

Python Documentation

version

October 21, 2020

Contents

Welcome to moseq2-app's documentation!	1
moseq2_app package	1
Main Module	1
General Utilities Module	2
Subpackages	2
moseq2_app.flip package	2
Submodules	2
Flip - Controller Module	2
Flip - Widgets Module	3
Module contents	4
moseq2_app.roi package	4
Submodules	4
ROI - Controller Module	4
ROI - Validation Module	6
ROI - View Module	9
ROI - Widgets Module	9
Module contents	9
moseq2_app.gui package	9
Submodules	9
GUI - Progress Module	9
GUI - Widgets Module	11
GUI - Wrappers Module	11
Module contents	13
moseq2_app.stat package	13
Submodules	13
Stats - Controller Module	13
Stats - View Module	15
Stats - Widgets Module	17
Module contents	17
moseq2_app.viz package	17
Submodules	17
Viz - Controller Module	17
Viz - View Module	19
Viz - Widgets Module	20
Module contents	20
Index	20
Index	21
Python Module Index	25

Welcome to moseq2-app's documentation!

moseq2_app package

Main Module

Main functions that facilitate all jupyter notebook functionality. All functions will call a wrapper function to handle any non front-end settings.

`moseq2_app.main.flip_classifier_tool` (input_dir, output_file, max_frames=1000000.0, tail_filter_iters=1, space_filter_size=3)

Flip Classifier Notebook main functionality access point.

Parameters:

- **input_dir (str)** (*Path to base directory containing extraction session folders*)
- **max_frames (int)** (*Maximum number of frames to include in the dataset.*)
- **output_file (str)** (*Path to save the outputted flip classifier.*)
- **tail_filter_iters (int)** (*Number of tail filtering iterations*)
- **prefilter_kernel_size (int)** (*Size of the median spatial filter.*)

Returns: `flip_obj (FlipRangeTool)` – notebook.

Return type: Object that holds all saved interactive functionality and data to be used throughout the

`moseq2_app.main.interactive_crowd_movie_comparison` (progress_paths, group_movie_dir, get_pdfs=True)

Interactive crowd movie/position heatmap comparison function. Launched via the notebook.

Parameters:

- **progress_paths (dict)** (*dictionary of notebook progress paths.*)
- **group_movie_dir (str)** (*path to generate new grouped crowd movies in.*)
- **get_pdfs (bool)** (*indicates whether to also generate position heatmaps.*)

`moseq2_app.main.interactive_group_setting` (index_file)

Interactive group setting wrapper function that displays a excel-like table to update the current group selection

Parameters: `index_file (str)` (*Path to index file to update.*)

`moseq2_app.main.interactive_roi_detector` (input_dir, progress_paths, compute_all_bgs=True)

Function to launch ROI detector interactive GUI in jupyter notebook

Parameters:

- **input_dir (str)** (*path to parent directory containing session folders*)
- **progress_paths (dict)** (*dictionary of notebook progress paths.*)
- **compute_all_bgs (bool)** (*if True, computes all the sessions' background images to speed up the UI.*)

`moseq2_app.main.interactive_syllable_stats` (progress_paths, max_syllable=None)

Generates the interactive syllable statistics viewer, consisting of a dot-line plot and a dendrogram.

Parameters:

- **progress_paths (dict)** (*dictionary of notebook progress paths.*)
- **max_syllables (int or None)** (*manual maximum number of syllables to label.*)

`moseq2_app.main.interactive_transition_graph` (progress_paths, max_syllables=None)

Displays group transition graphs with a configurable number of syllables. Launched via the the jupyter notebook.

Parameters:

- **progress_paths (dict)** (*dictionary of notebook progress paths.*)
- **max_syllables (int or None)** (*manual maximum number of syllables to label.*)

Welcome to moseq2-app's documentation!

`moseq2_app.main.label_syllables` (progress_paths, max_syllables=None, n_explained=99)
Interactive syllable labeling tool accessible from the jupyter notebook.

Parameters:

- **progress_paths (dict)** (*dictionary of notebook progress paths.*)
- **max_syllables (int or None)** (*manual maximum number of syllables to label.*)
- **n_explained (int)** (*Percentage of explained variance to use to compute max_syllables to compute.*)

`moseq2_app.main.preview_extractions` (input_dir)
Function to launch a dynamic video loader that displays extraction session mp4s.

Parameters: **input_dir (str)** (*Path to parent directory containing extracted sessions folders*)

`moseq2_app.main.validate_extractions` (input_dir)

Wrapper function that facilitates the extraction validation step from *main.py*.

Prints all the flagged session outlier details.

Parameters: **input_dir (str)** (*Path to parent directory containing extracted sessions folders*)

`moseq2_app.main.view_extraction` (extractions, default=0)
Prompts user to select which extracted video(s) to preview.

Parameters:

- **extractions (list)** (*list of paths to all extracted avi videos.*)
- **default (int)** (*index of the default extraction to display*)

Returns: **extractions (list)**

Return type: list of selected extractions.

General Utilities Module

General utility functions.

`moseq2_app.util.index_to_dataframe` (index_path)
Reads the index file into a dictionary and converts it into an editable DataFrame.

Parameters: **index_path (str)** (*Path to index file*)

Returns: **index_data (dict)** (*Dict object containing all parsed index file contents*) **df (pd.DataFrame)** (*Formatted dict in DataFrame form including each session's metadata*)

Subpackages

moseq2_app.flip package

Submodules

Flip - Controller Module

`class moseq2_app.flip.controller.FlipRangeTool` (input_dir, max_frames, output_file, tail_filter_iters, prefilter_kernel_size)

Bases: `moseq2_app.flip.widgets.FlipClassifierWidgets`

augment_dataset ()

Augments the dataset with 90 degree rotated versions of the truth values, and creates the X and Y train/test sets.

clear_on_click (b)

Clears the output.

Parameters: **b (button click)**

get_corrected_data ()

Apply the selected flip orientation ranges to the entire dataset to correct the
incorrectly oriented frames.

interactive_launch_frame_selector (num)

Interactive tool that displays the frame to display with the selected data box. Users will use the start_range button to add frame ranges to the range box list.

Parameters: **num (int or ipywidgets.IntSlider)** (*Currently displayed frame number.*)

load_sessions (input_dir, max_frames=1000000.0, tail_filter_iters=1, space_filter_size=3)

Recursively searches for completed h5 extraction files, and loads total_frames=max_frames to
include in the total dataset. Additionally applies some image filtering prior to returning the data.

Parameters:

- **input_dir (str)** (*Path to input directory*)
- **max_frames (int)** (*Maximum number of frames to included in loaded dataset.*)
- **tail_filter_iters (int)** (*Number of tail filtering iterations to run.*)
- **space_filter_size (int)** (*Size of median filtering kernel.*)

Returns: **clean_merged_data (3D np.ndarray)**

Return type: Loaded and filtered data to get correctly oriented frame ranges from.

prepare_datasets (test_size)

Correct data after the appropriate flip ranges have been selected, augment and create X,y training sets,
and split the data to training and testing splits.

Parameters: **test_size (int)** (*Test dataset percent split size*)

start_stop_frame_range (b)

Callback function that triggers the “Add Range” functionality.

If user clicks the button == 'Start Range', then the function will start including frames in the correct flip set. Else, it will end the included range and truncate the slider range from the start index.

Parameters: **b (button click)**

train_and_evaluate_model ()

Trains the flip classifier and prints its performance accuracy on the test set.

train_and_save_model ()

Trains the model on the entire dataset and saves the model to the desired output file path.

