Python Documentation

version

April 15, 2020

Contents

Welcome to moseq2-pca's documentation! 1

moseq2-pca 1

moseq2_pca package 1

Subpackages 1

moseq2_pca.helpers package 1

Helpers - Data Module 1

Helpers - Wrapper Module 1

moseq2_pca.pca package 2

PCA - Utilties Module 2

moseq2_pca.tests package 5

Subpackages 5

moseq2_pca.tests.integration_tests package 5

Integration Tests - CLI Tests Module 5

Integration Tests - GUI Tests Module 5

moseq2_pca.tests.unit_tests package 5

Unit Tests - PCA Utilities Tests Module 5

Unit Tests - General Utilities Tests Module 5

Unit Tests - Visualization Tests Module 6

CLI Module 6

cli 6

apply-pca 6

clip-scores 7

compute-changepoints 8

train-pca 9

GUI Module 10

Utilities Module 11

Visualization Module 14

Indices and tables 14

Index 15

Python Module Index 21

Welcome to moseq2-pca's documentation!

moseq2-pca

moseq2_pca package

Subpackages

moseq2_pca.helpers package

Helpers - Data Module

`moseq2_pca.helpers.data.`**`get_pca_yaml_data`** (pca_yaml)
Reads PCA yaml file and returns metadata

Parameters: **pca_yaml (str)** (*path to pca.yaml*)

Returns: **use_fft (bool)** (*indicates whether to use FFT*) **clean_params (dict)** (*dict of image filtering parameters*) **mask_params (dict)** (*dict of mask parameters)*) **missing_data (bool)** (*indicates whether to use mask_params*)

`moseq2_pca.helpers.data.`**`load_pcs_for_cp`** (pca_file_components, config_data)
Load computed Principal Components.

Parameters:
- **pca_file_components (str)** (*path to pca h5 file to read PCs*)
- **config_data (dict)** (*config parameters*)

Returns: **pca_components (str)** (*path to pca components*) **changepoint_params (dict)** (*dict of relevant changepoint parameters*) **cluster (dask Cluster)** (*Dask Cluster object.*) **client (dask Client)** (*Dask Client Object*) **missing_data (bool)** (*Indicates whether to use mask_params*) **mask_params (dict)** (*Mask parameters to use when computing CPs*)

`moseq2_pca.helpers.data.`**`setup_cp_command`** (input_dir, config_data, output_dir, output_file, output_directory)
Helper function for changepoints_wrapper to perform data-path existence checks.

Parameters:
- **input_dir (int)** (*path to directory containing all h5+yaml files*)
- **config_data (dict)** (*dict of relevant PCA parameters (image filtering etc.)*)
- **output_dir (str)** (*path to directory to store PCA data*)
- **output_file (str)** (*pca model filename*)
- **output_directory (str)** (*alternative output_dir*)

Returns: **config_data (dict)** (*updated config_data dict with the proper paths*) **pca_file_components (str)** (*path to trained pca file*) **pca_file_scores (str)** (*path to pca_scores file*) **h5s (list)** (*list of relevant pca h5 files*) **yamls (list)** (*list of relevant pca metadata yaml files*) **save_file (str)** (*path to save changepoints*)

Helpers - Wrapper Module

`moseq2_pca.helpers.wrappers.`**`apply_pca_wrapper`** (input_dir, config_data, output_dir, output_file, output_directory=None, gui=False)
Wrapper function to obtain PCA Scores.

Parameters:
- **input_dir (int)** (*path to directory containing all h5+yaml files*)
- **config_data (dict)** (*dict of relevant PCA parameters (image filtering etc.)*)
- **output_dir (str)** (*path to directory to store PCA data*)
- **output_file (str)** (*pca model filename*)
- **output_directory (str)** (*alternative output_dir*)
- **gui (bool)** (*indicate GUI is running*)

Returns: **config_data (dict)**

Return type: updated config_data variable to write back in GUI API

`moseq2_pca.helpers.wrappers.`**`compute_changepoints_wrapper`** (input_dir, config_data, output_dir, output_file, gui=False, output_directory=None)

Wrapper function to compute model-free (PCA based) Changepoints.

Parameters:
- **input_dir (int)** (*path to directory containing all h5+yaml files*)
- **config_data (dict)** (*dict of relevant PCA parameters (image filtering etc.)*)
- **output_dir (str)** (*path to directory to store PCA data*)
- **output_file (str)** (*pca model filename*)
- **output_directory (str)** (*alternative output_dir*)
- **gui (bool)** (*indicate GUI is running*)

Returns: **config_data (dict)**

Return type: updated config_data variable to write back in GUI API

`moseq2_pca.helpers.wrappers.`**`train_pca_wrapper`** (input_dir, config_data, output_dir, output_file, output_directory=None, gui=False)

Wrapper function to train PCA.

Parameters:
- **input_dir (int)** (*path to directory containing all h5+yaml files*)
- **config_data (dict)** (*dict of relevant PCA parameters (image filtering etc.)*)
- **output_dir (str)** (*path to directory to store PCA data*)
- **output_file (str)** (*pca model filename*)
- **output_directory (str)** (*alternative output_dir*)
- **gui (bool)** (*indicate GUI is running*)

Returns: **config_data (dict)**

Return type: updated config_data variable to write back in GUI API

moseq2_pca.pca package

PCA - Utilties Module

`moseq2_pca.pca.util.`**`apply_pca_dask`** (pca_components, h5s, yamls, use_fft, clean_params, save_file, chunk_size, mask_params, missing_data, client, fps=30, gui=False)

"Apply" trained PCA on input frame data to obtain PCA Scores using Distributed Dask cluster.

