

# Python Documentation

version

November 25, 2020



# Contents

<b>Welcome to moseq2-pca's documentation!</b>	<b>1</b>
moseq2_pca package	1
CLI Module	1
moseq2-pca	1
apply-pca	1
clip-scores	2
compute-changepoints	2
train-pca	3
GUI Module	5
Utilities Module	6
Visualization Module	9
Subpackages	10
moseq2_pca.helpers package	10
Helpers - Data Module	10
Helpers - Wrapper Module	10
moseq2_pca.pca package	12
PCA - Utilities Module	12
<b>Indices and tables</b>	<b>15</b>
<b>Index</b>	<b>17</b>
<b>Python Module Index</b>	<b>23</b>



# Welcome to moseq2-pca's documentation!

## moseq2\_pca package

### CLI Module

#### *moseq2-pca*

```
moseq2-pca [OPTIONS] COMMAND [ARGS]...
```

#### Options

##### **--version**

Show the version and exit. [default: False]

#### *apply-pca*

Computes PCA Scores of extraction data given a pre-trained PCA

```
moseq2-pca apply-pca [OPTIONS]
```

#### Options

##### **--chunk-size** <chunk\_size>

Number of frames per chunk [default: 4000]

##### **--h5-mask-path** <h5\_mask\_path>

Path to log-likelihood mask in h5 files [default: /frames\_mask]

##### **--h5-path** <h5\_path>

Path to data in h5 files [default: /frames]

##### **--config-file** <config\_file>

Path to configuration file

##### **-o, --output-dir** <output\_dir>

Directory to store results [default: /Users/aymanzeine/Desktop/moseq/moseq2-pca/docs/\_pca]

##### **-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

##### **--timeout** <timeout>

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

##### **-w, --wall-time** <wall\_time>

Wall time for workers [default: 06:00:00]

##### **-m, --memory** <memory>

Total RAM usage per worker [default: 15GB]

##### **-p, --processes** <processes>

Number of processes to run on each worker [default: 1]

##### **-c, --cores** <cores>

Number of cores per worker [default: 1]

##### **-n, --nworkers** <nworkers>

Number of workers [default: 10]

##### **-q, --queue** <queue>

Cluster queue/partition for submitting jobs [default: debug]

Welcome to moseq2-pca's documentation!

```
--dask-port <dask_port>
  Port to access dask dashboard [default: 8787]

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

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

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

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

--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]

-v, --verbose
  Print sessions as they are being loaded. [default: False]

--overwrite-pca-apply <overwrite_pca_apply>
  Used to bypass the pca overwrite question. If True: skip question, run automatically [default: False]
```

### **clip-scores**

Clips specified number of frames from PCA scores at the beginning or end

```
moseq2-pca clip-scores [OPTIONS] PCA_FILE CLIP_SAMPLES
```

#### Options

```
--from-end
  [default: False]
```

#### Arguments

**PCA\_FILE**  
Required argument

**CLIP\_SAMPLES**  
Required argument

### **compute-changepoints**

Computes the Model-Free Syllable Changepoints based on the PCA/PCA\_Scores

```
moseq2-pca compute-changepoints [OPTIONS]
```

#### Options

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

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

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

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

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

Welcome to moseq2-pca's documentation!

**-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

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

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

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

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

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

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

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

**--dask-port** <dask\_port>  
Port to access dask dashboard [default: 8787]

**-d, --dask-cache-path** <dask\_cache\_path>  
Path to spill data to disk for dask local scheduler [default: /Users/aymanzeine/moseq2\_pca]

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

**--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]

**-v, --verbose**  
Print sessions as they are being loaded. [default: False]

### ***train-pca***

Trains PCA on all extracted results (h5 files) in input directory

```
moseq2-pca train-pca [OPTIONS]
```

### Options

**--chunk-size** <chunk\_size>  
Number of frames per chunk [default: 4000]

**--h5-mask-path** <h5\_mask\_path>  
Path to log-likelihood mask in h5 files [default: /frames\_mask]

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

**--config-file** <config\_file>  
Path to configuration file

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

**-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

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

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

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

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

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

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

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

**--dask-port** <dask\_port>  
Port to access dask dashboard [default: 8787]

**-d, --dask-cache-path** <dask\_cache\_path>  
Path to spill data to disk for dask local scheduler [default: /Users/aymanzeine/moseq2\_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; will be automatically set to True if cable-filter-iters > 1 from the extract step. [default: False]

