Pipeline Documentation:

1. Experimental Setup

We used a total of six ChAT-Cre mice. Each mouse underwent surgery to first inject a GCaMP-expressing viral vector, followed by implantation of an optical fiber above the medial septum-diagonal band of Broca (MSDB). This setup allowed us to record population activity of cholinergic neurons using a fiber photometry system. Neural signals were acquired at a sampling rate of 610 Hz using TDT Synapse software. Following recovery, mice were tested in an Object Location Memory (ObLoM) task. Experiments took place in a 40 × 40 cm^2 open-field arena made of black acrylic with 30 cm high walls. A white triangular cue card was affixed to one wall and remained in the same position throughout all experiments to provide a spatial reference. Two identical cylindrical objects (50 mL conical tubes) were placed near adjacent or non-adjacent corners of the arena during the Sample phase. In the Test phase, one of the objects was moved to a novel location. Each mouse completed one/two ObLoM task of 15-minute Sample session followed by a 15-minute Test session, with a 1-hour delay in the home cage between sessions. Mice were tracked from above the arena using a ceiling-mounted camera. The frame rate ranged from 10 to 30 frames per second, and all videos were resampled via Fourier-based interpolation to 30 Hz to standardize temporal resolution across sessions. Synchronization between video and photometry data was achieved using TTL pulses.

2. Preprocessing:

Fiber Photometry Signal processing was performed following the methods described in the manuscript. Briefly, the data recorded using the TDT fiber photometry system were loaded into Python using the load_TDT_data function. Both the calcium-dependent (GCaMP) signal and the isosbestic control signal were resampled to match the video frame rate (30 Hz). To correct for photobleaching and motion artifacts, we subtracted an adjusted control signal from the calcium signal. This adjusted control signal was computed in two main steps; (i) Baseline Fitting: The isosbestic control signal was subtracted from the calcium signal, a second-degree polynomial was fitted to the resulting residual, and the fitted curve was then added back to the original control signal. (ii) Amplitude Optimization: The amplitude of the adjusted control signal was scaled to maximize its correlation with the calcium signal. This was done by solving an optimization problem that minimized the squared error between the calcium signal and the scaled control signal. All of these steps are implemented in the calculate_dfof_TDT function. The output of this function is the ΔF/F signal, which we use as a measure of cholinergic activity in all subsequent analyses.

Movement speed was estimated using position data extracted from DeepLabCut's output. The speed_analysis_new() function processes the raw X and Y coordinates of a tracked body part along with DeepLabCut's likelihood estimates. The sampling rate for all video recordings was standardized to 30 Hz, and a pixel-to-centimeter conversion factor was applied based on the known arena dimensions. Coordinates with low confidence (likelihood < 0.9) were excluded and

linearly interpolated NaN values to ensure continuity in the position signal. The frame-by-frame speed was then computed by measuring the Euclidean distance traveled between consecutive frames and dividing by the time interval. Extremely high values, presumed to reflect tracking errors or noise, were thresholded (e.g., speeds > 150 cm/s) and removed. The final speed signal was saved as a .csv file and used as a continuous behavioral regressor in our subsequent statistical modeling of neural activity.

For behavioral classification, we used DeepEthogram (Bohnslav et al., 2021) to automatically classify mouse behaviors during each recording session. To train the model, the experimenters manually labeled short 2–3-minute segments from each video. These labels were based on a predefined list of behaviors relevant to our study: locomotion, rearing, grooming, and object exploration. Once trained, the model was used to label the rest of the video. The output was a frame-by-frame list showing whether each behavior was present (1) or not (0). This gave us a detailed behavioral profile aligned with the photometry data, which we used for further analysis.

For each recorded session (Sample or Test), a .csv file was generated containing the ΔF/F signal, movement speed, and binary behavioral labels. Each file is stored in a separate folder named according to the experiment's metadata. All processed data files can be found within the all_data directory.

3. Data analysis

In this section, we provide a detailed explanation of the analyses presented in the manuscript.

Each analysis function requires two key inputs: a directory_path pointing to the all_data folder, and a user-defined path for saving the output files (e.g., plots or CSVs). When a function is executed, it loops through all recorded sessions stored in the all_data directory. For each session, it reads the session metadata from two files: detail.txt, which specifies the timing (in seconds) of when the mouse was placed in the maze, and mouse_id.txt, which contains the mouse ID, trial number (1 or 2), and task type (Sample = 0, Test = 1). It then loads the main session data, including ΔF/F signals, movement speed, and behavioral labels, from a corresponding CSV file. This structure allows each function to process all available sessions in a standardized, automated way.