Parameters:
- **pca_components (np.array)** (*array of computed Principal Components*)
- **h5s (list)** (*list of h5 files*)
- **yamls (list)** (*list of yaml files*)
- **use_fft (bool)** (*indicate whether to use 2D-FFT*)
- **clean_params (dict)** (*dictionary containing filtering options*)
- **save_file (str)** (*path to pca_scores filename to save*)
- **chunk_size (int)** (*size of chunks to process*)
- **mask_params (dict)** (*dictionary of masking parameters (if missing data)*)
- **missing_data (bool)** (*indicates whether to use mask arrays.*)
- **fps (int)** (*frames per second*)

Returns:

Return type: None

`moseq2_pca.pca.util.`**`apply_pca_local`** (pca_components, h5s, yamls, use_fft, clean_params, save_file, chunk_size, mask_params, missing_data, fps=30)

"Apply" trained PCA on input frame data to obtain PCA Scores using local cluster/platform.

Parameters:
- **pca_components (np.array)** (*array of computed Principal Components*)
- **h5s (list)** (*list of h5 files*)
- **yamls (list)** (*list of yaml files*)
- **use_fft (bool)** (*indicate whether to use 2D-FFT*)
- **clean_params (dict)** (*dictionary containing filtering options*)
- **save_file (str)** (*path to pca_scores filename to save*)
- **chunk_size (int)** (*size of chunks to process*)
- **mask_params (dict)** (*dictionary of masking parameters (if missing data)*)
- **missing_data (bool)** (*indicates whether to use mask arrays.*)
- **fps (int)** (*frames per second*)

Returns:

Return type: None

`moseq2_pca.pca.util.`**`get_changepoints_dask`** (changepoint_params, pca_components, h5s, yamls, save_file, chunk_size, mask_params, missing_data, client, fps=30, pca_scores=None, progress_bar=False, gui=False)

Computes model-free changepoints using PCs and PC Scores on distributed dask cluster.

Parameters:
- **changepoint_params (dict)** (*dict of changepoint parameters*)
- **pca_components (np.array)** (*computed principal components*)
- **h5s (list)** (*list of h5 files*)
- **yamls (list)** (*list of yaml files*)
- **save_file (str)** (*path to save changepoint files*)
- **chunk_size (int)** (*size of chunks to process in dask.*)
- **mask_params (dict)** (*dict of missing_data mask parameters.*)
- **missing_data (bool)** (*indicate whether to use mask_params*)
- **client (dask Client)** (*initialized Dask Client object*)
- **fps (int)** (*frames per second*)
- **pca_scores (np.array)** (*computed principal component scores*)
- **progress_bar (bool)** (*display progress bar*)
- **gui (bool)** (*indicate GUI use*)

Returns:

Return type: None

`moseq2_pca.pca.util.`**`mask_data`** (original_data, mask, new_data)

Create a mask subregion given a boolean mask if missing data flag is used.

Parameters:
- **original_data (3d np.ndarray)** (*input frames*)
- **mask (3d boolean np.ndarray)** (*mask array*)
- **new_data (3d np.ndarray)** (*frames to use*)

Returns: **output (3d np.ndarray)**

Return type: masked data array

`moseq2_pca.pca.util.`**`train_pca_dask`** (dask_array, clean_params, use_fft, rank, cluster_type, client, workers, cache, mask=None, iters=10, recon_pcs=10, min_height=10, max_height=100)

Train PCA using dask arrays.

Parameters:
- **dask_array (dask array)** (*chunked frames to train PCA*)
- **clean_params (dict)** (*dictionary containing filtering parameters*)
- **use_fft (bool)** (*indicates whether to use 2d-FFT on images.*)
- **rank (int)** (*Matrix rank to use*)
- **cluster_type (str)** (*indicates which cluster to use.*)
- **client (Dask.Client)** (*client object to execute dask operations*)
- **workers (int)** (*number of dask workers*)
- **cache (str)** (*path to cache directory*)
- **mask (dask array)** (*dask array of masked data if missing_data parameter==True*)
- **iters (int)** (*number of SVD iterations*)
- **recon_pcs (int)** (*number of PCs to reconstruct. (if missing_data = True)*)
- **min_height (int)** (*minimum mouse height from floor in (mm)*)
- **max_height (int)** (*maximum mouse height from floor in (mm)*)

Returns: **output_dict (dict)**

Return type: dictionary containing PCA training results.

moseq2_pca.tests package

Subpackages

moseq2_pca.tests.integration_tests package

Integration Tests - CLI Tests Module

class moseq2_pca.tests.integration_tests.test_cli.**TestCli** (methodName='runTest')
Bases: **unittest.case.TestCase**

test_apply_pca ()

test_clip_scores ()

test_compute_changepoints ()

test_train_pca ()

Integration Tests - GUI Tests Module

class moseq2_pca.tests.integration_tests.test_gui.**TestGUI** (methodName='runTest')
Bases: **unittest.case.TestCase**

test_apply_pca_command ()

test_compute_changepoints_command ()

test_train_pca_command ()

moseq2_pca.tests.unit_tests package

Unit Tests - PCA Utilities Tests Module

class moseq2_pca.tests.unit_tests.test_pca_util.**TestPCAUtils** (methodName='runTest')
Bases: **unittest.case.TestCase**

test_mask_data ()

test_train_pca_dask ()