**--missing-data-iters** <missing\_data\_iters>  
Missing data PCA iterations [default: 10]



Welcome to moseq2-pca's documentation!

```
--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: 25]

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

--local-processes <local_processes>
  Used with a local cluster. If True: use processes, If False: use threads [default: False]

--overwrite-pca-train <overwrite_pca_train>
  Used to bypass the pca overwrite question. If True: skip question, run automatically [default: False]
```

## GUI Module

GUI front-end operations. This module contains all the functionality and configurable parameters users can alter to most accurately process their data.

Note: These functions perform jupyter notebook specific preprocessing, loads in corresponding parameters from the CLI functions, then call the corresponding wrapper function with the given input parameters.

`moseq2_pca.gui.apply_pca_command(progress_paths, output_file)`  
Compute PCA Scores given trained PCA using Jupyter Notebook.

**Parameters:**

- **progress\_paths (dict)** (*dictionary containing notebook progress paths*)
- **output\_file (str)** (*name of output pca file.*)

**Returns:** (str)

**Return type:** success string.

`moseq2_pca.gui.compute_changepoints_command(input_dir, progress_paths, output_file)`  
Compute Changepoint distribution using Jupyter Notebook.

**Parameters:**

- **input\_dir (str)** (*path to directory containing training data*)
- **progress\_paths (dict)** (*dictionary containing notebook progress paths*)
- **output\_file (str)** (*name of output pca file.*)

**Returns:** (str)

**Return type:** success string.

`moseq2_pca.gui.train_pca_command(progress_paths, output_dir, output_file)`  
Train PCA through Jupyter notebook, and updates config file.

**Parameters:**

- **progress\_paths (dict)** (*dictionary containing notebook progress paths*)
- **output\_dir (str)** (*path to output pca directory*)
- **output\_file (str)** (*name of output pca file.*)

**Returns:**

**Return type:** None

## Utilities Module

**Utility and helper functions for traversing directories to find and read files, filtering operations,**  
Dask initialization, and changepoint helper functions.

`moseq2_pca.util.check_timestamps(h5s)`

Helper function to determine whether timestamps and/or metadata is missing from extracted files. Function will emit a warning if either pieces of data are missing.

**Parameters:** **h5s (list)** (*List of paths to all extracted h5 files.*)

**Returns:**

**Return type:** None

`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.close_dask(client, cluster, timeout)`

Shuts down the Dask client and cluster. Dumps all cached data.

**Parameters:**

- **client (Dask Client)** (*Client object*)
- **cluster (dask Cluster)** (*initialized Cluster*)
- **timeout (int)** (*Time to wait for client to close gracefully (minutes)*)

**Returns:**

**Return type:** None

`moseq2_pca.util.command_with_config(config_file_param_name)`

Provides a cli helper function to assign variables from a config file

`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 and its corresponding distribution. Changepoints describe the magnitude of frame-to-frame changes of mouse pose.

**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 (2D numpy array)** (*array of changepoint values*) **normed\_df (1D numpy array)** (*array of values for bar plot*)

`moseq2_pca.util.get_env_cpu_and_mem` ()

Reads current system environment and returns the amount of available memory and CPUs to allocate to the created cluster.

**Returns:** **mem (float)** (*Optimal number of memory (in bytes) to allocate to initialized dask cluster*) **cpu (int)** (*Optimal number of CPUs to allocate to dask*)

`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 acquisition 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 the random projections.*)

**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.h5_to_dict(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.initialize_dask(nworkers=50, processes=1, memory='4GB', cores=1, wall_time='01:00:00', queue='debug', local_processes=False, cluster_type='local', timeout=10, cache_path='/Users/aymanzeine/moseq2_pca', dashboard_port='8787', data_size=None, **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)** (*flag to use processes or threads when using a local cluster*)
- **cluster\_type (str)** (*indicate what cluster to use (local or slurm)*)
- **scheduler (str)** (*indicate what scheduler to use*)
- **timeout (int)** (*how many minutes to wait for workers to initialize*)
- **cache\_path (str or Pathlike)** (*path to store cached data*)
- **dashboard\_port (str)** (*port number to find dask statistics*)
- **data\_size (float)** (*size of the dask array in number of bytes.*)
- **kwargs** (*extra keyword arguments*)

**Returns:** **client (dask Client)** (*initialized Client*) **cluster (dask Cluster)** (*initialized Cluster*) **workers (dask Workers)** (*initialized workers*)

`moseq2_pca.util.insert_nans(timestamps, data, fps=30)`

Fills NaN values with 0 in timestamps.