Flip - Widgets Module

```
class moseq2_app.flip.widgets.FlipClassifierWidgets
```

Bases: **object**

Module contents

moseq2_app.roi package

Submodules

ROI - Controller Module

Interactive ROI detection and extraction preview functionalities. This module utilizes the widgets from the widgets.py file to facilitate the real-time interaction.

```
class moseq2_app.roi.controller.InteractiveExtractionViewer (data_path)
```

Bases: `object`

`clear_on_click (b)`

Clears the cell output

Parameters: `b (button click)`

`get_extraction (input_file)`

Returns a div containing a video object to display.

Parameters: `input_file (str)` (*Path to session extraction video to view.*)

```
class moseq2_app.roi.controller.InteractiveFindRoi (data_path, config_file, session_config, compute_bgs=True)
```

Bases: `moseq2_app.roi.widgets.InteractiveROIWidgets`

`check_all_sessions (b)`

Callback function to run the ROI area comparison test on all the existing sessions. Saving their individual session parameter sets in the session_parameters dict in the process. Additionally updates the button styles to reflect the outcome.

Parameters: `b (button event)` (*User click*)

`clear_on_click (b)`

Clears the cell output

Parameters: `b (button click)` (*User clicks Clear Button.*)

`compute_all_bgs ()`

Computes all the background images before displaying the app to speed up user interaction.

`extract_button_clicked (b)`

Updates the true depth autodetection parameter

such that the true depth is autodetected for each found session

Parameters: `b (ipywidgets Button)` (*Button click event.*)

`get_extraction (input_file, bground_im, roi)`

Extracts selected frame range (with the currently set session parameters) and displays the extraction as a Bokeh HTML-embedded div.

Parameters:

- `input_file (str)` (*Path to session to extract*)
- `config_data (dict)` (*Extraction configuration parameters.*)
- `bground_im (2D np.array)` (*Computed session background.*)
- `roi (2D np.array)` (*Computed Region of interest array to mask bground_im with.*)

get_pixels_per_metric (pixel_width)

Helper function that computes a pixels_per_metric value, and handles
cases without user input.

Parameters: pixel_width (int) (*width of the ROI bounding box in pixels*)
Returns: pixels_per_inch (float)
Return type: Computed ratio of real life

get_roi_and_depths (bground_im, session)

Performs bucket centroid estimation to find the coordinates to use as the true depth value. The true depth will be used to estimate the background depth_range, then it will update the widget values in real time.

Parameters:

- bground_im (2D np.array) (*Computed session background*)
- session (str) (*path to currently processed session*)
- config_data (dict) (*Extraction configuration parameters*)

Returns: results (dict)
Return type: dict that contains computed information. E.g. its ROI, and if it was flagged.

get_selected_session (event)

Updates the selected value in checked_list, triggering the view to update with the currently
selected session's data.

Parameters: event (ipywidgets event) (*Current value of checked_list has changed*)

interactive_depth_finder (session, bground_im, dr, di)

Interactive helper function that updates that views whenever the depth range or dilation iterations sliders are changed. At initial launch, it will auto-detect the depth estimation, then it will preserve the parameters across session changes.

Parameters:

- session (str or ipywidget DropdownMenu) (*path to input file*)
- bground_im (2D np.array) (*Computed session background*)
- config_data (dict) (*Extraction configuration parameters*)
- dr (tuple or ipywidget IntRangeSlider) (*Depth range to capture*)
- di (int or ipywidget IntSlider) (*Dilation iterations*)

interactive_find_roi_session_selector (session)

First function that is called to find the current selected session's background and display the widget interface.

Parameters:

- session (str or ipywidget DropdownMenu) (*path to chosen session.*)
- config_data (dict) (*ROI/Extraction configuration parameters*)

mark_passing_button_clicked (b)

Callback function that sets the current session as Passing. The indicator will still remain Flagged if the segmented frame or the crop-rotated frame are not appearing.

Parameters: b (ipywidgets event) (*Button click*)

prepare_data_to_plot (input_file, bground_im, roi, minmax_heights, fn)

Helper function that generates the display plots with the currently selected parameters,
and checks if the min-max height parameters are acceptable, updating the success indicator if any issues arise.

Parameters:

- **input_file (str)** (*Path to currently processed depth file*)
- **bground_im (2D np.ndarray)** (*Median depth image of the read video.*)
- **roi (2D Boolean np.ndarray)** (*Computed ROI representing the bucket floor*)
- **minmax_heights (2-tuple)** (*Min and max mouse heights to threshold from background subtracted image.*)
- **fn (int)** (*Frame index to display.*)

save_clicked (b)

Callback function to save the current session_parameters dict into the file of their choice (given in the top-most wrapper function).

Parameters: **b (button event)** (*User click*)

test_all_sessions (session_dict)

Helper function to test the current configurable UI values on all the sessions that were found.

Parameters:

- **session_dict (dict)** (*dict of session directory names paired with their absolute paths.*)
- **config_data (dict)** (*ROI/Extraction configuration parameters.*)

Returns: **all_results (dict)** (*dict of session names and values used to indicate if a session was flagged,) with their computed ROI for convenience.*

update_checked_list (results)

Helper function to update the session selector passing indicators when a parameter test is run.

Parameters: **results (dict)** (*ROI detection results dict containing the flag and return code to display.*)

update_config_fr (event)

Callback function to update config dict with current UI depth range values

Parameters: **event (ipywidget callback)** (*Any user interaction.*)

update_minmax_config (event)

Callback function to update config dict with current UI min/max height range values

Parameters: **event (ipywidget callback)** (*Any user interaction.*)

ROI - Validation Module

The module contains extraction validation functions that test extractions' scalar values, timestamps, and position heatmaps.

`moseq2_app.roi.validation.check_timestamp_error_percentage (timestamps, fps)`

<https://www.mathworks.com/help/imaq/examples/determining-the-rate-of-acquisition.html>

Returns the proportion of dropped frames relative to the respective recorded timestamps and frames per second.

Parameters:

- **timestamps (1D np.array)** (*Session's recorded timestamp array.*)
- **fps (int)** (*Frames per second*)

Returns: **percentError (float)**

Return type: Percentage of frames that were dropped/missed during acquisition.

`moseq2_app.roi.validation.compute_kl_divergences (pdfs, groups, sessions, sessionNames, oob=False)`

Computes KL divergence for all sessions and returns the divergences Consider trying Jensen Shannon or Wasserstein instead!!

Parameters:

- **pdfs (list)** (*list of 2d probability density functions (heatmaps) describing mouse position.*)
- **groups (list)** (*list of groups corresponding to the pdfs indices*)
- **sessions (list)** (*list of sessions corresponding to the pdfs indices*)
- **sessionNames (list)** (*list of sessionNames corresponding to the pdfs indices*)
- **oob (boolean)** (*Compute out-of-bag KL-divergences*)

Returns: **kl_divergences (pd.DataFrame)**

Return type: dataframe with mouse group, session, subjectname, and kl divergence

`moseq2_app.roi.validation.count_frames_with_small_areas (scalar_df)`

Counts the number of frames where the mouse area is smaller than 2 standard deviations of all mouse areas.

Parameters: **scalar_df (pd.DataFrame)** (*Computed Scalar DataFrame*)

Returns: **corrupt_frames (int)**

Return type: Number of frames where the recorded mouse area is too small

`moseq2_app.roi.validation.count_missing_mouse_frames (scalar_df)`

Counts the number of frames where the mouse is not found.

Parameters: **scalar_df (pd.DataFrame)** (*Computed Scalar DataFrame*)

Returns: **missing_mouse_frames (int)**

Return type: Number of frames with recorded mouse area $\sim= 0$

`moseq2_app.roi.validation.count_nan_rows (scalar_df)`

Counts the number of rows with NaN scalar values.