Unit Tests - General Utilities Tests Module

class moseq2_pca.tests.unit_tests.test_util.**TestUtils** (methodName='runTest')
Bases: **unittest.case.TestCase**

test_clean_frames ()

test_gauss_smooth ()

test_gaussian_kernel1d ()

test_get_changepoints ()

test_get_metadata_path ()

test_get_rps ()

test_get_rsp_dask ()

test_get_timestamp_path ()

test_initialize_dask ()

test_insert_nans ()

test_read_yaml ()

test_recursive_find_h5s ()

test_select_strel ()

Unit Tests - Visualization Tests Module

class moseq2_pca.tests.unit_tests.test_viz.**TestViz** (methodName='runTest')
Bases: **unittest.case.TestCase**

changepoint_dist ()

test_display_components ()
cmap = 'gray' im_size = int(np.sqrt(components.shape[1])) plotv = components.reshape((-1, im_size, im_size)) plotv = skimage.util.montage(plotv) plt.switch_backend('agg') fig, ax = plt.subplots(1, 1, figsize=(10, 10)) plt.imshow(plotv, cmap=cmap) plt.xticks([]) plt.yticks([])

test_scree_plot ()

CLI Module

cli

```
cli [OPTIONS] COMMAND [ARGS]...
```

apply-pca

```
cli apply-pca [OPTIONS]
```

Options

-i, --input-dir <input_dir>
Directory to find h5 files [default: /Users/aymanzeine/Desktop/moseq/moseq2-pca/docs]

--cluster-type <cluster_type>
Cluster type [default: local]

Options: local|slurm|nodask

-o, --output-dir <output_dir>
Directory to store results [default: /Users/aymanzeine/Desktop/moseq/moseq2-pca/docs/_pca]

--output-file <output_file>
Name of h5 file for storing pca results [default: pca_scores]

--h5-path <h5_path>
Path to data in h5 files [default: /frames]

--h5-mask-path <h5_mask_path>

Path to log-likelihood mask in h5 files [default: /frames_mask]

--pca-path <pca_path>
Path to pca components [default: /components]

--pca-file <pca_file>
Path to PCA results

--chunk-size <chunk_size>
Number of frames per chunk [default: 4000]

--fill-gaps <fill_gaps>
Fill dropped frames with nans [default: True]

--fps <fps>
Fps (only used if no timestamps found) [default: 30]

--detrend-window <detrend_window>
Length of detrend window (in seconds, 0 for no detrending) [default: 0]

--config-file <config_file>
Path to configuration file

-d, --dask-cache-path <dask_cache_path>
Path to spill data to disk for dask local scheduler [default: /Users/aymanzeine/moseq2_pca]

-q, --queue <queue>
Cluster queue/partition for submitting jobs [default: debug]

-n, --nworkers <nworkers>
Number of workers [default: 10]

-c, --cores <cores>
Number of cores per worker [default: 1]

-p, --processes <processes>
Number of processes to run on each worker [default: 1]

-m, --memory <memory>
RAM usage per workers [default: 15GB]

-w, --wall-time <wall_time>
Wall time for workers [default: 06:00:00]

--timeout <timeout>
Time to wait for workers to initialize before proceeding (minutes) [default: 5]

clip-scores

Clips PCA scores from the beginning or end

Args:

pca_file (string): Path to PCA scores clip_samples (int): number of samples to clip from beginning or end from_end (bool): if true clip from end rather than beginning

Note that scores are modified *in place*.

```
cli clip-scores [OPTIONS] PCA_FILE CLIP_SAMPLES
```

Options

--from-end
[default: False]

Arguments

PCA_FILE
Required argument

CLIP_SAMPLES
Required argument

compute-changepoints

```
cli compute-changepoints [OPTIONS]
```

Options

-i, --input-dir <input_dir>
Directory to find h5 files [default: /Users/aymanzeine/Desktop/moseq/moseq2-pca/docs]

-o, --output-dir <output_dir>
Directory to store results [default: /Users/aymanzeine/Desktop/moseq/moseq2-pca/docs/_pca/]

--output-file <output_file>
Name of h5 file for storing pca results [default: changepoints]

--cluster-type <cluster_type>
Cluster type [default: local]

Options: local|slurm

--pca-file-components <pca_file_components>
Path to PCA components

--pca-file-scores <pca_file_scores>
Path to PCA results

--pca-path <pca_path>
Path to pca components [default: /components]

--neighbors <neighbors>
Neighbors to use for peak identification [default: 1]

--threshold <threshold>
Peak threshold to use for changepoints [default: 0.5]

-k, --klags <klags>
Lag to use for derivative calculation [default: 6]

-s, --sigma <sigma>
Standard deviation of gaussian smoothing filter [default: 3.5]

-d, --dims <dims>
Number of random projections to use [default: 300]

--fps <fps>
Fps (only used if no timestamps found) [default: 30]

--h5-path <h5_path>
Path to data in h5 files [default: /frames]

--h5-mask-path <h5_mask_path>
Path to log-likelihood mask in h5 files [default: /frames_mask]

--chunk-size <chunk_size>
Number of frames per chunk [default: 4000]

--config-file <config_file>
Path to configuration file

--dask-cache-path <dask_cache_path>
Path to spill data to disk for dask local scheduler [default: /Users/aymanzeine/moseq2_pca]

--visualize-results <visualize_results>
Visualize results [default: True]

-q, --queue <queue>
Cluster queue/partition for submitting jobs [default: debug]