- The function plot_speed_dfof includes three optional parts that can be uncommented depending on the desired analysis. First, you can generate **Figures 2A and 2D**, which show the correlation between movement speed and cholinergic activity. To use the logarithm of movement speed in this plot, set the input variable log = 1. Second, the function scatter_plot_dfof_speed visualizes the relationship between ΔF/F and speed for a given session, as shown in **Figures 2B and 2E**. It optionally smooths both signals and plots every 5th frame, fitting either a logarithmic or linear curve depending on the input. Third, the analysis for **Figures 2C and 2F** using avg_scatter_plot_dfof_speed function examines how ΔF/F varies with speed on average across sessions. For this, the signals are smoothed, speed is log-transformed, and ΔF/F is binned and averaged. The result is plotted across all sessions and separately for Sample and Test conditions. To assess statistical differences

between task types, a linear mixed-effects model is fit with speed, task type, and their interaction as fixed effects, and session ID as a random effect.

- The find_best_smoothing_corr function evaluates how the correlation between movement speed and cholinergic activity (ΔF/F) varies across different smoothing window sizes shown in **Figure 2G**. For each recorded session, the function loads speed and ΔF/F signals, applies a range of temporal smoothing windows (from 0.25 to 256 seconds), and computes the Pearson correlation coefficient between the logarithm of speed and smoothed ΔF/F signal using the corr_speed_dfof function. This allows us to identify the optimal smoothing timescale that maximizes the relationship between movement and cholinergic activity. The results are visualized as session-level and averaged across all sessions.

- The function novel_env analyzes cholinergic activity dynamics at the start of Sample and Test sessions to assess whether elevated activity reflects environmental changes beyond what can be explained by movement speed. It iterates through all session folders, extracts dF/F and speed signals, applies 1-second smoothing, z-scores the dF/F trace, and fits a linear regression model to predict dF/F from log-transformed speed. Predicted and speed-corrected dF/F signals are averaged across sessions and plotted alongside raw dF/F and speed (**Figure 2H–I**). Uncommenting the relevant section allows plotting these signals for individual sessions. A nonparametric cluster-based permutation test compares observed and speed-predicted dF/F, reporting the largest cluster mass, its duration, and p-values. The null distribution is saved as a CSV file, and the number of iterations (default 500) can be modified.

- The function time_analysis_washout generates data for **Figure 3** by quantifying exploration times of stationary and non-stationary objects during Sample and Test sessions. It loops through session folders, extracts object exploration signals from processed CSV files, and segments the data into 3-minute windows starting from the session onset. For each window, it computes the total exploration time for each object and calculate the discrimination index (DI) and stores the results in a DataFrame (wash_time_ratio.csv), where each column represents a session and each row a time window. In addition, the function calculates total exploration times over the full analysis window (first 12 minutes for Sample, first 3 minutes for Test), and computes the normalized difference between them as a DI. These session-level values are saved in summary_time.csv, along with task labels and experiment identifiers. This function supports the analysis shown in Figures 3A–C by enabling time-resolved and total comparisons of object preference behavior across task phases. Remember to change the analysis_win value to 3 minutes for Test sessions and 12 minutes for Sample sessions when generating the summary data.

- The function behavioral_interactions generates the data and visualizations for **Figure 4A and 4B** by quantifying co-occurrence patterns among behaviors during Sample or Test sessions. It iterates through session folders, extracts binarized time series for five behaviors (exploring non-stationary and stationary objects, locomotion, rearing, grooming), and computes pairwise overlap across frames to build a behavioral interaction

matrix for each session. These matrices are saved individually as CSV files, then averaged across sessions and normalized by the sampling rate (30 Hz). The function plots a full heatmap of the average interaction matrix, a heatmap excluding diagonal self-interactions, and a pie chart showing the distribution of time spent in each individual behavior. To switch between Sample and Test analyses, change the task_type condition in the script.

- • The function Create_dataset_for_LLM prepares the full session-level dataset used for linear mixed-effects modeling (LMM, in Matlab code) by aggregating behavioral and ΔF/F data across all Sample and Test sessions. It loops through session folders, extracts metadata (session ID, mouse ID, trial numbr → ObLoM ID), and reads behavioral variables and z-scored ΔF/F signals, applying a 0.5-second moving average and log-transforming speed. For each frame, it creates a row with session ID, task type (Sample Vs Test), time, and six behavioral predictors (exploring objects, locomotion, rearing, grooming). All sessions are concatenated and saved as a single long-format dataframe (all20sessions_allbehaviors.csv), ready for LMM analysis.