Parameters: **scalar_df (pd.DataFrame)** (*Computed Scalar DataFrame*)

Returns: **n_missing_frames (int)**

Return type: Number of frames with NaN computed scalar values.

`moseq2_app.roi.validation.count_stationary_frames (scalar_df)`

Counts the number of frames where mouse is not moving.

Parameters: **scalar_df (pd.DataFrame)** (*Computed Scalar DataFrame*)

Returns: **motionless_frames (int)**

Return type: Number of frames where the mouse is not moving

`moseq2_app.roi.validation.get_anomaly_dict (scalar_df, status_dicts)`

Helper function that prepares a dictionary with all the relevant values to print at the end of the validation wrapper.

Parameters:

- **scalar_df (pd.DataFrame)** (*Computed Scalar DataFrame*)
- **status_dicts (dict)** (*stacked dictionary object containing all the sessions' flag status dicts.*)

Returns: **anomaly_dict (dict)**

Return type: Dict object containing specific session flags to display

`moseq2_app.roi.validation.get_iqr_anomaly_sessions (scalar_df, status_dicts)`

Finds sessions that have a mean scalar value (for a subset of scalars), that are outside of the accepted inter-quartile range.

Parameters:

- **scalar_df (pd.DataFrame)** (*Computed Scalar DataFrame*)
- **status_dicts (dict)** (*stacked dictionary object containing all the sessions' flag status dicts.*)

Returns: **status_dicts (dict)**

Return type: stacked dictionary object containing updated scalar_anomaly flags.

`moseq2_app.roi.validation.get_kl_divergence_outliers(kl_divergences)`

Returns the position PDFs that are over 2 standard deviations away from the mean position divergence.

Parameters: **kl_divergences (pd.DataFrame)** (*dataframe with group, session, subjectName, and divergence*)

Returns: **outliers (pd.DataFrame)**

Return type: dataframe of outlier sessions

`moseq2_app.roi.validation.get_scalar_df(path_dict)`

Computes a scalar dataframe that contains all the extracted sessions

recorded scalar values along with their metadata.

Parameters: **path_dict (dict)** (*dictionary of session folder names paired with their extraction paths*)

Returns: **scalar_df (pd.DataFrame)**

Return type: DataFrame containing loaded scalar info from each h5 extraction file.

`moseq2_app.roi.validation.make_session_status_dicts(paths)`

Returns the flag status dicts for all the found completed extracted sessions. Additionally performs
dropped frames test on all sessions.

Parameters: **paths (dict)** (*path dict of session names paired with their mp4 paths.*)

Returns: **status_dicts (dict)**

Return type: stacked dictionary object containing all the sessions' flag status dicts.

`moseq2_app.roi.validation.plot_heatmap(heatmap, title)`

Plots and displays outlier heatmap with the SessionName as the title.

Parameters:

- **heatmap (2d np.array)** (*outlier position PDF*)
- **title (str)** (*plot title*)

`moseq2_app.roi.validation.print_validation_results(anomaly_dict)`

Displays all the outlier sessions flag names and values. Additionally plots the flagged
position heatmap.

Parameters: **anomaly_dict (dict)** (*Dict object containing specific session flags to print*)

`moseq2_app.roi.validation.run_heatmap_kl_divergence_test(scalar_df, status_dicts)`

Finds the position PDF outlier sessions and updates the status_dicts with the respective position heatmap flag.

Parameters:

- **scalar_df (pd.DataFrame)** (*Computed Scalar DataFrame*)
- **status_dicts (dict)** (*stacked dictionary object containing all the sessions' flag status dicts.*)

Returns: **status_dicts (dict)**

Return type: stacked dictionary object containing updated position_heatmap flags.

`moseq2_app.roi.validation.run_validation_tests(scalar_df, status_dicts)`

Main function that runs all the available extraction validation tests and updates the status_dicts
flags accordingly.

Parameters:

- **scalar_df (pd.DataFrame)** (*Computed Scalar DataFrame*)
- **status_dicts (dict)** (*stacked dictionary object containing all the sessions' flag status dicts.*)

Returns: **status_dicts (dict)**

Return type: stacked dictionary object containing all the sessions' updated flag status dicts.

ROI - View Module

Interactive ROI/Extraction Bokeh visualization functions.

`moseq2_app.roi.view.bokeh_plot_helper` (bk_fig, image)

Helper function that creates the Bokeh image glyphs in the created canvases/figures.

Parameters:

- **bk_fig (Bokeh figure)** (*figure canvas to draw image/glyph on*)
- **image (2D np.array)** (*image to draw.*)

`moseq2_app.roi.view.plot_roi_results` (sessionName, bground_im, roi, overlay, filtered_frames, depth_frames, fn)

Main ROI plotting function that uses Bokeh to facilitate 3 interactive plots. Plots the background image, and an axis-connected plot of the ROI, and an independent plot of the thresholded background subtracted segmented image.

Parameters:

- **sessionName (str)** (*Name of session currently being plotted*)
- **bground_im (2D np.array)** (*Computed session background*)
- **roi (2D np.array)** (*Computed ROI based on given depth ranges*)
- **overlay (2D np.array)**
- **filtered_frames (2D np.array)**
- **depth_frames (2D np.array)**
- **fn (int or ipywidget IntSlider)** (*Current frame number to display*)

`moseq2_app.roi.view.show_extraction` (input_file, video_file)

Visualization helper function to display manually triggered extraction. Function will facilitate visualization through creating a HTML div to display in a jupyter notebook or web page.

Parameters:

- **input_file (str)** (*session name to display.*)
- **video_file (str)** (*path to video to display*)

ROI - Widgets Module

Ipywidgets used to facilitate interactive ROI detection

`class moseq2_app.roi.widgets.InteractiveROIWidgets`

Bases: `object`

Class that contains Ipywidget widgets and layouts to facilitate interactive ROI finding functionality. This class is extended by the controller class InteractiveFindRoi.

Module contents

`moseq2_app.gui` package

Submodules

GUI - Progress Module

This module handles all jupyter notebook progress related functionalities.

`moseq2_app.gui.progress.check_progress` (base_dir, progress_filepath)

Checks whether progress file exists and prompts user input on whether to overwrite, load old, or generate a new one.

Parameters:

- **base_dir (str)** (*path to directory to create/find progress file*)
- **progress_filepath (str)** (*path to progress filename*)

Returns:

Return type: All restored variables or None.

`moseq2_app.gui.progress.count_total_found_items(i_dict)`

Counts the total number of found progress items

Parameters: `i_dict (dict)` (*Dict containing paths to respective pipelines items.*)

Returns: `num_files (int)`

Return type: Number of found previously computed paths.

`moseq2_app.gui.progress.generate_missing_metadata(sess_dir, sess_name)`

Generates default metadata.json file for session that does not already include one.

Parameters:

- **sess_dir (str)** (*Path to directory to create metadata.json file in.*)
- **sess_name (str)** (*Name of the directory to set the metadata SessionName.*)

`moseq2_app.gui.progress.get_extraction_progress(progress_vars)`

Counts the number of fully extracted sessions, and prints the session directory names
of the incomplete or missing extractions.

Parameters: `progress_vars (dict)` (*notebook progress dict.*)

Returns: `path_dict (dict)` (*Dict with paths to all found sessions*) `num_extracted (int)` (*Total number of completed extractions*)

`moseq2_app.gui.progress.get_pca_progress(progress_vars, pca_progress)`

Updates the PCA progress dict variables and prints the names of the missing keys.

Parameters:

- **progress_vars (dict)** (*Notebook progress dict including the relevant PCA paths*)
- **pca_progress (dict)** (*PCA progress boolean dict used to display progress bar*)

Returns: `pca_progress (dict)`

Return type: Updated PCA progress boolean dict.