**Parameters:**

- **timestamps (1D array)** (*timestamp values*)
- **data (1D or 2D array)** (*additional data to fill with NaN values - can be PC scores*)
- **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-strings*)

`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

Welcome to moseq2-pca's documentation!

`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)** (path to base directory to begin recursive search in.)
- **ext (str)** (extension to search for)
- **yaml\_string (str)** (string for filename formatting when saving data)

**Returns:** **h5s (list)** (list of found h5 files) **dicts (list)** (list of found metadata files) **yamls (list)** (list of found yaml files)

`moseq2_pca.util.select_strel (string='e', size=(10,10))`

**Selects Structuring Element Shape. Accepts shapes ('ellipse', 'rectangle'), if neither are given then 'ellipse' is used.**

**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.set_dask_config (memory={'pause': False, 'spill': False, 'target': 0.85,  
'terminate': 0.95})`

Sets initial dask configuration parameters

**Parameters:** **memory (dict)**

## Visualization Module

Visualization operations for plotting computed PCs, a Scree Plot, and the Changepoint PDF histogram.

`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 (2D 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.plot_pca_results (output_dict, save_file, output_dir)`

Convenience function to graph and save Trained PCA results.

**Parameters:**

- **output\_dict (dict)** (Dict object containing PCA training results)
- **save\_file (str)** (Path to write images to.)
- **output\_dir (str)** (Directory containing logger)

**Returns:**

**Return type:** None

`moseq2_pca.viz.scree_plot (explained_variance_ratio, headless=False)`

Creates Scree plot describing principal components.

**Parameters:**

- **explained\_variance\_ratio** (1D 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

## Subpackages

### moseq2\_pca.helpers package

#### Helpers - Data Module

Helper functions for reading files and directories in preparation for changepoint analysis or apply pca.

moseq2\_pca.helpers.data.get\_pca\_paths (config\_data, output\_dir)

Helper function for changepoints\_wrapper to perform data-path existence checks. Returns paths to saved pre-trained PCA components and PCA Scores files.

**Parameters:**

- **config\_data** (dict) (*dict of relevant PCA parameters (image filtering etc.)*)
- **output\_dir** (str) (*path to directory to store PCA data*)

**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*)

moseq2\_pca.helpers.data.get\_pca\_yaml\_data (pca\_yaml)

Reads PCA yaml file and returns enclosed 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 for Model-free Changepoint Analysis.

**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*) **workers** (dask Workers) (*initialized workers or None if cluster\_type = 'local'*) **missing\_data** (bool) (*Indicates whether to use mask\_params*) **mask\_params** (dict) (*Mask parameters to use when computing CPs*)

#### Helpers - Wrapper Module

Wrapper functions for all functionality included in MoSeq2-PCA that is accessible via CLI or GUI. Each wrapper function executes the functionality from end-to-end given it's dependency parameters are inputted. (See CLI Click parameters)

moseq2\_pca.helpers.wrappers.apply\_pca\_wrapper (input\_dir, config\_data, output\_dir, output\_file)

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)
- **kwargs (dict)** (dictionary containing loaded h5s, yamls and dicts found in given input\_dir)

**Returns:** **config\_data (dict)**

**Return type:** updated config\_data variable to write back in GUI API

`moseq2_pca.helpers.wrappers.clip_scores_wrapper` (pca\_file, clip\_samples, from\_end=False)  
Clips PCA scores from the beginning or end. Note that scores are modified *in place*.

