CellTK

Release 0.4.3

Stevan Jeknic

Oct 31, 2022

CONTENTS:

CellTK

CellTK is collection of tools to simplify working with live-cell and fixed-cell microscopy data. It contains tools for segmenting, tracking, and analyzing images of both mammalian cells and bacterial microcolonies. Once the data are extracted, CellTK further includes, among others, tools for filtering data, building plots, and finding peaks. These tools can be used as stand-alone functions or as part of a larger analysis pipeline. More tools are on the way, and if there is anything you would like to see added, please create an issue on github.

CHAPTER

ONE

INSTALLATION

The easiest way to install CellTK is to use pip.

```
pip install celltk2
```

To install BayesianTracker

```
pip install celltk2[btrack]
```

Note: If you are using a Mac with Apple silicon processor, you may run into some issues with this installation. Your best bet may be to try to use conda to install cvxopt before doing the pip install above.

To install graph-based tracking method:

```
pip install celltk2[kit]
```

If you run into any problems, please open an issue.

CHAPTER

TWO

ACKNOWLEDGMENTS

This would not be possible without the original CellTK.

2.1 Quickstart Guide

2.1.1 Important classes

1. `Pipeline` - runs CellTK on a single folder of images.
2. `Orchestrator` - runs CellTK on multiple folders of images.
3. `Operation` - includes `Segment`, `Process`, `Track`, `Extract`, and `Evaluate`. Each of these holds the functions for analyzing images and can be found in `CellTK/celltk`.

2.1.2 Setting up a Pipeline

To use, save all of the following in a python script and run it. First, initialize a `Pipeline` and pass it the folder to your images. For this example, we will use the images in the example folder. Optionally, you can also pass a path to `output_folder`. By default, outputs will be stored in a new directory called `outputs`.

```
import celltk
pipe = celltk.Pipeline(parent_folder='/your/path/to/CellTK/examples/live_cell_example')
```

Next, build a set of `Operations` that you would like to use on those images. We first need to initialize the operation and let it know which images to use by passing a unique string that is in the name of those image files. In this case the nuclear channel can be identified with "channel000". We tell it that the output should be called "nuc". We also add a few extra options to save our final output and skip functions if it finds the files are already made.

```
seg = celltk.Segment(images=['channel000'], output='seg', save=True, force_rerun=False)
```

Next, we add the functions to the operation. For this example, we want to use UNet to find the nuclei followed by a simple constant threshold and cleaning to label the nuclei. Any kwargs those functions require can be passed to `add_function`. We will use the example weights for UNet, but if you have your own weights, you can pass them with the kwarg `weight_path`. For any function, you can add the kwarg `save_as` to save those output files. It's best to add this to time consuming operations so that they do not need to be repeated.

```
seg.add_function('unet_predict', save_as='unet_nuc')
seg.add_function('constant_thres', thres=0.8)
seg.add_function('clean_labels', min_radius=3, relabel=True)
```

Next, we will add a tracking operation using the same format as above. This time we import the output from `seg` to be used by the `Tracker`. We will use the `kit_sch_ge_tracker` function which runs a tracking algorithm from Katharina Loeffler and colleagues.

```
tra = celltk.Track(images=['channel000'], masks='seg', output='nuc',
                   save=True, force_rerun=False)
tra.add_function('kit_sch_ge_tracker')
```

Finally, we need to add an operation to extract the data and save it in an easy to use file. For this, we use `Extract`. This is the operation to pass most of the experimental metadata to. No functions are added to this operation.

```
ext = celltk.Extract(images=['channel000', 'channel0001'], tracks=['nuc'],
                     regions=['nuc'], channels=['tritc', 'fitc'],
                     time=10, min_trace_length=5, force_rerun=True)
```

Now we are ready to run everything! For this, we simply add the operations to the pipeline and hit go.

```
pipe.add_operations([seg, tra, ext])
pipe.run()
```

This will create the output folder, run all the operations, and save a file called `data_frame.hdf5` with all of the data saved as a `ConditionArray`.

2.1.3 Setting up an Orchestrator

Orchestrator es a very similar API as `Pipeline`. `parent_folder` should point to a directory that contains sub-directories of images.