`moseq2_app.gui.progress.get_session_paths(data_dir, extracted=False, exts=['dat', 'mkv', 'avi'])`

Find all depth recording sessions and their paths (with given extensions) to work on given base directory.

Parameters:

- **data_dir (str)** (*path to directory containing all session folders.*)
- **exts (list)** (*list of depth file extensions to search for.*)

Returns: `path_dict (dict)`

Return type: session directory name keys pair with their respective absolute paths.

`moseq2_app.gui.progress.handle_progress_restore_input(base_progress_vars, progress_filepath)`

Helper function that handles user input for restoring progress variables.

Parameters:

- **base_progress_vars (dict)** (*dict of default progress name to path pairs.*)
- **progress_filepath (str)** (*path to progress filename*)

Returns: `progress_vars (dict)`

Return type: loaded progress variables

`moseq2_app.gui.progress.print_progress(progress_vars)`

Searches for all the paths included in the progress file and displays 4 progress bars, one for each pipeline step.
Displays tqdm progress bars checking a users jupyter notebook progress.

Parameters: `progress_vars (dict)` (*notebook progress dict*)

`moseq2_app.gui.progress.restore_progress_vars(progress_file)`

Restore all saved progress variables to Jupyter Notebook.

Parameters: `progress_file (str)` (*path to progress file*)

Returns: `vars (dict)`

Return type: All progress file variables

`moseq2_app.gui.progress.show_progress_bar` (nfound, total, desc)
Helper function to print progress bars for each MoSeq-step progress dict

Parameters:

- **i_dict (dict)** (*Progress dict.*)
- **nfound (int)** (*Total number of found progress items*)
- **total (int)** (*Total number of progress items*)
- **desc (str)** (*Progress description text to display.*)

`moseq2_app.gui.progress.update_progress` (progress_file, varK, varV)
Updates progress file with new notebook variable

Parameters:

- **progress_file (str)** (*path to progress file*)
- **varK (str)** (*key in progress file to update*)
- **varV (str)** (*updated value to write*)

Returns: **progress (dict)**

Return type: Loaded path dict from the progress yaml file.

GUI - Widgets Module

`class moseq2_app.gui.widgets.GroupSettingWidgets`
Bases: `object`

GUI - Wrappers Module

Wrapper functions for all the functionality included in moseq2-app. These functions are executed directly from main.py.

`moseq2_app.gui.wrappers.get_frame_flips_wrapper` (input_dir, output_file, max_frames=1000000.0, tail_filter_iters=1, space_filter_size=3)

Wrapper function that facilitates the interactive

Parameters:

- **input_dir (str)** (*Input directory containing extracted session folders.*)
- **output_file (str)** (*Path to save flip classifier in.*)
- **max_frames (int)** (*Maximum number of frames to load from the extracted data.*)
- **tail_filter_iters (int)** (*Number of tail filtering iterations*)
- **prefilter_kernel_size (int)** (*Size of the median spatial filter.*)

Returns: **flip_finder (FlipRangeTool)** – hold the labeled accepted frame ranges and selected paths/info.

Return type: Flip Classifier Training object that will be used throughout the notebook to

`moseq2_app.gui.wrappers.interactive_crowd_movie_comparison_preview_wrapper`
(config_filepath, index_path, model_path, syll_info_path, output_dir, df_path=None, get_pdfs=True)

Wrapper function that launches an interactive crowd movie comparison application. Uses ipywidgets and Bokeh to facilitate real time user interaction.

Parameters:

- **config_filepath (str)** (path to config file containing crowd movie generation parameters)
- **index_path (str)** (path to index file with paths to all the extracted sessions)
- **model_path (str)** (path to trained model containing syllable labels.)
- **syll_info_path (str)** (path to syllable information file containing syllable labels)
- **output_dir (str)** (path to directory to store crowd movies)
- **df_path (str)** (optional path to pre-existing syllable information to plot)
- **get_pdfs (bool)** (indicates whether to compute and display position heatmaps)

`moseq2_app.gui.wrappers.interactive_extraction_preview_wrapper` (input_dir)

Interactive extraction previewing tool. Upon extracted session selection, function automatically displays the extraction mp4 video file.

Parameters: **input_dir (str)** (path to base directory containing extraction directories)

`moseq2_app.gui.wrappers.interactive_group_setting_wrapper` (index_filepath)

Wrapper function that handles the interactive group display and value updating.

Parameters: **index_filepath (str)** (Path to index file.)

`moseq2_app.gui.wrappers.interactive_plot_transition_graph_wrapper` (model_path, index_path, info_path, df_path=None, max_syllables=None)

Wrapper function that works as a background process that prepares the data for the interactive graphing function.

Parameters:

- **model_path (str)** (Path to trained model.)
- **index_path (str)** (Path to index file contained trained data metadata.)
- **info_path (str)** (Path to user-labeled syllable information file.)
- **df_path (str)** (Path to pre-saved syllable information.)
- **max_syllables (int or None)** (Limit maximum number of displayed syllables.)

`moseq2_app.gui.wrappers.interactive_roi_wrapper` (data_path, config_file, session_config=None, compute_bgs=True)

Interactive ROI detection wrapper function. Users can use run this wrapper to find the required extraction parameters, as well as preview examples of the extraction with the found parameters.

Parameters:

- **data_path (str)** (Path to base directory containing session folders.)
- **config_data (dict)** (ROI and Extraction configuration parameters)
- **session_parameters (str)** (Path to file containing individual session parameter sets.)

`moseq2_app.gui.wrappers.interactive_syllable_labeler_wrapper` (model_path, config_file, index_file, crowd_movie_dir, output_file, max_syllables=None, n_explained=99)

Wrapper function to launch a syllable crowd movie preview and interactive labeling application.

Parameters:

- **model_path (str)** (Path to trained model.)
- **crowd_movie_dir (str)** (Path to crowd movie directory)
- **output_file (str)** (Path to syllable label information file)
- **max_syllables (int)** (Maximum number of syllables to preview and label.)

`moseq2_app.gui.wrappers.interactive_syllable_stat_wrapper` (index_path, model_path, info_path, df_path=None, max_syllables=None)

Wrapper function to launch the interactive syllable statistics API. Users will be able to view different syllable statistics, sort them according to their metric of choice, and dynamically group the data to view individual sessions or group averages.

Parameters:

- **index_path (str)** (*Path to index file.*)
- **model_path (str)** (*Path to trained model file.*)
- **info_path (str)** (*Path to syllable information file.*)
- **max_syllables (int)** (*Maximum number of syllables to plot.*)

`moseq2_app.gui.wrappers.validate_extractions_wrapper` (input_dir)

Wrapper function to test the measured scalar values to determine whether some sessions should be flagged and diagnosed before aggregating the sessions.

Parameters: **input_dir (str)** (*path to parent directory containing all the extracted session folders*)

Module contents

moseq2_app.stat package

Submodules

Stats - Controller Module

Main interactive model syllable statistics results app functionality.

`class moseq2_app.stat.controller.InteractiveSyllableStats` (index_path, model_path, df_path, info_path, max_sylls)

Bases: `moseq2_app.stat.widgets.SyllableStatWidgets`

Interactive Syllable Statistics grapher class that holds the context for the current inputted session.

`clear_on_click (b)`
Clears the cell output

Parameters: **b (button click)**

`compute_dendrogram ()`
Computes the pairwise distances between the included model AR-states, and generates the graph information to be plotted after the stats.

`interactive_stat_helper ()`

Computes and saves the all the relevant syllable information to be displayed.

Loads the syllable information dict and merges it with the syllable statistics DataFrame.

