2nd order to it with a 10Hz cutoff.
       2. Onset - the first sample in which the 3 consecutive samples had a velocity greater than 2cm/s and a total acceleration greater than 2cm/s^2.
       3. Offset - point closest to the screen.
    2. Normalize\_trajs – Fit a B-spline function to the data and extract 200 points equally spaced along the total path traveled on the Z axis. This allows to compare X coordinates between trials.

1. **Trial screening –**

Marks bad trials for these reasons:

* 1. Too early – movement initiation started less than 100ms after target was displayed indicating a predictive response.
  2. Late response – movement didn't start 320ms after target display.
  3. Slow movement – Movement duration was longer than 420 ms.
  4. Very slow movement – movement 3 SD slower than average duration across valid trials.
  5. Short reach distance – distance along the Z axis between first and last sample was shorter than:

*Onset variation* is a 3cm error margin that compensates for small variations in the location of movement

onset

* 1. Too many missing samples – trajectory has more than 100ms of missing samples (usually due to recording issues or obstruction of the marker from the camera's view).
  2. Short sample – Recorded trajectory is shorter than 100ms.
  3. Missed targets – The last sample is further than 12cm from either of the targets (on the X,Y plane).
  4. Bad stimulus duration – the duration of one of the stimuli (fixation, masks, prime, target) deviated by more than 2ms from its designated duration.
  5. Incorrect – target classification was wrong.
  6. Subject didn't perform the trial (quit).
  7. Bad preprocessed length – preprocessed trajectory’s length is different than the one assigned to ‘trim\_len’ in ‘preprocessing’ section in ‘analysis.m’.

1. **Subject screening –**

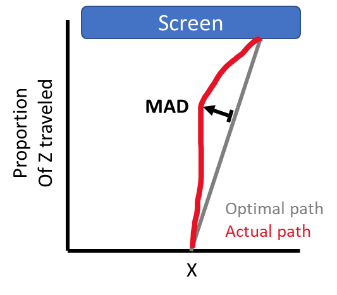
Excludes subjects that fall within the following criteria:

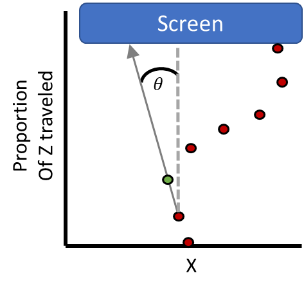
* 1. Not enough trials – The number of valid trials which also had a PAS rating of 1 is smaller than 60.
  2. Not enough trials in each condition – The number of valid trials in each condition (congruent / incongruent) which also had a PAS rating of 1 is smaller than 25.
  3. Bad performance – subject was correct less than 70% of the times in the target classification task (counts only trials with PAS=1).
  4. Aware of prime – prime recognition was above chance indicating the subject is aware of the prime.

1. **Perform feature extraction**

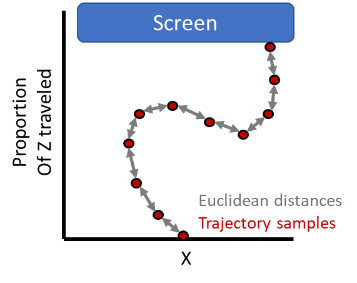
:

(MAD), The furthest distance from the line connecting the start and end points.

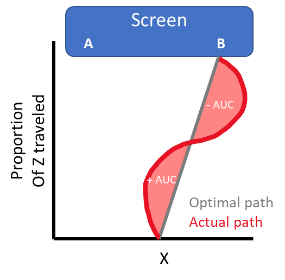




Total distance traveled – sum of Euclidean distances.



AUC



I – intersection between tangent to the trajectory and the screen.

C – number of times the implied end point moves from one half of the screen to the other.

For more information see Pre-reg and manuscript.

1. **Average features within participant**

Averages trials within each participant, excluding invalid trials or trials whose PAS isn't 1.

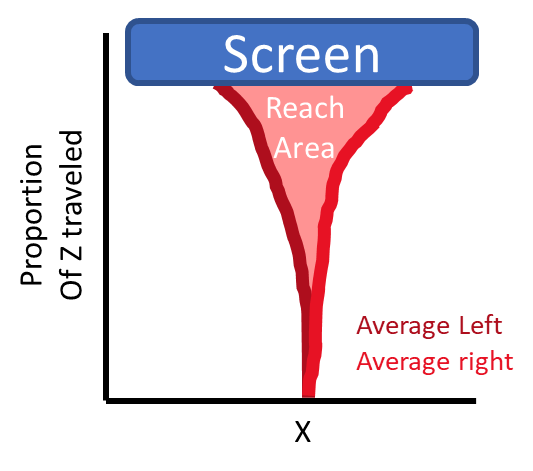
Averages:

* 1. Trajectories
  2. Response time
  3. Reaction time
  4. Movement time
  5. Prime recognition
  6. MAD
  7. Maximally deviating point.
  8. Total STD of the X coordinates – collapses the STD across time.