**Parameters:**

- **pca\_file (str)** (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)

`moseq2_pca.helpers.wrappers.compute_changepoints_wrapper` (input\_dir, config\_data, output\_dir, output\_file)

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)
- **kwargs (dict)** (dictionary containing loaded h5s, yamls and dicts found in given input\_dir)

**Returns:** **config\_data (dict)**

**Return type:** updated config\_data variable to write back in GUI API

`moseq2_pca.helpers.wrappers.load_and_check_data` (input\_dir, output\_dir)

Executes initialization functionality that is common among all 3 PCA related operations. Function will load relevant h5 and yaml files found in given input directory, then check for timestamps and warn the user if they are missing.

**Parameters:**

- **input\_dir (str)** (input directory containing h5 files to find)
- **output\_dir (str)** (directory name to save pca results)
- **changepoints (bool)** (boolean for whether to find data from the aggregate\_results directory)

**Returns:** **output\_dir (str)** (absolute output directory path) **h5s (list)** (list of found h5 files) **yamls (list)** (list of corresponding yaml files) **dicts (list)** (list of corresponding metadata.json files)

`moseq2_pca.helpers.wrappers.train_pca_wrapper` (input\_dir, config\_data, output\_dir, output\_file)

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)
- **kwargs (dict)** (dictionary containing loaded h5s, yamls and dicts found in given input\_dir)

**Returns:** **config\_data (dict)**

**Return type:** updated config\_data variable to write back in GUI API

## *moseq2\_pca.pca package*

### *PCA - Utilities Module*

Utility functions for all PCA-related operations.

`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`, `h5_path='/frames'`, `h5_mask_path='/frames_mask'`, `verbose=False`)

Multiply input frame data by the transpose of the given PCs 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*)
- **h5\_path (str)** (*path to frames within selected h5 file (default: '/frames')*)
- **h5\_mask\_path (str)** (*path to masked frames within selected h5 file (default: '/frames\_mask')*)
- **verbose (bool)** (*print session names as they are being loaded.*)

**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`, `h5_path='/frames'`, `h5_mask_path='/frames_mask'`, `verbose=False`)

Multiply input frame data by the transpose of the given PCs 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)
- **h5\_path (str)** (path to frames within selected h5 file (default: '/frames'))
- **h5\_mask\_path (str)** (path to masked frames within selected h5 file (default: '/frames\_mask'))
- **verbose (bool)** (print session names as they are being loaded.)

**Returns:**

**Return type:** None

`moseq2_pca.pca.util.compute_explained_variance(s, nsamples, total_var)`

Computes the explained variance and explained variance ratio contributed by each computed Principal Component.

**Parameters:**

- **s (1d array)** (computed singular values.)
- **nsamples (int)** (number of included samples.)
- **total\_var (float)** (total variance captured by principal components.)

**Returns:** **explained\_variance (1d-array)** (list of floats denoting the explained variance per PC.)  
**explained\_variance\_ratio (1d-array)** (list of floats denoting the explained variance ratios per PC.)

`moseq2_pca.pca.util.compute_svd(dask_array, mean, rank, iters, missing_data, mask, recon_pcs, min_height, max_height, client)`

Runs Singular Vector Decomposition on the inputted frames of shape (nframes, nfeatures). Data is centered by subtracting it by the mean value of the data. If `missing_data == True`, It will iteratively recompute the svd on the mean-centered data to reconstruct the PCs from the missing data until it converges.

**Parameters:**

- **dask\_array (dask 2d-array)** (Reshaped input data array of shape (nframes x nfeatures))
- **mean (1d array)** (Means of each row in dask\_array.)
- **rank (int)** (Rank of the desired thin SVD decomposition.)
- **iters (int)** (Number of SVD iterations)
- **missing\_data (bool)** (Indicates whether to compute SVD with a masked array)
- **mask (dask 2d-array)** (None if `missing_data == False`, else mask array of shape dask\_array)
- **recon\_pcs (int)** (Number of PCs to reconstruct for missing data.)
- **min\_height (int)** (Minimum height of mouse above the ground, used to filter reconstructed PCs.)
- **max\_height (int)** (Maximum height of mouse above the ground, used to filter reconstructed PCs.)
- **client (dask Client)** (Dask client to process batches.)

**Returns:** **s (1d array)** (computed singular values (eigen-values).) **v (2d array)** (computed principal components (eigen-vectors).) **mean (1d array)** (updated mean of dask array if missing\_data == True.) **total\_var (float)** (total variance captured by principal components.)