`interactive_syll_stats_grapher` (stat, sort, groupby, errorbar, sessions, ctrl_group, exp_group)

Helper function that is responsible for handling ipywidgets interactions and updating the currently displayed Bokeh plot.

Parameters:

- **stat (str or ipywidgets.DropDown)** (*Statistic to plot: ['usage', 'speed', 'distance to center']*)
- **sort (str or ipywidgets.DropDown)** (*Statistic to sort syllables by (in descending order). – ['usage', 'speed', 'distance to center', 'similarity', 'difference']*).
- **groupby (str or ipywidgets.DropDown)** (*Data to plot; either group averages, or individual session data.*)
- **errorbar (str or ipywidgets.DropDown)** (*Error bar to display. ['SEM', 'STD']*)
- **sessions (list or ipywidgets.MultiSelect)** (*List of selected sessions to display data from.*)
- **ctrl_group (str or ipywidgets.DropDown)** (*Name of control group to compute group difference sorting with.*)
- **exp_group (str or ipywidgets.DropDown)** (*Name of comparative group to compute group difference sorting with.*)

`class moseq2_app.stat.controller.InteractiveTransitionGraph (model_path, index_path, info_path, df_path, max_sylls)`

Bases: `moseq2_app.stat.widgets.TransitionGraphWidgets`

Interactive transition graph class used to facilitate interactive graph generation and thresholding functionality.

`clear_on_click (b)`

Clears the cell output

Parameters: **b (button click)**

`compute_entropies (labels, label_group)`

Compute individual syllable entropy and transition entropy rates for all sessions with in a label_group.

Parameters:

- **labels (2d list)** (*list of session syllable labels over time.*)
- **label_group (list)** (*list of groups computing entropies for.*)

`compute_entropy_differences ()`

Computes cross group entropy/entropy-rate differences

and casts them to OrderedDict objects

`initialize_transition_data ()`

Performs all necessary pre-processing to compute the transition graph data and

syllable metadata to display via HoverTool.

Stores the syll_info dict, groups to explore, maximum number of syllables, and

the respective transition graphs and syllable scalars associated.

`interactive_transition_graph_helper (layout, scalar_color, edge_threshold, usage_threshold, speed_threshold)`

Helper function that generates all the transition graphs given the currently selected thresholding values, then displays them in a Jupyter notebook or web page.

Parameters:

- **edge_threshold (tuple or ipywidgets.FloatRangeSlider)** (*Transition probability range to include in graphs.*)
- **usage_threshold (tuple or ipywidgets.FloatRangeSlider)** (*Syllable usage range to include in graphs.*)
- **speed_threshold (tuple or ipywidgets.FloatRangeSlider)** (*Syllable speed range to include in graphs.*)

`on_set_scalar (event)`

Updates the scalar threshold slider filter criteria according to the current node coloring. Changes the name of the slider as well.

Parameters: **event (dropdown event)** (*User changes selected dropdown value*)

set_range_widget_values ()

After the dataset is initialized, the threshold range sliders' values will be set
according to the standard deviations of the dataset.

Stats - View Module

`moseq2_app.stat.view.bokeh_plotting` (df, stat, sorting, mean_df=None, groupby='group', errorbar='SEM', syllable_families=None, sort_name='usage')

Generates a Bokeh plot with interactive tools such as the HoverTool, which displays additional syllable information and the associated crowd movie.

Parameters:

- **df (pd.DataFrame)** (*Mean syllable statistic DataFrame.*)
- **stat (str)** (*Statistic to plot*)
- **sorting (list)** (*List of the current/selected syllable ordering*)
- **groupby (str)** (*Value to group data by. Either by unique group name or session name.*)
- **errorbar (str)** (*Error bar type to display*)
- **syllable_families (dict)** (*dict containing cladogram figure*)
- **sort_name (str)** (*Syllable sorting name displayed in title.*)

Returns: **p (bokeh figure)**

Return type: Displayed stat plot with optional color pickers.

`moseq2_app.stat.view.clamp` (val, minimum=0, maximum=255)

Caps the given R/G/B value to set min and max values

https://thadeusb.com/weblog/2010/10/10/python_scale_hex_color/

Parameters:

- **val (float)** (*value for given color tuple member*)
- **minimum (int)** (*min thresholding value*)
- **maximum (int)** (*max thresholding value*)

Returns: **val (int)**

Return type: thresholded color tuple member

`moseq2_app.stat.view.colorscale` (hexstr, scalefactor)

Scales a hex string by scalefactor. Returns scaled hex string.

To darken the color, use a float value between 0 and 1. To brighten the color, use a float value greater than 1.

```
>>> colorscale("#DF3C3C", .5)
#6F1E1E
>>> colorscale("#52D24F", 1.6)
#83FF7E
>>> colorscale("#4F75D2", 1)
#4F75D2
```

`moseq2_app.stat.view.draw_stats` (fig, df, groups, colors, sorting, groupby, stat, errorbar, line_dash='solid')

Helper function to bokeh_plotting that iterates through the given DataFrame and plots the data grouped by some user defined column ('group', 'SessionName'), with the errorbars of their choice.

Parameters:

- **fig (bokeh figure)** (*Figure to draw line plot glyphs on*)
- **df (pd.DataFrame)** (*DataFrame containing all relevant data to plot*)
- **groups (list of str)** (*List of group names to iterate by*)
- **colors (list of str)** (*List of group-corresponding colors to iterate by*)
- **sorting (list of int)** (*Pre-selected syllable index order*)
- **groupby (str)** (*string that indicates which DataFrame column is being grouped.*)
- **stat (str)** (*String that indicates the statistic that is being plotted.*)
- **errorbar (str)** (*String that indicates the type of error bars to be plotted.*)

Returns: **pickers (list of ColorPickers)**

Return type: List of interactive color picker widgets to update the graph colors

`moseq2_app.stat.view.format_graphs` (graphs, group)

Formats multiple transition graphs to be stacked in vertical column-order with graph positions corresponding to the difference graphs.

For example for 3 groups output would look like this: [a b-a c-a] [b c-b] [c]

Parameters:

- **graphs (list)** (*list of generated Bokeh figures.*)
- **group (list)** (*list of unique groups*)

Returns: **formatted_plots (2D list)**

Return type: list of lists corresponding to rows of figures being plotted.

`moseq2_app.stat.view.get_neighbors_and_entropies` (graph, node_indices, entropies, entropy_rates, group_name)

Computes the incoming and outgoing syllable entropies, entropy rates, previous nodes and
neighboring nodes for all the nodes included in node_indices.

Parameters:

- **graph (networkx DiGraph)** (*Generated DiGraph to convert to Bokeh glyph and plot.*)
- **node_indices (list)** (*List of node indices included in the given graph*)
- **entropies (list)** (*Syllable usage entropy values for each included syllable in the graph*)
- **entropy_rates (list)** (*Syllable transition entropy rates for all possible edge transitions in the graph*)
- **group_name (str)** (*Graph's group name.*)

Returns: **entropy_in (list)** (*List of computed incoming syllable entropy values for all the given node indices*) **entropy_out (list)** (*List of computed outgoing syllable entropy values for all the given node indices*) **prev_states (list)** (*List of previous nodes for each node index in the graph.*) **next_states (list)** (*List of successor nodes/syllables for each node in the graph*) **neighbor_edge_colors (list)** (*List of colors determining whether an edge is incoming or outgoing from each node.*) – Where orange = incoming, and purple = outgoing

`moseq2_app.stat.view.graph_dendrogram` (obj)

Graphs the distance sorted dendrogram representing syllable neighborhoods. Distance sorting is computed by processing the respective syllable AR matrices.