2.1.4 Utilizing extracted data

After the pipeline runs, the data will be saved in an `hdf5` file. To access these data, load the file as a `ConditionArray`. For this example, we will use `examples/example_df.hdf5`.

```
array = celltk.ConditionArray.load('examples/example_df.hdf5')
print(array.shape)
> (1, 2, 24, 42, 6)
```

All `ConditionArrays` are five-dimensional. The dimensions are regions (e.g. nucleus, cytoplasm), channels (e.g. TRITC, FITC), metrics (e.g. median_intensity, area), cells, and frames. The first three dimensions can be indexed using strings, while the last two dimensions are indexed using integers. Currently, every indexing operation on a `ConditionArray` returns an `np.ndarray`. Addtionally, the array will always be at least two dimensional.

```
data = array['nuc', 'fitc', 'area']
print(data.shape)
> (42, 6)
print(type(data))
> numpy.ndarray
```

You can also index multiple items in each axis using a `list` or `tuple`. For example, you may want to get the `x` and `y` positions of each cell.

```
position = array['nuc', 'fitc', ('x', 'y')]
print(data.shape)
> (2, 42, 6)
```

For an `ExperimentArray` you can index in a couple different ways. One way is to index exactly as above, and pass the keys that you would like from each `ConditionArray`. This will return a list of `np.ndarray`, one for each `ConditionArray` in the `ExperimentArray`. You can also first index a list of conditions from the `ExperimentArray` and then index that list as above.

```
array = celltk.ExperimentArray.load('example/example_experiment.hdf5')
```

TODO

2.1.5 Plotting

CellTK includes a `PlotHelepr` class to helpp quickly generate plots from the data received by indexing an `ExperimentArray`. All functions return a `go.Figure` object, so customization of the plot and traces after it is produced is possible.

2.2 Pipelines

class celltk.core.pipeline.**Pipeline**(*parent_folder=None*, *output_folder=None*, *image_folder=None*, *mask_folder=None*, *array_folder=None*, *name=None*, *frame_rng=None*, *skip_frames=None*, *file_extension='tif'*, *overwrite=True*, *log_file=True*, *yaml_path=None*, *verbose=False*, *_split_key='&'*)

Pipeline organizes running Operations on saved images.

Parameters

- **parent_folder** (`Optional[str]`, default: `None`) – Location of the raw images
- **output_folder** (`Optional[str]`, default: `None`) – Location to store outputs. Defaults to 'parent_folder/outputs'
- **image_folder** (`Optional[str]`, default: `None`) – Location of images if different from parent_folder or output_folder
- **mask_folder** (`Optional[str]`, default: `None`) – Location of masks if different from parent_folder or output_folder
- **array_folder** (`Optional[str]`, default: `None`) – Location of arrays if different from parent_folder or output_folder
- **name** (`Optional[str]`, default: `None`) – Used to identify output array. Defaults to name parent_folder.
- **frame_rng** (`Optional[Tuple[int]]`, default: `None`) – Specify the frames to be loaded and used. If a single int, will load that many images. Otherwise designates the bounds of the range.
- **skip_frames** (`Optional[Tuple[int]]`, default: `None`) – Use to specify frames to be skipped. For example, frames that are out of focus.
- **file_extension** (`str`, default: `'tif'`) – File extension of image files to be loaded

- **ovewrite** – If True, outputs in output_folder are overwritten
- **log_file** (bool, default: True) – If True, save log outputs to a text file in output folder
- **yaml_path** (Optional[str], default: None) – Path to yaml file defining the Pipeline to be run
- **verbose** (bool, default: False) – If True, increases logging verbosity
- **_split_key** (str, default: '&') – Used to specify outputs from functions that return multiple outputs. For example, if you align two channels the outputs will be saved as 'align&channel000' and 'align&channel001'.

Returns
None

Parameters
overwrite (bool, default: True) –

add_operations(*operation*)

Adds Operations to the Pipeline.

Parameters
operation (Collection[*Operation*]) – A single operation or collection of operations to be run in the order passed

Return type
None

Returns
None

classmethod load_from_yaml(*path*)

Builds Pipeline class from specifications in YAML file

Parameters
path (str) – Path to yaml file

Returns
Pipeline with specified configuration

Return type
Pipeline class

run()

Load images and run the operations

Retrun
Generator returning outputs of the last operation

Return type
Generator

save_as_yaml(*path=None*, *fname=None*)

Saves Pipeline configuration as a YAML file

Parameters

- **path** (Optional[str], default: None) – Path to save yaml file in. Defaults to output_folder
- **fname** (Optional[str], default: None) – Name of the YAML file. Defaults to parent_folder