`moseq2_pca.pca.util.copy_metadata_to_scores(f, f_scores, uuid)`  
Copies metadata from individual session extract h5 files to the PCA scores h5 file.

**Parameters:**

- **f (read-open h5py File)** (open "results\_00.h5" h5py.File object in read-mode)
- **f\_scores (read-open h5py File)** (open "pca\_scores.h5" h5py.File object in read-mode)
- **uuid (str)** (uuid of inputted session h5 "f".)

**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, h5_path='/frames', h5_mask_path='/frames_mask', verbose=False)`  
Computes model-free changepoint block durations 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)
- **h5\_path (str)** (path to frames within selected h5 file (default: '/frames'))
- **h5\_mask\_path (str)** (path to masked frames within selected h5 file (default: '/frames\_mask'))
- **verbose (bool)** (print session names as they are being loaded.)

**Returns:**

**Return type:** None

`moseq2_pca.pca.util.get_timestamps(f, frames, fps=30)`  
Reads the timestamps from a given h5 file.

**Parameters:**

- **f (read-open h5py File)** (open "results\_00.h5" h5py.File object in read-mode)
- **frames (3d-array)** (list of 2d frames contained in opened h5 File.)
- **fps (int)** (frames per second.)

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

**Return type:** array of timestamps for inputted frames variable

`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, 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*)
- **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.

## Indices and tables

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



# Index

## Symbols

		--gaussfilter-space <gaussfilter_space>	moseq2-pca-train-pca	command line option
		--gaussfilter-time <gaussfilter_time>	moseq2-pca-train-pca	command line option
--chunk-size <chunk_size>	moseq2-pca-apply-pca	--h5-mask-path <h5_mask_path>	moseq2-pca-apply-pca	command line option
	moseq2-pca-compute-changepoints		moseq2-pca-compute-changepoints	command line option
	moseq2-pca-train-pca		moseq2-pca-train-pca	command line option
cluster-type <cluster_type>	moseq2-pca-apply-pca	--h5-path <h5_path>	moseq2-pca-apply-pca	command line option
	moseq2-pca-compute-changepoints		moseq2-pca-compute-changepoints	command line option
	moseq2-pca-train-pca		moseq2-pca-train-pca	command line option
--config-file <config_file>	moseq2-pca-apply-pca	--input-dir <input_dir>	moseq2-pca-apply-pca	command line option
	moseq2-pca-compute-changepoints		moseq2-pca-compute-changepoints	command line option
	moseq2-pca-train-pca		moseq2-pca-train-pca	command line option
--cores <cores>	moseq2-pca-apply-pca	--klags <klags>	moseq2-pca-compute-changepoints	command line option
	moseq2-pca-compute-changepoints	--local-processes <local_processes>	moseq2-pca-train-pca	command line option
	moseq2-pca-train-pca	--mask-height-threshold <mask_height_threshold>	moseq2-pca-train-pca	command line option
dask-cache-path <dask_cache_path>	moseq2-pca-apply-pca	--mask-threshold <mask_threshold>	moseq2-pca-train-pca	command line option
	moseq2-pca-compute-changepoints	--max-height <max_height>	moseq2-pca-train-pca	command line option
	moseq2-pca-train-pca	medfilter-space <medfilter_space>	moseq2-pca-train-pca	command line option
		--medfilter-time <medfilter_time>	moseq2-pca-train-pca	command line option
--dask-port <dask_port>	moseq2-pca-apply-pca	--memory <memory>	moseq2-pca-apply-pca	command line option
	moseq2-pca-compute-changepoints		moseq2-pca-compute-changepoints	command line option
	moseq2-pca-train-pca		moseq2-pca-train-pca	command line option
--detrend-window <detrend_window>	moseq2-pca-apply-pca	min-height <min_height>	moseq2-pca-train-pca	command line option
--dims <dims>	moseq2-pca-compute-changepoints	--missing-data	moseq2-pca-train-pca	command line option
--fill-gaps <fill_gaps>	moseq2-pca-apply-pca	missing-data-iters <missing_data_iters>	moseq2-pca-train-pca	command line option
--fps <fps>	moseq2-pca-apply-pca	--neighbors <neighbors>	moseq2-pca-compute-changepoints	command line option
	moseq2-pca-compute-changepoints			
--from-end	moseq2-pca-clip-scores			