- The function cholinergicLevel_diffBehaviors generates a boxplot comparing z-scored cholinergic activity (ΔF/F) across five behaviors—locomotion, grooming, rearing, exploration of stationary and non-stationary objects—during Sample and Test sessions. It filters frames where each behavior is active, collects corresponding ΔF/F values, and organizes them into a long-format DataFrame grouped by behavior and task phase. It then plots the distribution of ΔF/F values for each behavior using side-by-side boxplots for Sample (Task 0) and Test (Task 1) sessions, omitting outliers for clarity. The **figure 5C** provides a summary of behavior-specific cholinergic levels across conditions and is saved in the specified output path.

- The script LLMMatlab.m fits a linear mixed-effects model (LMM) to assess how task phase and behavioral states predict cholinergic activity (ΔF/F) across all sessions. It loads the combined session dataset (all20sessions_allbehaviors.csv), encodes identifiers as categorical variables, and defines a full interaction model with fixed effects for task_phase × (log(speed), object exploration, rearing, grooming), and random intercepts and slopes nested within mouse ID, trial (ObLoM) ID, and session ID. The fitted model (lme) is saved to a .mat file for reproducibility. To evaluate differences in cholinergic activity between exploring non-stationary versus stationary objects during the Test session relative to Sample session, the script performs a linear contrast analysis using a coefficient-based F-test. The result provides a p-value and F-statistic for this key behavioral comparison.

- The time_wash_LLM.m script fits a linear mixed-effects model (LMM) across sliding 3-minute windows to examine how cholinergic activity relates to behavioral states over time (**Figures 4D–E**). Using the combined session dataset (all20sessions_allbehaviors.csv), it loops over overlapping time windows (1-minute step, 3-minute width), filters sessions with at least 1s of both stationary and non-stationary object exploration, and fits a full LMM with fixed effects for task × behaviors and nested random intercepts/slopes for mouse ID, trial (ObLoM) ID, and session ID. Fixed-effect estimates, standard errors, t-statistics, and p-values are extracted from each model, reshaped into wide format, and

saved in LMM_windowed.csv. This analysis captures time-resolved behavioral contributions to cholinergic dynamics throughout Sample and Test sessions.

- The function bar_plot_timewashedLLM visualizes the time-resolved effects of object exploration on cholinergic activity based on linear mixed-effects models fit across 3-minute sliding windows. It loads precomputed estimates and standard errors of fixed effects from LMM_windowed.csv, then plots coefficients of exploration of non-stationary vs. stationary objects separately for Sample and Test sessions (**Figure 4D–E)**. It also computes and plots the difference between these two conditions across time. To statistically evaluate whether this difference varies by task phase, the function fits an ordinary least squares (OLS) model with task_phase and time_window.
- The function strech_time_Behaviors_only_4_statistics performs a detailed event-triggered, time-warped analysis of cholinergic activity (ΔF/F), predicted activity from movement speed, and speed itself, aligned to specific behavioral bouts such as locomotion, grooming, rearing, and object exploration.

Inputs**:**

- name_behavior (str)
  Behavior to analyze: "behavior_walking" (locomotion), "behavior_grooming", "behavior_rearings", "behavior_exp_statobj", or "behavior_exp_non_statobj".
- whichTask (str)
  Task phase to include: "Sample", "Test", or "both".
- mouseid (int)
  Specific mouse ID to include in the analysis (values from 1 to 6), or 0 to include all mice**.**
- trialid (int)
  ObLoM task: 1, 2, or 0 to include both Object Location Memory (ObLoM) tasks.
- num_iters_clusterTest (int)
  Number of iterations for the nonparametric cluster-based permutation test**.**

For each session that meets the task and mouse criteria, the function loads the relevant behavioral and cholinergic signals, smooths the ΔF/F and speed traces using a 0.5-second moving average, z-scores the ΔF/F, and fits a linear regression model to predict ΔF/F from log-transformed speed. Using this model, it generates predicted cholinergic activity to later compare against observed activity.

The function then detects valid bouts of the selected behavior and aligns the ΔF/F, predicted ΔF/F, and speed signals around those bouts, extracting a window from 5 seconds before the behavior to 5 seconds after its offset. It also removes any short bouts (<2 s) and excluded bouts if the same behavior occurred within 4 s before the onset or after the offset of the bout.

The middle segment (i.e., the behavior duration) is linearly stretched to a fixed length using interpolation, allowing comparison across bouts of varying durations. Simultaneously, it extracts matching baseline segments from non-overlapping periods for

comparison. It also captures time-warped co-occurring behaviors such as locomotion, grooming, rearing, and object exploration, as well as non-behavioral background activity. These signals are aggregated across sessions, and the function computes the average and standard error of the mean (SEM) for each signal during pre-, during-, and post-behavior windows. To visualize the behavioral context, the function generates plots showing the number and percentage of co-occurring behaviors across the aligned timepoints, helping verify whether the analyzed behavior occurred in isolation. It then merges the time-warped signals into single traces for each of ΔF/F, predicted ΔF/F, and speed, and overlays them in a final plot with dual y-axes—one for speed, and one for ΔF/F and predicted ΔF/F—along with vertical lines marking behavior onset and offset.