Computes:

* 1. STD (between trials) of the x coordinates at each point along Z.
  2. Number of trials with each PAS rating.
  3. Difference between average trajectory in the congruent and incongruent conditions.

1. **Extract features that are computed over the average trajectories**
   1. Reach area – Area circumscribed between the average trajectory to the left target and the average trajectory to the right target.



Computes for each condition separately.

Area calculation: Finds minimal X value and draws a line parallel to Z axis at that value.

Computes the area between each trajectory and this line and subtracts the results, producing the area between the trajectories. To avoid negative values after intersections, calculate the area separately between each pair of intersections.

* 1. D’ computation - Uses machine learning to classify trials to each of the conditions. The classification’s accuracy in an indication of the main tasks’ sensitivity to the conditions (see Meyen et al. (2022). Advancing research on unconscious priming: When can scientists claim an indirect task advantage?).
  2. Velocity profile – distribution of velocities at each point along the trajectory.

1. **Average features between subjects.**

Averages the following values (after excluding the bad subjects):

* 1. Trajectory
  2. RT
  3. Reaction time
  4. Movement time
  5. Prime recognition
  6. Number of trials with each PAS rating.
  7. MAD
  8. Maximally deviating point.
  9. Reach area
  10. STD of the x coordinates at each point along Z.
  11. Total STD of X

1. **Conduct FDA-** (Optional)

For explanations about FDA see Craig Chapman’s documentation within

‘/imported\_code/craig\_code/Using Functional Data Analysis\_v1\_april2011.pdf’).

we decided against using this analysis since it wasn’t fruitful.

Here I used subject as a random factor.

getRMMeans – calculates the mean trajectory for each subject at each condition.

Fanovan – runs a repeated measures aNOVA analysis at every point along the trajectory.

1. **Count trials within each condition**
2. **Plotting params section**

Sets all the parameters for plotting, and aggregates all the subjects’ data to a single variable.

When prompted to enter the “number of permutation + clustering analyses” use 2 (trajectory

and implied end point). This will correct p-value for these 2 analyses.

1. **Single sub plots –** (Optional)

makes a separate plot for each subject.

1. **Multiple subs average plots -** (Optional)

makes plots with the average data of all subjects.

1. **Plots for paper**

Neatly Organizes all the plots I needed for the paper. After running this run the next section ‘Add labels to subplot’.

1. **GUI –** (Optional)

Creates a GUI to visually examine the processed trajectory that was fit to each trial.

1. **Movement Time Percentiles -** (Optional

Helps to determine what duration to trim the trajectories to (when NORM\_TRAJ = 0). Shorter trajectories are

excluded from analysis. E.g. we are looking for a movement time that is longer than 10% of the trials (i.e. exclude only 10%). Then we look for the RT of the 10th percentile.

1. **Format to R**

Converts analysis output to R suitable format. Run this before running R scripts.

1. **Examine t-test assumptions with R**

R code can either:

* Create a mixed regression model with ‘side of correct answer’ (left/right) as a random effect.

To do so, uncomment the functions that contain ‘trials’ and ‘coefs’ in their name, and comment those that have ‘avgs’.

* Test assumptions of the statistical tests used in MATLAB, Run permutation tests where they fail, Compute effect sizes for these tests (and the regular t-tests).

To run the code:

* 1. Define the following parameters within main.R:
     1. DAY
     2. SUBS
     3. PICKED\_TRAJS
     4. NORM\_FRAMES
     5. STNDRD - relevant when creating mixed regression models.
     6. RAND\_EFF - relevant when creating mixed regression models.
     7. R/K\_VAR\_NAMES
  2. Run exe\_analysis.R

1. **Tree-BH Correction**

Run this after running the R scripts to get the final significance values of the MATLAB+R analysis.

1. **Classify trials (as congruent/incongruent) with machine learning with Python**

Identical to the ‘D’ computation’ section in MATLAB (see above), however, uses more complex machine learning

methods.

To run the code:

* 1. Define ‘SUBS’ and ‘DAY’ within fit\_n\_pred.m.
  2. Un/comment algorithms you wish to use (Naive base / Ensemble / Gradient boosting).
  3. Run fit\_n\_pred.m.