--nworkers <nworkers>	moseq2-pca-apply-pca line option	command	tailfilter-size <tailfilter_size>	moseq2-pca-train-pca command line option
	moseq2-pca-compute-changepoints command line option		-t threshold <threshold>	moseq2-pca-compute-changepoints command line option
	moseq2-pca-train-pca line option	command	--timeout <timeout>	moseq2-pca-apply-pca command line option
--output-dir <output_dir>	moseq2-pca-apply-pca line option	command		moseq2-pca-compute-changepoints command line option
	moseq2-pca-compute-changepoints command line option			moseq2-pca-train-pca command line option
	moseq2-pca-train-pca line option	command	--use-fft	moseq2-pca-train-pca command line option
--output-file <output_file>	moseq2-pca-apply-pca line option	command	--verbose	moseq2-pca-apply-pca command line option
	moseq2-pca-compute-changepoints command line option			moseq2-pca-compute-changepoints command line option
	moseq2-pca-train-pca line option	command	--version	moseq2-pca command line option
overwrite-pca-apply <overwrite_pca_apply>	moseq2-pca-apply-pca command line option		--wall-time <wall_time>	moseq2-pca-apply-pca command line option
overwrite-pca-train <overwrite_pca_train>	moseq2-pca-train-pca command line option			moseq2-pca-compute-changepoints command line option
	--pca-file <pca_file>	moseq2-pca-apply-pca command line option		moseq2-pca-train-pca command line option
le-components <pca_file_components>	moseq2-pca-compute-changepoints command line option			moseq2-pca-apply-pca command line option
a-file-scores <pca_file_scores>	moseq2-pca-compute-changepoints command line option			moseq2-pca-compute-changepoints command line option
--pca-path <pca_path>	moseq2-pca-apply-pca line option	command		moseq2-pca-train-pca command line option
	moseq2-pca-compute-changepoints command line option	-d		moseq2-pca-apply-pca command line option
--processes <processes>	moseq2-pca-apply-pca line option	command		moseq2-pca-compute-changepoints command line option [1]
	moseq2-pca-compute-changepoints command line option			moseq2-pca-train-pca command line option
	moseq2-pca-train-pca line option	command-i		moseq2-pca-apply-pca command line option
--queue <queue>	moseq2-pca-apply-pca line option	command		moseq2-pca-compute-changepoints command line option
	moseq2-pca-compute-changepoints command line option			moseq2-pca-train-pca command line option
	moseq2-pca-train-pca line option	command	-k	moseq2-pca-compute-changepoints command line option
--rank <rank>	moseq2-pca-train-pca command line option		-m	moseq2-pca-apply-pca command line option
--recon-pcs <recon_pcs>	moseq2-pca-train-pca command line option			moseq2-pca-compute-changepoints command line option
--sigma <sigma>	moseq2-pca-compute-changepoints command line option			moseq2-pca-train-pca command line option
--tailfilter-shape <tailfilter_shape>	moseq2-pca-train-pca command line option			

-n	moseq2-pca-apply-pca	command line option
	moseq2-pca-compute-changepoints	command line option
	moseq2-pca-train-pca	command line option
-o	moseq2-pca-apply-pca	command line option
	moseq2-pca-compute-changepoints	command line option
	moseq2-pca-train-pca	command line option
-p	moseq2-pca-apply-pca	command line option
	moseq2-pca-compute-changepoints	command line option
	moseq2-pca-train-pca	command line option
-q	moseq2-pca-apply-pca	command line option
	moseq2-pca-compute-changepoints	command line option
	moseq2-pca-train-pca	command line option
-s	moseq2-pca-compute-changepoints	command line option
-v	moseq2-pca-apply-pca	command line option
	moseq2-pca-compute-changepoints	command line option
-w	moseq2-pca-apply-pca	command line option
	moseq2-pca-compute-changepoints	command line option
	moseq2-pca-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](#))  
[check\\_timestamps\(\)](#) (in module [moseq2\\_pca.util](#))  
[clean\\_frames\(\)](#) (in module [moseq2\\_pca.util](#))