Parameters: **obj (InteractiveSyllableStats object)** (*Syllable Stats object containing syllable stat information.*)

`moseq2_app.stat.view.plot_interactive_transition_graph` (graphs, pos, group, group_names, usages, syll_info, entropies, entropy_rates, scalars, scalar_color='default')

Converts the computed networkx transition graphs to Bokeh glyph objects that can be interacted with and updated throughout run-time.

Parameters:

- **graphs (list of nx.DiGraphs)** (list of created networkx graphs.)
- **pos (nx.Layout)** (shared node position coordinates layout object.)
- **group (list)** (list of unique group names.)
- **group_names (list)** (list of names for all the generated transition graphs + difference graphs)
- **usages (list of OrderedDicts)** (list of OrderedDicts containing syllable usages.)
- **syll_info (dict)** (dict of syllable label information to display with HoverTool)
- **scalars (dict)** (dict of syllable scalar information to display with HoverTool)

Stats - Widgets Module

```
class moseq2_app.stat.widgets.SyllableStatWidgets
```

Bases: `object`

```
class moseq2_app.stat.widgets.TransitionGraphWidgets
```

Bases: `object`

```
edge_thresholder = widgets.FloatRangeSlider(value=[0.0025, 1], min=0, max=1, step=0.001, style=style,
readout_format='.4f',
```

```
description='Edges weights to display', continuous_update=False)
```

```
usage_thresholder = widgets.FloatRangeSlider(value=[0, 1], min=0, max=1, step=0.001, style=style,
readout_format='.4f',
```

```
description='Usage nodes to display', continuous_update=False)
```

```
speed_thresholder = widgets.FloatRangeSlider(value=[-25, 200], min=-50, max=200, step=1, style=style,
readout_format='.1f',
```

```
description='Threshold nodes by speed', continuous_update=False)
```

Module contents

moseq2_app.viz package

Submodules

Viz - Controller Module

Main syllable crowd movie viewing, comparing, and labeling functionality.

```
class moseq2_app.viz.controller.CrowdMovieComparison (config_data, index_path, df_path, model_path,
syll_info, output_dir, get_pdfs)
```

Bases: `moseq2_app.viz.widgets.CrowdMovieCompareWidgets`

Crowd Movie Comparison application class. Contains all the user inputted parameters within its context.

```
clear_on_click (b)
```

Clears the cell output

Parameters: `b`

```
crowd_movie_preview (syllable, groupby, nexamples)
```

Helper function that triggers the crowd_movie_wrapper function and creates the HTML divs containing the generated crowd movies. Function is triggered whenever any of the widget function inputs are changed.

Parameters:

- **syllable** (int or ipywidgets.DropDownMenu) (*Currently displayed syllable.*)
- **nexamples** (int or ipywidgets.IntSlider) (*Number of mice to display per crowd movie.*)

generate_crowd_movie_divs ()

Generates HTML divs containing crowd movies and syllable metadata tables

from the given syllable dict file.

Returns: **divs** (list of Bokeh.models.Div) (*Divs of HTML videos and metadata tables.*) **bk_plots** (list) (*list of corresponding position heatmap figures.*)

get_mean_group_dict (group_df)

Creates a dict object to convert to a displayed table containing syllable scalars.

Parameters: **group_df** (pd.DataFrame) (*DataFrame containing mean syllable scalar data for each session and their groups*)

get_pdf_plot (group_syllable_pdf, group_name)

Helper function that creates a bokeh plot with the given PDF heatmap and figure title.

Parameters:

- **group_syllable_pdf** (2D np.ndarray) (*Mean syllable position PDF heatmap.*)
- **group_name** (str) (*Name of group for generated syllable pdf*)

Returns: **pdf_fig** (bokeh.figure)

Return type: Create bokeh figure.

get_selected_session_syllable_info (sel_sessions)

Prepares dict of session-based syllable information to display.

Parameters: **sel_sessions** (list) (*list of selected session names.*)

get_session_mean_syllable_info_df (model_fit, sorted_index)

Populates session-based syllable information dict with usage and scalar information.

Parameters:

- **model_fit** (dict) (*dict containing trained model syllable data*)
- **sorted_index** (dict) (*sorted index file containing paths to extracted session h5s*)

on_click_trigger_button (b)

Generates crowd movies and displays them when the user clicks the trigger button

Parameters: **b** (ipywidgets.Button click event) (*User clicks "Generate Movies" button*)

select_session (event)

Callback function to save the list of selected sessions to config_data,

and get session syllable info to pass to crowd_movie_wrapper and create the accompanying syllable scalar metadata table.

Parameters: **event** (event) (*User clicks on multiple sessions in the SelectMultiple widget*)

set_default_cm_parameters ()

show_session_select (change)

Callback function to change current view to show session selector when user switches DropDownMenu selection to 'SessionName', and hides it if the user selects 'groups'.

Parameters: **change** (event) (*User switches their DropDownMenu selection*)

`class moseq2_app.viz.controller.SyllableLabeler (model_fit, index_file, max_sylls, save_path)`

Bases: `moseq2_app.viz.widgets.SyllableLabelerWidgets`

Class that contains functionality for previewing syllable crowd movies and

user interactions with buttons and menus.

`clear_on_click(b)`

Clears the cell output

Parameters: **b (button click)**

`get_crowd_movie_paths(index_path, model_path, config_data, crowd_movie_dir)`

Populates the syllable information dict with the respective crowd movie paths.

Parameters: **crowd_movie_dir (str)** (*Path to directory containing all the generated crowd movies*)

`get_mean_group_dict(group_df)`

Creates a dict object to convert to a displayed table containing syllable scalars.

Parameters: **group_df (pd.DataFrame)** (*DataFrame containing mean syllable scalar data for each session and their groups*)

`get_mean_syllable_info()`

Populates syllable information dict with usage and scalar information.

`interactive_syllable_labeler(syllables)`

Helper function that facilitates the interactive view. Function will create a Bokeh Div object that will display the current video path.

Parameters: **syllables (int or ipywidgets.DropDownMenu)** (*Current syllable to label*)

`on_next(event)`

Callback function to trigger an view update when the user clicks the "Next" button.

Parameters: **event (ipywidgets.ButtonClick)** (*User clicks next button.*)

`on_prev(event)`

Callback function to trigger an view update when the user clicks the "Previous" button.

Parameters: **event (ipywidgets.ButtonClick)** (*User clicks 'previous' button.*)

`on_set(event)`

Callback function to save the dict to syllable information file.

Parameters: **event (ipywidgets.ButtonClick)** (*User clicks the 'Save' button.*)

`set_default_cm_parameters(config_data)`

Sets default crowd movie generation parameters that may be manually updated.

Parameters: **config_data (dict)** (*Dict of main moseq configuration parameters.*)

Returns: **config_data (dict)** – with default crowd movie generation parameters.

Return type: Updated dict of main moseq configuration parameters

`set_group_info_widgets(group_info)`

Display function that reads the syllable information into a pandas DataFrame, converts it to an HTML table and displays it in a Bokeh Div facilitated via the Output() widget.

Parameters: **group_info (dict)** (*Dictionary of grouped current syllable information*)

Viz - View Module

Helper function that displays a grid of crowd movies and plotted bokeh figures of position heatmaps.

`moseq2_app.viz.view.display_crowd_movies(widget_box, curr_name, desc, divs, bk_figs)`

Crowd movie comparison helper function that displays the widgets and embedded HTML divs to a running jupyter notebook cell or HTML webpage.