-n, --nworkers <nworkers>
Number of workers [default: 10]

-c, --cores <cores>

Number of cores per worker [default: 1]

-p, --processes <processes>
Number of processes to run on each worker [default: 1]

-m, --memory <memory>
RAM usage per workers [default: 15GB]

-w, --wall-time <wall_time>
Wall time for workers [default: 06:00:00]

--timeout <timeout>
Time to wait for workers to initialize before proceeding (minutes) [default: 5]

train-pca

```
cli train-pca [OPTIONS]
```

Options

-i, --input-dir <input_dir>
Directory to find h5 files [default: /Users/aymanzeine/Desktop/moseq/moseq2-pca/docs]

--cluster-type <cluster_type>
Cluster type [default: local]

Options: local|slurm

-o, --output-dir <output_dir>
Directory to store results [default: /Users/aymanzeine/Desktop/moseq/moseq2-pca/docs/_pca]

--gaussfilter-space <gaussfilter_space>
Spatial filter for data (Gaussian) [default: 1.5, 1]

--gaussfilter-time <gaussfilter_time>
Temporal filter for data (Gaussian) [default: 0]

--medfilter-space <medfilter_space>
Median spatial filter [default: 0]

--medfilter-time <medfilter_time>
Median temporal filter [default: 0]

--missing-data
Use missing data PCA [default: False]

--missing-data-iters <missing_data_iters>
Missing data PCA iterations [default: 10]

--mask-threshold <mask_threshold>
Threshold for mask (missing data only) [default: -16]

--mask-height-threshold <mask_height_threshold>
Threshold for mask based on floor height [default: 5]

--min-height <min_height>
Min mouse height from floor (mm) [default: 10]

--max-height <max_height>
Max mouse height from floor (mm) [default: 100]

--tailfilter-size <tailfilter_size>
Tail filter size [default: 9, 9]

--tailfilter-shape <tailfilter_shape>
Tail filter shape [default: ellipse]

--use-fft
Use 2D fft [default: False]

--recon-pcs <recon_pcs>

Number of PCs to use for missing data reconstruction [default: 10]

--rank <rank>
Rank for compressed SVD (generally>>nPCS) [default: 50]

--output-file <output_file>
Name of h5 file for storing pca results [default: pca]

--chunk-size <chunk_size>
Number of frames per chunk [default: 4000]

--visualize-results <visualize_results>
Visualize results [default: True]

--config-file <config_file>
Path to configuration file

-d, --dask-cache-path <dask_cache_path>
Path to spill data to disk for dask local scheduler [default: /Users/aymanzeine/moseq2_pca]

--local-processes <local_processes>
Use processes with local scheduler [default: True]

-q, --queue <queue>
Cluster queue/partition for submitting jobs [default: debug]

-n, --nworkers <nworkers>
Number of workers [default: 10]

-c, --cores <cores>
Number of cores per worker [default: 1]

-p, --processes <processes>
Number of processes to run on each worker [default: 1]

-m, --memory <memory>
Total RAM usage per worker [default: 15GB]

-w, --wall-time <wall_time>
Wall time for workers [default: 06:00:00]

--timeout <timeout>
Time to wait for workers to initialize before proceeding (minutes) [default: 5]

GUI Module

moseq2_pca.gui.**apply_pca_command** (input_dir, index_file, config_file, output_dir, output_file, output_directory=None)
Compute PCA Scores given trained PCA using Jupyter Notebook.

Parameters:
- **input_dir (str)** (*path to directory containing training data*)
- **index_file (str)** (*path to index file.*)
- **config_file (str)** (*path to config file*)
- **output_dir (str)** (*path to output pca directory*)
- **output_file (str)** (*name of output pca file.*)
- **output_directory (str)** (*alternative output directory path*)

Returns: **(str)**

Return type: success string.

moseq2_pca.gui.**compute_changepoints_command** (input_dir, config_file, output_dir, output_file, output_directory=None)
Compute Changepoint distribution using Jupyter Notebook.

Parameters:
- **input_dir (str)** (*path to directory containing training data*)
- **config_file (str)** (*path to config file*)
- **output_dir (str)** (*path to output pca directory*)
- **output_file (str)** (*name of output pca file.*)
- **output_directory (str)** (*alternative output directory path*)

Returns: **(str)**

Return type: success string.

`moseq2_pca.gui.`**`train_pca_command`** (input_dir, config_file, output_dir, output_file, output_directory=None)
Train PCA through Jupyter notebook, and updates config file.

Parameters:
- **input_dir (str)** (*path to directory containing training data*)
- **config_file (str)** (*path to config file*)
- **output_dir (str)** (*path to output pca directory*)
- **output_file (str)** (*name of output pca file.*)
- **output_directory (str)** (*alternative output directory path*)

Returns:

Return type: None

Utilities Module

`moseq2_pca.util.`**`clean_frames`** (frames, medfilter_space=None, gaussfilter_space=None, medfilter_time=None, gaussfilter_time=None, detrend_time=None, tailfilter=None, tail_threshold=5)
Filters spatial/temporal noise from frames using Median and Gaussian filters, given kernel sizes for each respective requested filter.

Parameters:
- **frames (3D numpy array)** (*frames to filter.*)
- **medfilter_space (list)** (*median spatial filter kernel.*)
- **gaussfilter_space (list)** (*gaussian spatial filter kernel.*)
- **medfilter_time (list)** (*median temporal filter.*)
- **gaussfilter_time (list)** (*gaussian temporal filter.*)
- **detrend_time (int)** (*number of frames to lag for.*)
- **tailfilter (int)** (*size of tail-filter kernel.*)
- **tail_threshold (int)** (*threshold value to use for tail filtering*)

Returns: **out (3D numpy array)**

Return type: filtered frames.