## CLIP\_SAMPLES

[moseq2-pca-clip-scores](#) command line option

[clip\\_scores\\_wrapper\(\)](#) (in module [moseq2\\_pca.helpers.wrappers](#))  
[close\\_dask\(\)](#) (in module [moseq2\\_pca.util](#))  
[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](#))  
[compute\\_explained\\_variance\(\)](#) (in module [moseq2\\_pca.pca.util](#))  
[compute\\_svd\(\)](#) (in module [moseq2\\_pca.pca.util](#))  
[copy\\_metadata\\_to\\_scores\(\)](#) (in module [moseq2\\_pca.pca.util](#))

## 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\\_env\\_cpu\\_and\\_mem\(\)](#) (in module [moseq2\\_pca.util](#))  
[get\\_metadata\\_path\(\)](#) (in module [moseq2\\_pca.util](#))  
[get\\_pca\\_paths\(\)](#) (in module [moseq2\\_pca.helpers.data](#))  
[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](#))  
[get\\_timestamps\(\)](#) (in module [moseq2\\_pca.pca.util](#))

## H

[h5\\_to\\_dict\(\)](#) (in module [moseq2\\_pca.util](#))

## I

[initialize\\_dask\(\)](#) (in module [moseq2\\_pca.util](#))  
[insert\\_nans\(\)](#) (in module [moseq2\\_pca.util](#))

## L

[load\\_and\\_check\\_data\(\)](#) (in module [moseq2\\_pca.helpers.wrappers](#))  
[load\\_pcs\\_for\\_cp\(\)](#) (in module [moseq2\\_pca.helpers.data](#))

## M

[mask\\_data\(\)](#) (in module [moseq2\\_pca.pca.util](#))

### **moseq2-pca command line option**

--version

### **moseq2-pca-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>  
--dask-port <dask\_port>  
--detrend-window <detrend\_window>  
--fill-gaps <fill\_gaps>  
--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>  
--overwrite-pca-apply <overwrite\_pca\_apply>  
--pca-file <pca\_file>  
--pca-path <pca\_path>  
--processes <processes>  
--queue <queue>  
--timeout <timeout>  
--verbose  
--wall-time <wall\_time>  
-c  
-d  
-i  
-m  
-n  
-o  
-p  
-q  
-v  
-w

### **moseq2-pca-clip-scores command line option**

--from-end  
CLIP\_SAMPLES  
PCA\_FILE

### **moseq2-pca-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>  
--dask-port <dask\_port>  
--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>  
--verbose  
--wall-time <wall\_time>  
-c  
-d [1]  
-i  
-k  
-m  
-n  
-o  
-p  
-q  
-s  
-v  
-w

### **moseq2-pca-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>  
 --dask-port <dask\_port>  
 --gaussfilter-space <gaussfilter\_space>  
 --gaussfilter-time <gaussfilter\_time>  
 --h5-mask-path <h5\_mask\_path>  
 --h5-path <h5\_path>  
 --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>  
 --overwrite-pca-train <overwrite\_pca\_train>  
 --processes <processes>  
 --queue <queue>  
 --rank <rank>  
 --recon-pcs <recon\_pcs>  
 --tailfilter-shape <tailfilter\_shape>  
 --tailfilter-size <tailfilter\_size>  
 --timeout <timeout>  
 --use-fft  
 --wall-time <wall\_time>  
 -c  
 -d  
 -i  
 -m  
 -n  
 -o  
 -p  
 -q  
 -w

moseq2\_pca.gui (module)

moseq2\_pca.helpers.data (module)  
 moseq2\_pca.helpers.wrappers (module)  
 moseq2\_pca.pca.util (module)  
 moseq2\_pca.util (module)  
 moseq2\_pca.viz (module)

## P

### PCA\_FILE

moseq2\_pca-clip-scores command line option  
 plot\_pca\_results() (in module moseq2\_pca.viz)

## R

read\_yaml() (in module moseq2\_pca.util)  
 recursive\_find\_h5s() (in module moseq2\_pca.util)

## S

scree\_plot() (in module moseq2\_pca.viz)  
 select\_strel() (in module moseq2\_pca.util)  
 set\_dask\_config() (in module moseq2\_pca.util)

## T

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.util](#)

[moseq2\\_pca.viz](#)