Parameters: **divs** (list of `bokeh.models.Div`) (*list of HTML Div objects containing videos to display*)

Viz - Widgets Module

```
class moseq2_app.viz.widgets.CrowdMovieCompareWidgets
```

Bases: `object`

```
class moseq2_app.viz.widgets.SyllableLabelerWidgets
```

Bases: `object`

Module contents

Index

- `genindex`

Index

A

augment_dataset()
(moseq2_app.flip.controller.FlipRangeTool method)

B

bokeh_plot_helper() (in module moseq2_app.roi.view)
bokeh_plotting() (in module moseq2_app.stat.view)

C

check_all_sessions()
(moseq2_app.roi.controller.InteractiveFindRoi method)

check_progress() (in module
moseq2_app.gui.progress)

check_timestamp_error_percentage() (in module
moseq2_app.roi.validation)

clamp() (in module moseq2_app.stat.view)

clear_on_click()
(moseq2_app.flip.controller.FlipRangeTool method)

(moseq2_app.roi.controller.InteractiveExtractionViewer
method)

(moseq2_app.roi.controller.InteractiveFindRoi method)

(moseq2_app.stat.controller.InteractiveSyllableStats
method)

(moseq2_app.stat.controller.InteractiveTransitionGraph
method)

(moseq2_app.viz.controller.CrowdMovieComparison
method)

(moseq2_app.viz.controller.SyllableLabeler method)

colorscale() (in module moseq2_app.stat.view)

compute_all_bgs()
(moseq2_app.roi.controller.InteractiveFindRoi method)

compute_dendrogram()
(moseq2_app.stat.controller.InteractiveSyllableStats
method)

compute_entropies()
(moseq2_app.stat.controller.InteractiveTransitionGraph
method)

compute_entropy_differences()
(moseq2_app.stat.controller.InteractiveTransitionGraph
method)

compute_kl_divergences() (in module
moseq2_app.roi.validation)

count_frames_with_small_areas() (in module
moseq2_app.roi.validation)

count_missing_mouse_frames() (in module
moseq2_app.roi.validation)

count_nan_rows() (in module
moseq2_app.roi.validation)

count_stationary_frames() (in module
moseq2_app.roi.validation)

count_total_found_items() (in module
moseq2_app.gui.progress)

crowd_movie_preview()
(moseq2_app.viz.controller.CrowdMovieComparison
method)

CrowdMovieCompareWidgets (class in
moseq2_app.viz.widgets)

CrowdMovieComparison (class in
moseq2_app.viz.controller)

D

display_crowd_movies() (in module
moseq2_app.viz.view)

draw_stats() (in module moseq2_app.stat.view)

E

extract_button_clicked()
(moseq2_app.roi.controller.InteractiveFindRoi method)

F

flip_classifier_tool() (in module moseq2_app.main)

FlipClassifierWidgets (class in
moseq2_app.flip.widgets)

FlipRangeTool (class in moseq2_app.flip.controller)

format_graphs() (in module moseq2_app.stat.view)

G

generate_crowd_movie_divs()
(moseq2_app.viz.controller.CrowdMovieComparison
method)

generate_missing_metadata() (in module
moseq2_app.gui.progress)

get_anomaly_dict() (in module
moseq2_app.roi.validation)

get_corrected_data()
(moseq2_app.flip.controller.FlipRangeTool method)

get_crowd_movie_paths()
(moseq2_app.viz.controller.SyllableLabeler method)

get_extraction()
(moseq2_app.roi.controller.InteractiveExtractionViewer
method)

(moseq2_app.roi.controller.InteractiveFindRoi
method)

get_extraction_progress() (in module
moseq2_app.gui.progress)

<code>get_frame_flips_wrapper()</code> (<code>moseq2_app.gui.wrappers</code>)	(in module	<code>interactive_crowd_movie_comparison_preview_wrapper()</code> (in module <code>moseq2_app.gui.wrappers</code>)
<code>get_iqr_anomaly_sessions()</code> (<code>moseq2_app.roi.validation</code>)	(in module	<code>interactive_depth_finder()</code> (<code>moseq2_app.roi.controller.InteractiveFindRoi</code> method)
<code>get_kl_divergence_outliers()</code> (<code>moseq2_app.roi.validation</code>)	(in module	<code>interactive_extraction_preview_wrapper()</code> (in module <code>moseq2_app.gui.wrappers</code>)
<code>get_mean_group_dict()</code> (<code>moseq2_app.viz.controller.CrowdMovieComparison</code> method) (<code>moseq2_app.viz.controller.SyllableLabeler</code> method)		<code>interactive_find_roi_session_selector()</code> (<code>moseq2_app.roi.controller.InteractiveFindRoi</code> method)
<code>get_mean_syllable_info()</code> (<code>moseq2_app.viz.controller.SyllableLabeler</code> method)		<code>interactive_group_setting()</code> (in module <code>moseq2_app.main</code>)
<code>get_neighbors_and_entropies()</code> (in module <code>moseq2_app.stat.view</code>)		<code>interactive_group_setting_wrapper()</code> (in module <code>moseq2_app.gui.wrappers</code>)
<code>get_pca_progress()</code> (in module <code>moseq2_app.gui.progress</code>)		<code>interactive_launch_frame_selector()</code> (<code>moseq2_app.flip.controller.FlipRangeTool</code> method)
<code>get_pdf_plot()</code> (<code>moseq2_app.viz.controller.CrowdMovieComparison</code> method)		<code>interactive_plot_transition_graph_wrapper()</code> (in module <code>moseq2_app.gui.wrappers</code>)
<code>get_pixels_per_metric()</code> (<code>moseq2_app.roi.controller.InteractiveFindRoi</code> method)		<code>interactive_roi_detector()</code> (in module <code>moseq2_app.main</code>)
<code>get_roi_and_depths()</code> (<code>moseq2_app.roi.controller.InteractiveFindRoi</code> method)		<code>interactive_roi_wrapper()</code> (in module <code>moseq2_app.gui.wrappers</code>)
<code>get_scalar_df()</code> (in module <code>moseq2_app.roi.validation</code>)		<code>interactive_stat_helper()</code> (<code>moseq2_app.stat.controller.InteractiveSyllableStats</code> method)
<code>get_selected_session()</code> (<code>moseq2_app.roi.controller.InteractiveFindRoi</code> method)		<code>interactive_syll_stats_grapher()</code> (<code>moseq2_app.stat.controller.InteractiveSyllableStats</code> method)
<code>get_selected_session_syllable_info()</code> (<code>moseq2_app.viz.controller.CrowdMovieComparison</code> method)		<code>interactive_syllable_labeler()</code> (<code>moseq2_app.viz.controller.SyllableLabeler</code> method)
<code>get_session_mean_syllable_info_df()</code> (<code>moseq2_app.viz.controller.CrowdMovieComparison</code> method)		<code>interactive_syllable_labeler_wrapper()</code> (in module <code>moseq2_app.gui.wrappers</code>)
<code>get_session_paths()</code> (in module <code>moseq2_app.gui.progress</code>)		<code>interactive_syllable_stat_wrapper()</code> (in module <code>moseq2_app.gui.wrappers</code>)
<code>graph_dendrogram()</code> (in module <code>moseq2_app.stat.view</code>)		<code>interactive_syllable_stats()</code> (in module <code>moseq2_app.main</code>)
<code>GroupSettingWidgets</code> (class in <code>moseq2_app.gui.widgets</code>)		<code>interactive_transition_graph()</code> (in module <code>moseq2_app.main</code>)
		<code>interactive_transition_graph_helper()</code> (<code>moseq2_app.stat.controller.InteractiveTransitionGraph</code> method)
		<code>InteractiveExtractionViewer</code> (class in <code>moseq2_app.roi.controller</code>)
		<code>InteractiveFindRoi</code> (class in <code>moseq2_app.roi.controller</code>)
		<code>InteractiveROIWidgets</code> (class in <code>moseq2_app.roi.widgets</code>)
		<code>InteractiveSyllableStats</code> (class in <code>moseq2_app.stat.controller</code>)
		<code>InteractiveTransitionGraph</code> (class in <code>moseq2_app.stat.controller</code>)