`moseq2_pca.util.`**`command_with_config`** (config_file_param_name)

`moseq2_pca.util.`**`gauss_smooth`** (signal, win_length=None, sig=1.5, kernel=None)
Perform Gaussian Smoothing on a 1D signal.

Parameters:
- **signal (1d numpy array)** (*signal to perform smoothing*)
- **win_length (int)** (*window_size for gaussian kernel filter*)
- **sig (float)** (*variance of 1d gaussian kernel.*)
- **kernel (tuple)** (*kernel size to use for smoothing*)

Returns: **result (1d numpy array)**

Return type: smoothed signal

moseq2_pca.util.**gaussian_kernel1d** (n=None, sig=3)
Get 1D gaussian kernel.

Parameters:
- **n (int)** (*number of points to use.*)
- **sig (int)** (*variance of kernel to use.*)

Returns: **kernel (1d array)**

Return type: 1D numpy kernel.

moseq2_pca.util.**get_changepoints** (scores, k=5, sigma=3, peak_height=0.5, peak_neighbors=1, baseline=True, timestamps=None)
Compute changepoints distribution and CP Curve.

Parameters:
- **scores (3D numpy array)** (*nframes * r * c*)
- **k (int)** (*klags - Lag to use for derivative calculation.*)
- **sigma (int)** (*Standard deviation of gaussian smoothing filter.*)
- **peak_height (float)** (*user-defined peak Changepoint length.*)
- **peak_neighbors (int)** (*number of peaks in the CP curve.*)
- **baseline (bool)** (*normalize data.*)
- **timestamps (array)** (*loaded timestamps.*)

Returns: **cps (numpy array)** (*array of values for CP curve*) **normed_df (numpy array)** (*array of values for bar plot*)

moseq2_pca.util.**get_metadata_path** (h5file)
Return path within h5 file that contains the kinect extraction metadata.

Parameters: **h5file (str)** (*path to h5 file.*)

Returns: **(str)**

Return type: path to acquistion metadata within h5 file.

moseq2_pca.util.**get_rps** (frames, rps=600, normalize=True)
Get random projections of frames.

Parameters:
- **frames (2D or 3D numpy array)** (*Frames to get dimensions from.*)
- **rps (int)** (*Number of random projections.*)
- **normalize (bool)** (*indicates whether to normalize frames.*)

Returns: **rproj (2D or 3D numpy array)**

Return type: Computed random projections with same shape as frames

moseq2_pca.util.**get_timestamp_path** (h5file)
Return path within h5 file that contains the kinect timestamps

Parameters: **h5file (str)** (*path to h5 file.*)

Returns: **(str)**

Return type: path to metadata timestamps within h5 file

moseq2_pca.util.**initialize_dask** (nworkers=50, processes=1, memory='4GB', cores=1, wall_time='01:00:00', queue='debug', local_processes=False, cluster_type='local', scheduler='distributed', timeout=10, cache_path='/Users/aymanzeine/moseq2_pca', **kwargs)
Initialize dask client, cluster, workers, etc.

Parameters:
- **nworkers (int)** (*number of dask workers to initialize*)
- **processes (int)** (*number of processes per worker*)
- **memory (str)** (*amount of memory to allocate to dask cluster*)
- **cores (int)** (*number of cores to use.*)
- **wall_time (str)** (*amount of time to allow program to run*)
- **queue (str)** (*logging mode*)
- **local_processes (bool)** (*indicate whether the processes are local*)
- **cluster_type (str)** (*indicate what cluster to use*)
- **scheduler (str)** (*indicate what scheduler to use*)
- **timeout (int)** (*number of worker timeouts to allow*)
- **cache_path (str or Pathlike)** (*path to store cached data*)
- **kwargs** (*extra keyward arguments*)

Returns: **client (dask Client)** (*initialized Client*) **cluster (dask Cluster)** (*initialized Cluster*) **workers (dask Workers)** (*intialized workers*) **cache (dask Chest)** (*initialized Chest (cache) object*)

`moseq2_pca.util.`**`insert_nans`** (timestamps, data, fps=30)
Fills NaN values with 0 in timestamps.

Parameters:
- **timestamps (1D array)** (*timestamp time-strs*)
- **data (1D array)** (*timestamp values*)
- **fps (int)** (*frames per second*)

Returns: **filled_data (1D array)** (*filled missing timestamp values.*) **data_idx (1D array)** (*indices of inserted 0s*) **filled_timestamps (1D array)** (*filled timestamp-strs*)

`moseq2_pca.util.`**`read_yaml`** (yaml_file)
Reads yaml file and returns dictionary representation of file contents.

Parameters: **yaml_file (str)** (*path to yaml file*)
Returns: **return_dict (dict)**
Return type: dict of yaml file contents

`moseq2_pca.util.`**`recursive_find_h5s`** (root_dir='/Users/aymanzeine/Desktop/moseq/moseq2-pca/docs', ext='.h5', yaml_string='{}.yaml')
Recursively find h5 files, along with yaml files with the same basename

Parameters:
- **root_dir (str or os.Pathlike)** (*path to directory to start recursive search*)
- **ext (str)** (*extension to search for, e.g. .h5*)
- **yaml_string (str)** (*a format to use to name yaml files*)

Returns: **h5s (list)** (*list of h5 file paths*) **dicts (list)** (*list of metadata file paths*) **yamls (list)** (*list of yaml file paths*)

`moseq2_pca.util.`**`recursively_load_dict_contents_from_group`** (h5file, path)
Reads all contents from h5 and returns them in a nested dict object.