Finally, to statistically compare the observed and predicted ΔF/F signals during behavior, the function applies a cluster-based permutation test across the aligned windows. It computes the t-statistic time series between the two signals, identifies the largest contiguous cluster of significance, and generates a null distribution by randomly permuting the signal labels across events for the specified number of iterations (you can set that using num_iters_clusterTest input). It reports whether the observed cluster is statistically significant, along with its size, mass, and the percentage of the behavior duration showing a significant difference. All plots and permutation test results are saved to the specified output path. This function was used to generate **Figure 5** in the manuscript.

Outputs:

- behaviors_percentage_behavior_"name".svg
  A plot showing the *proportion of total behaviors* over time, computed as the ratio between the occurrence of the specified behavior at each time point and the sum of all behaviors at that time.
- main_behavior_"name".svg
  Time-aligned visualization of behavioral events across sessions, displaying:
  - movement speed (black),
  - observed cholinergic activity (ΔF/F, green),
  - predicted cholinergic activity from log(speed) (red),
    spanning a window of 5 seconds before onset, during the behavior, and 5 seconds after offset.
- main_signals_behavior_"name".svg
  Same as above but restricted to the *duration of the behavior* (onset to offset), highlighting only the ΔF/F signals.
- Clusters_behavior_"name"*Avg_dfof_VS_Avg_pdfof*"date_time".csv
  A CSV file containing cluster-based permutation test results comparing:
  - Avg_dfof: mean observed ΔF/F during the behavior
  - Avg_pdfof: mean predicted ΔF/F based on speed
    Includes the *mass* and *size* of the largest clusters found under the null distribution.

    - nullDis_Mass.svg
      Histogram of the null distribution of *maximum cluster mass values*, used to assess statistical significance of observed effects.
    - nullDis_size.svg
      Histogram of the null distribution of *maximum cluster sizes*, supporting evaluation of observed cluster size significance.

- The stat_cluster_test function evaluates the statistical significance of observed cluster mass and cluster size by comparing them to null distributions obtained from a permutation test. It takes as input the observed cluster mass and size values, a path to save output plots, and a CSV file containing the null distribution of cluster statistics from shuffled data. The function first loads the null distribution of the largest cluster mass (largestMass_cluster_obs), computes its absolute values, and plots a histogram saved as nullDis_Mass.svg. It then calculates a Monte Carlo p-value by determining the proportion of null mass values greater than or equal to the observed value. The same procedure is repeated for the largest cluster size (largestSize_cluster_obs), producing a second histogram (nullDis_size.svg) and p-value. These outputs help determine whether the observed effects are statistically significant compared to chance.
- The function strech_time_Behaviors_only_4_increaseTime is designed to quantify the temporal dynamics of cholinergic activity (ΔF/F) in relation to the onset or offset of specific behaviors (e.g., grooming, walking) across multiple sessions. The proceeeing it similar to function strech_time_Behaviors_only_4_statistics beside that this new function then calculates the rise or decay timing by applying a threshold—defined as the baseline plus 98% of the activity change—and finds when the signal crosses this threshold. For each bout, the median of those crossing time points is used as an estimate of response timing. These estimates are aggregated and saved for both the observed and predicted signals.

  Inputs:

    - name_behavior *(str)*
      Behavior to analyze: "behavior_walking", "behavior_grooming", "behavior_rearings", "behavior_exp_statobj", or "behavior_exp_non_statobj".
    - whichTask *(str)*
      Task phase: "Sample", "Test", or "both".
    - StartOrEnd *(bool)*
      True for rise time (relative to onset), False for decay time (relative to offset).
    - activityThreshold *(float)*
      Threshold multiplier (e.g., 0.98) for detecting meaningful signal changes.
    - considered_seconds *(int)*
      Time window (in seconds) before or after onset/offset to search for threshold crossings.

Outputs:

- CSV file time_<activityThreshold><name_behavior><StartOrEnd><considered_seconds>.csv
  - Columns: dfof (observed), pdfof (predicted) response times.
- Histograms
  - Distribution of rise/decay times for dfof and pdfof.
- Behavior plots
  - behaviors_numbers<name_behavior>.svg
  - behaviors_percentage<name_behavior>.svg
- Signal plot
  - main_signals<name_behavior>.svg showing speed, observed ΔF/F, and predicted ΔF/F with SEM.