H

<code>handle_progress_restore_input()</code> (<code>moseq2_app.gui.progress</code>)	(in module
--	------------

I

<code>index_to_dataframe()</code> (in module <code>moseq2_app.util</code>)
<code>initialize_transition_data()</code> (<code>moseq2_app.stat.controller.InteractiveTransitionGraph</code> method)
<code>interactive_crowd_movie_comparison()</code> (in module <code>moseq2_app.main</code>)

L

[label_syllables\(\)](#) (in module [moseq2_app.main](#))
[load_sessions\(\)](#)
([moseq2_app.flip.controller.FlipRangeTool](#) method)

M

[make_session_status_dicts\(\)](#) (in module [moseq2_app.roi.validation](#))
[mark_passing_button_clicked\(\)](#)
([moseq2_app.roi.controller.InteractiveFindRoi](#) method)

module

[moseq2_app.flip](#)
[moseq2_app.flip.controller](#)
[moseq2_app.flip.widgets](#)
[moseq2_app.gui](#)
[moseq2_app.gui.progress](#)
[moseq2_app.gui.widgets](#)
[moseq2_app.gui.wrappers](#)
[moseq2_app.main](#)
[moseq2_app.roi](#)
[moseq2_app.roi.controller](#)
[moseq2_app.roi.validation](#)
[moseq2_app.roi.view](#)
[moseq2_app.roi.widgets](#)
[moseq2_app.stat](#)
[moseq2_app.stat.controller](#)
[moseq2_app.stat.view](#)
[moseq2_app.stat.widgets](#)
[moseq2_app.util](#)
[moseq2_app.viz](#)
[moseq2_app.viz.controller](#)
[moseq2_app.viz.view](#)
[moseq2_app.viz.widgets](#)

moseq2_app.flip

[module](#)

moseq2_app.flip.controller

[module](#)

moseq2_app.flip.widgets

[module](#)

moseq2_app.gui

[module](#)

moseq2_app.gui.progress

[module](#)

moseq2_app.gui.widgets

[module](#)

moseq2_app.gui.wrappers

[module](#)

moseq2_app.main

[module](#)

moseq2_app.roi

[module](#)

moseq2_app.roi.controller

[module](#)

moseq2_app.roi.validation

[module](#)

moseq2_app.roi.view

[module](#)

moseq2_app.roi.widgets

[module](#)

moseq2_app.stat

[module](#)

moseq2_app.stat.controller

[module](#)

moseq2_app.stat.view

[module](#)

moseq2_app.stat.widgets

[module](#)

moseq2_app.util

[module](#)

moseq2_app.viz

[module](#)

moseq2_app.viz.controller

[module](#)

moseq2_app.viz.view

[module](#)

moseq2_app.viz.widgets

[module](#)

O

[on_click_trigger_button\(\)](#)
([moseq2_app.viz.controller.CrowdMovieComparison](#) method)
[on_next\(\)](#) ([moseq2_app.viz.controller.SyllableLabeler](#) method)
[on_prev\(\)](#) ([moseq2_app.viz.controller.SyllableLabeler](#) method)
[on_set\(\)](#) ([moseq2_app.viz.controller.SyllableLabeler](#) method)
[on_set_scalar\(\)](#)
([moseq2_app.stat.controller.InteractiveTransitionGraph](#) method)

P

[plot_heatmap\(\)](#) (in module [moseq2_app.roi.validation](#))

[plot_interactive_transition_graph\(\)](#) (in module [moseq2_app.stat.view](#))
[plot_roi_results\(\)](#) (in module [moseq2_app.roi.view](#))
[prepare_data_to_plot\(\)](#)
 ([moseq2_app.roi.controller.InteractiveFindRoi](#) method)
[prepare_datasets\(\)](#)
 ([moseq2_app.flip.controller.FlipRangeTool](#) method)
[preview_extractions\(\)](#) (in module [moseq2_app.main](#))
[print_progress\(\)](#) (in module [moseq2_app.gui.progress](#))
[print_validation_results\(\)](#) (in module [moseq2_app.roi.validation](#))

R

[restore_progress_vars\(\)](#) (in module [moseq2_app.gui.progress](#))
[run_heatmap_kl_divergence_test\(\)](#) (in module [moseq2_app.roi.validation](#))
[run_validation_tests\(\)](#) (in module [moseq2_app.roi.validation](#))

S

[save_clicked\(\)](#)
 ([moseq2_app.roi.controller.InteractiveFindRoi](#) method)
[select_session\(\)](#)
 ([moseq2_app.viz.controller.CrowdMovieComparison](#) method)
[set_default_cm_parameters\(\)](#)
 ([moseq2_app.viz.controller.CrowdMovieComparison](#) method)
 ([moseq2_app.viz.controller.SyllableLabeler](#) method)
[set_group_info_widgets\(\)](#)
 ([moseq2_app.viz.controller.SyllableLabeler](#) method)
[set_range_widget_values\(\)](#)
 ([moseq2_app.stat.controller.InteractiveTransitionGraph](#) method)
[show_extraction\(\)](#) (in module [moseq2_app.roi.view](#))
[show_progress_bar\(\)](#) (in module [moseq2_app.gui.progress](#))
[show_session_select\(\)](#)
 ([moseq2_app.viz.controller.CrowdMovieComparison](#) method)
[start_stop_frame_range\(\)](#)
 ([moseq2_app.flip.controller.FlipRangeTool](#) method)
[SyllableLabeler](#) (class in [moseq2_app.viz.controller](#))
[SyllableLabelerWidgets](#) (class in [moseq2_app.viz.widgets](#))
[SyllableStatWidgets](#) (class in [moseq2_app.stat.widgets](#))

T

[test_all_sessions\(\)](#)
 ([moseq2_app.roi.controller.InteractiveFindRoi](#) method)
[train_and_evaluate_model\(\)](#)
 ([moseq2_app.flip.controller.FlipRangeTool](#) method)
[train_and_save_model\(\)](#)
 ([moseq2_app.flip.controller.FlipRangeTool](#) method)
[TransitionGraphWidgets](#) (class in [moseq2_app.stat.widgets](#))

U

[update_checked_list\(\)](#)
 ([moseq2_app.roi.controller.InteractiveFindRoi](#) method)
[update_config_fr\(\)](#)
 ([moseq2_app.roi.controller.InteractiveFindRoi](#) method)
[update_minmax_config\(\)](#)
 ([moseq2_app.roi.controller.InteractiveFindRoi](#) method)
[update_progress\(\)](#) (in module [moseq2_app.gui.progress](#))

V

[validate_extractions\(\)](#) (in module [moseq2_app.main](#))
[validate_extractions_wrapper\(\)](#) (in module [moseq2_app.gui.wrappers](#))
[view_extraction\(\)](#) (in module [moseq2_app.main](#))

Python Module Index

m

- moseq2_app
- moseq2_app.flip
- moseq2_app.flip.controller
- moseq2_app.flip.widgets
- moseq2_app.gui
- moseq2_app.gui.progress
- moseq2_app.gui.widgets
- moseq2_app.gui.wrappers
- moseq2_app.main
- moseq2_app.roi
- moseq2_app.roi.controller
- moseq2_app.roi.validation
- moseq2_app.roi.view
- moseq2_app.roi.widgets
- moseq2_app.stat
- moseq2_app.stat.controller
- moseq2_app.stat.view
- moseq2_app.stat.widgets
- moseq2_app.util
- moseq2_app.viz
- moseq2_app.viz.controller
- moseq2_app.viz.view
- moseq2_app.viz.widgets