Parameters:
- **h5file (str)** (*path to h5 file*)
- **path (str)** (*path to group within h5 file*)

Returns: **ans (dict)**
Return type: dictionary of all h5 group contents

`moseq2_pca.util.`**`select_strel`** (string='e', size=10, 10)
Selects Structuring Element Shape

Parameters:
- **string (str)** (*e for Ellipse, r for Rectangle*)
- **size (tuple)** (*size of StructuringElement*)

Returns: **strel (cv2.StructuringElement)**

Return type: returned StructuringElement with specified size.

`moseq2_pca.util.`**`shutdown_dask`** (scheduler)
Graceful shutdown dask scheduler. source: https://github.com/dask/distributed/issues/1703#issuecomment-361291492

Parameters: **scheduler (dask Scheduler)** (*scheduler to shutdown.*)

Returns:

Return type: None

Visualization Module

`moseq2_pca.viz.`**`changepoint_dist`** (cps, headless=False)
Creates bar plot describing computed Changepoint Distribution.

Parameters:
- **cps (np.ndarray)** (*changepoints to graph*)
- **headless (bool)** (*trim first element in PC list*)

Returns: **plt (plt.figure)** (*figure to save/graph*) **ax (plt.ax)** (*figure axis variable*)

`moseq2_pca.viz.`**`display_components`** (components, cmap='gray', headless=False)
Creates grid of computed Principal Components.

Parameters:
- **components (np.ndarray)** (*components to graph*)
- **cmap (str)** (*color map to use*)
- **headless (bool)** (*trim first element in PC list*)

Returns: **plt (plt.figure)** (*figure to save/graph*) **ax (plt.ax)** (*figure axis variable*)

`moseq2_pca.viz.`**`scree_plot`** (explained_variance_ratio, headless=False)
Creates Scree plot describing principal components.

Parameters:
- **explained_variance_ratio (np.array)** (*explained variance ratio of each principal component*)
- **headless (bool)** (*trim first element in PC list*)

Returns: **plt (plt.figure)**

Return type: figure to save/graph

Indices and tables

- **`genindex`**
- **`modindex`**
- **`search`**

Index

Symbols

--chunk-size <chunk_size> cli-apply-pca command line option
cli-compute-changepoints command line option
cli-train-pca command line option
--cluster-type <cluster_type> cli-apply-pca command line option
cli-compute-changepoints command line option
cli-train-pca command line option
--config-file <config_file> cli-apply-pca command line option
cli-compute-changepoints command line option
cli-train-pca command line option
--cores <cores> cli-apply-pca command line option
cli-compute-changepoints command line option
cli-train-pca command line option
-dask-cache-path <dask_cache_path> cli-apply-pca command line option
cli-compute-changepoints command line option
cli-train-pca command line option
--detrend-window <detrend_window> cli-apply-pca command line option
--dims <dims> cli-compute-changepoints command line option
--fill-gaps <fill_gaps> cli-apply-pca command line option
--fps <fps> cli-apply-pca command line option
cli-compute-changepoints command line option
--from-end cli-clip-scores command line option
--gaussfilter-space <gaussfilter_space> cli-train-pca command line option
--gaussfilter-time <gaussfilter_time> cli-train-pca command line option
--h5-mask-path <h5_mask_path> cli-apply-pca command line option
cli-compute-changepoints command line option
--h5-path <h5_path> cli-apply-pca command line option
cli-compute-changepoints command line option
--input-dir <input_dir> cli-apply-pca command line option
cli-compute-changepoints command line option
cli-train-pca command line option
--klags <klags> cli-compute-changepoints command line option
--local-processes <local_processes> cli-train-pca command line option
--mask-height-threshold <mask_height_threshold> cli-train-pca command line option
--mask-threshold <mask_threshold> cli-train-pca command line option
--max-height <max_height> cli-train-pca command line option
--medfilter-space <medfilter_space> cli-train-pca command line option
--medfilter-time <medfilter_time> cli-train-pca command line option
--memory <memory> cli-apply-pca command line option
cli-compute-changepoints command line option
cli-train-pca command line option
--min-height <min_height> cli-train-pca command line option
--missing-data cli-train-pca command line option
--missing-data-iters <missing_data_iters> cli-train-pca command line option
--neighbors <neighbors> cli-compute-changepoints command line option

--nworkers <nworkers> cli-apply-pca command line option
cli-compute-changepoints command line option
cli-train-pca command line option
--output-dir <output_dir> cli-apply-pca command line option
cli-compute-changepoints command line option
cli-train-pca command line option
--output-file <output_file> cli-apply-pca command line option
cli-compute-changepoints command line option
cli-train-pca command line option
--pca-file <pca_file> cli-apply-pca command line option
-file-components <pca_file_components> cli-compute-changepoints command line option
-pca-file-scores <pca_file_scores> cli-compute-changepoints command line option
--pca-path <pca_path> cli-apply-pca command line option
cli-compute-changepoints command line option
--processes <processes> cli-apply-pca command line option
cli-compute-changepoints command line option
cli-train-pca command line option
--queue <queue> cli-apply-pca command line option
cli-compute-changepoints command line option
cli-train-pca command line option
--rank <rank> cli-train-pca command line option
--recon-pcs <recon_pcs> cli-train-pca command line option
--sigma <sigma> cli-compute-changepoints command line option
--tailfilter-shape <tailfilter_shape> cli-train-pca command line option
--tailfilter-size <tailfilter_size> cli-train-pca command line option
--threshold <threshold> cli-compute-changepoints command line option
--timeout <timeout> cli-apply-pca command line option
cli-compute-changepoints command line option
cli-train-pca command line option
--use-fft cli-train-pca command line option
--visualize-results <visualize_results> cli-compute-changepo command line option
cli-train-pca comm line option
--wall-time <wall_time> cli-apply-pca command line option
cli-compute-changepoints command line option
cli-train-pca command line option
-C cli-apply-pca command line option
cli-compute-changepoints command line option
cli-train-pca command line option
-d cli-apply-pca command line option
cli-compute-changepoints command line option
cli-train-pca command line option
-i cli-apply-pca command line option
cli-compute-changepoints command line option
cli-train-pca command line option
-k cli-compute-changepoints command line option
-m cli-apply-pca command line option
cli-compute-changepoints command line option
cli-train-pca command line option
-n cli-apply-pca command line option
cli-compute-changepoints command line option
cli-train-pca command line option

-o cli-apply-pca command line option
cli-compute-changepoints command line option
cli-train-pca command line option
-p cli-apply-pca command line option
cli-compute-changepoints command line option
cli-train-pca command line option
-q cli-apply-pca command line option
cli-compute-changepoints command line option
cli-train-pca command line option
-s cli-compute-changepoints command line option
-w cli-apply-pca command line option
cli-compute-changepoints command line option
cli-train-pca command line option

A

apply_pca_command() (in module moseq2_pca.gui)
apply_pca_dask() (in module moseq2_pca.pca.util)
apply_pca_local() (in module moseq2_pca.pca.util)
apply_pca_wrapper() (in module moseq2_pca.helpers.wrappers)

C

changepoint_dist() (in module moseq2_pca.viz)
(moseq2_pca.tests.unit_tests.test_viz.TestViz method)
clean_frames() (in module moseq2_pca.util)
cli-apply-pca command line option
--chunk-size <chunk_size>
--cluster-type <cluster_type>
--config-file <config_file>
--cores <cores>
--dask-cache-path <dask_cache_path>
--detrend-window <detrend_window>
--fps <fps>
--h5-mask-path <h5_mask_path>
--h5-path <h5_path>
--input-dir <input_dir>
--memory <memory>
--nworkers <nworkers>
--output-dir <output_dir>
--output-file <output_file>
--pca-file <pca_file>
--pca-path <pca_path>
--processes <processes>
--queue <queue>
--timeout <timeout>
--wall-time <wall_time>
-c
-d
-i
-m
-n
-o
-p
-q
-w
cli-clip-scores command line option
--from-end
CLIP_SAMPLES
PCA_FILE
cli-compute-changepoints command line option
--chunk-size <chunk_size>
--cluster-type <cluster_type>
--config-file <config_file>
--cores <cores>
--dask-cache-path <dask_cache_path>
--dims <dims>
--fps <fps>
--h5-mask-path <h5_mask_path>
--h5-path <h5_path>
--input-dir <input_dir>
--klags <klags>
--memory <memory>
--neighbors <neighbors>
--nworkers <nworkers>
--output-dir <output_dir>

--output-file <output_file>
--pca-file-components <pca_file_components>
--pca-file-scores <pca_file_scores>
--pca-path <pca_path>
--processes <processes>
--queue <queue>
--sigma <sigma>
--threshold <threshold>
--timeout <timeout>
--visualize-results <visualize_results>
--wall-time <wall_time>
-c
-d
-i
-k
-m
-n
-o
-p
-q
-s
-w

cli-train-pca command line option

--chunk-size <chunk_size>
--cluster-type <cluster_type>
--config-file <config_file>
--cores <cores>
--dask-cache-path <dask_cache_path>
--gaussfilter-space <gaussfilter_space>
--gaussfilter-time <gaussfilter_time>
--input-dir <input_dir>
--local-processes <local_processes>
--mask-height-threshold <mask_height_threshold>
--mask-threshold <mask_threshold>
--max-height <max_height>
--medfilter-space <medfilter_space>
--medfilter-time <medfilter_time>
--memory <memory>
--min-height <min_height>
--missing-data
--missing-data-iters <missing_data_iters>
--nworkers <nworkers>
--output-dir <output_dir>
--output-file <output_file>
--processes <processes>
--queue <queue>
--rank <rank>
--recon-pcs <recon_pcs>
--tailfilter-shape <tailfilter_shape>
--tailfilter-size <tailfilter_size>
--timeout <timeout>
--use-fft
--visualize-results <visualize_results>
--wall-time <wall_time>
-c
-d
-i
-m
-n
-o
-p
-q
-w

CLIP_SAMPLES

cli-clip-scores command line option

command_with_config() (in module moseq2_pca.util)

compute_changepoints_command() (in module moseq2_pca.gui)

compute_changepoints_wrapper() (in module moseq2_pca.helpers.wrappers)

D

display_components() (in module moseq2_pca.viz)

G

gauss_smooth() (in module moseq2_pca.util)

gaussian_kernel1d() (in module moseq2_pca.util)

get_changepoints() (in module moseq2_pca.util)

get_changepoints_dask() (in module moseq2_pca.pca.util)

get_metadata_path() (in module moseq2_pca.util)

get_pca_yaml_data() (in module moseq2_pca.helpers.data)

get_rps() (in module moseq2_pca.util)

get_timestamp_path() (in module moseq2_pca.util)

I

initialize_dask() (in module moseq2_pca.util)

insert_nans() (in module moseq2_pca.util)

L

load_pcs_for_cp() (in module moseq2_pca.helpers.data)

M

mask_data() (in module moseq2_pca.pca.util)

module

- moseq2_pca.gui
- moseq2_pca.helpers.data
- moseq2_pca.helpers.wrappers
- moseq2_pca.pca.util
- moseq2_pca.tests.integration_tests.test_cli
- moseq2_pca.tests.integration_tests.test_gui
- moseq2_pca.tests.unit_tests.test_pca_util
- moseq2_pca.tests.unit_tests.test_util
- moseq2_pca.tests.unit_tests.test_viz
- moseq2_pca.util
- moseq2_pca.viz

moseq2_pca.gui

- module

moseq2_pca.helpers.data

- module

moseq2_pca.helpers.wrappers

- module

moseq2_pca.pca.util

- module

moseq2_pca.tests.integration_tests.test_cli

- module

moseq2_pca.tests.integration_tests.test_gui

- module

moseq2_pca.tests.unit_tests.test_pca_util

- module

moseq2_pca.tests.unit_tests.test_util

- module

moseq2_pca.tests.unit_tests.test_viz

- module

moseq2_pca.util

- module

moseq2_pca.viz

- module

P

PCA_FILE

- cli-clip-scores command line option

R

read_yaml() (in module moseq2_pca.util)

recursive_find_h5s() (in module moseq2_pca.util)

recursively_load_dict_contents_from_group() (in module moseq2_pca.util)

S

scree_plot() (in module moseq2_pca.viz)

select_strel() (in module moseq2_pca.util)

setup_cp_command() (in module moseq2_pca.helpers.data)

shutdown_dask() (in module moseq2_pca.util)

T

test_apply_pca() (moseq2_pca.tests.integration_tests.test_cli.TestCli method)

test_apply_pca_command() (moseq2_pca.tests.integration_tests.test_gui.TestGUI method)

test_clean_frames() (moseq2_pca.tests.unit_tests.test_util.TestUtils method)

test_clip_scores() (moseq2_pca.tests.integration_tests.test_cli.TestCli method)

test_compute_changepoints() (moseq2_pca.tests.integration_tests.test_cli.TestCli method)

test_compute_changepoints_command() (moseq2_pca.tests.integration_tests.test_gui.TestGUI method)

test_display_components() (moseq2_pca.tests.unit_tests.test_viz.TestViz method)

test_gauss_smooth() (moseq2_pca.tests.unit_tests.test_util.TestUtils method)

test_gaussian_kernel1d() (moseq2_pca.tests.unit_tests.test_util.TestUtils method)

test_get_changepoints() (moseq2_pca.tests.unit_tests.test_util.TestUtils method)

test_get_metadata_path() (moseq2_pca.tests.unit_tests.test_util.TestUtils method)

test_get_rps() (moseq2_pca.tests.unit_tests.test_util.TestUtils method)

test_get_rsp_dask() (moseq2_pca.tests.unit_tests.test_util.TestUtils method)

test_get_timestamp_path() (moseq2_pca.tests.unit_tests.test_util.TestUtils method)

test_initialize_dask() (moseq2_pca.tests.unit_tests.test_util.TestUtils method)

test_insert_nans() (moseq2_pca.tests.unit_tests.test_util.TestUtils method)

test_mask_data() (moseq2_pca.tests.unit_tests.test_pca_util.TestPCAUtils method)

test_read_yaml() (moseq2_pca.tests.unit_tests.test_util.TestUtils method)

test_recursive_find_h5s() (moseq2_pca.tests.unit_tests.test_util.TestUtils method)

test_scree_plot() (moseq2_pca.tests.unit_tests.test_viz.TestViz method)

test_select_strel() (moseq2_pca.tests.unit_tests.test_util.TestUtils method)

test_train_pca() (moseq2_pca.tests.integration_tests.test_cli.TestCli method)

test_train_pca_command() (moseq2_pca.tests.integration_tests.test_gui.TestGUI method)

test_train_pca_dask() (moseq2_pca.tests.unit_tests.test_pca_util.TestPCAUtils method)

TestCli (class in moseq2_pca.tests.integration_tests.test_cli)

TestGUI (class in moseq2_pca.tests.integration_tests.test_gui)

TestPCAUtils (class in moseq2_pca.tests.unit_tests.test_pca_util)

TestUtils (class in moseq2_pca.tests.unit_tests.test_util)

TestViz (class in moseq2_pca.tests.unit_tests.test_viz)

train_pca_command() (in module moseq2_pca.gui)

train_pca_dask() (in module moseq2_pca.pca.util)

train_pca_wrapper() (in module moseq2_pca.helpers.wrappers)

Python Module Index

m

moseq2_pca

moseq2_pca.gui

moseq2_pca.helpers.data

moseq2_pca.helpers.wrappers

moseq2_pca.pca.util

moseq2_pca.tests.integration_tests.test_cli

moseq2_pca.tests.integration_tests.test_gui

moseq2_pca.tests.unit_tests.test_pca_util

moseq2_pca.tests.unit_tests.test_util

moseq2_pca.tests.unit_tests.test_viz

moseq2_pca.util

moseq2_pca.viz