Return type
None

Returns
None

save_operations_as_yaml(*path=None*, *fname='operations.yaml'*)

Save configuration of Operations in Pipeline as a YAML file

Parameters

- **path** (Optional[str], default: None) – Path to save yaml file in. Defaults to output_folder
- **fname** (str, default: 'operations.yaml') – Name of the YAML file. Defaults to operations.yaml

Return type
None

Returns
None

class celltk.core.orchestrator.**Orchestrator**(*yaml_folder=None*, *parent_folder=None*, *output_folder=None*, *match_str=None*, *image_folder=None*, *mask_folder=None*, *array_folder=None*, *condition_map={}*, *position_map=None*, *cond_map_only=False*, *name='experiment'*, *frame_rng=None*, *skip_frames=None*, *file_extension='tif'*, *overwrite=True*, *log_file=True*, *save_master_df=True*, *job_controller=None*, *verbose=True*)

Orchestrator organizes running multiple Pipelines on a set of folders

Parameters

- **yaml_folder** (Optional[str], default: None) – Absolute path to location of Pipeline yamls
- **parent_folder** (Optional[str], default: None) – Absolute path of the folders with raw images
- **output_folder** (Optional[str], default: None) – Absolute path to folder to store outputs. Defaults to 'parent_folder/outputs'
- **match_str** (Optional[str], default: None) – If provided, folders that do not contain match_str are excluded from analysis
- **image_folder** (Optional[str], default: None) – Absolute path to folder with images if different from parent_folder or output_folder
- **mask_folder** (Optional[str], default: None) – Absolute path to folder with masks if different from parent_folder or output_folder
- **array_folder** (Optional[str], default: None) – Absolute path to folder with arrays if different from parent_folder or output_folder
- **condition_map** (dict, default: {}) – Dictionoary that maps folder names too the condition in the experiment. i.e {A1-Site_0: control}
- **position_map** (Union[dict, Callable, None], default: None) – If multiple positions have the same condition position map is used to uniquely identify them

- **cond_map_only** (bool, default: False) – If True, folders not in cond_map are not run. Only used if condition_map is provided.
- **name** (str, default: 'experiment') – Used to identify output array. Defaults to name parent_folder.
- **frame_rng** (Optional[Tuple[int]], default: None) – Specify the frames to be loaded and used. If a single int, will load that many images. Otherwise designates the bounds of the range.
- **skip_frames** (Optional[Tuple[int]], default: None) – Use to specify frames to be skipped. For example, frames that are out of focus.
- **file_extension** (str, default: 'tif') – File extension of image files to be loaded
- **ovewrite** – If True, outputs in output_folder are overwritten
- **log_file** (bool, default: True) – If True, save log outputs to a text file in output folder
- **save_master_df** (bool, default: True) – If True, saves a single hdf5 file containing the data from all of the pipelines
- **job_controller** (Optional[JobController], default: None) – If given, controls how pipelines are run
- **verbose** (bool, default: True) – If True, increases logging verbosity

Returns
None

Parameters
overwrite (bool, default: True) –

add_operations(*operation*, *index=-1*)

Adds Operations to the Orchestrator and all pipelines.

Parameters

- **operation** (Collection[*Operation*]) – A single operation or collection of operations to be run in the order passed
- **index** (int, default: -1) – If given, dictates where to insert the new operations

Return type
None

Returns
None

build_experiment_file(*match_str=None*)

Builds a single ExperimentArray from all of the ConditionArrays found in the Pipeline folders.

Parameters
match_str (Optional[str], default: None) – If given, only files containing match_str are included

Return type
None

load_operations_from_yaml(*path*)

Loads Operations from a YAML file

Parameters
path (str) – path to YAML file

Return type
None

Returns
None

run(*n_cores=1*)

Load images and run all of the pipelines

Parameters
n_cores (int, default: 1) – Not currently implemented

Retrun
None

Return type
None

save_condition_map_as_yaml(*path=None*, *fname='conditions.yaml'*)

Saves the conditions in Orchestrator as a YAML file

Parameters

- **path** (Optional[str], default: None) – Path to save the file at
- **fname** (str, default: 'conditions.yaml') –

Return type
None

Returns
None

save_operations_as_yaml(*path=None*, *fname='operations.yaml'*)

Save configuration of Operations in Pipeline as a YAML file

Parameters

- **path** (Optional[str], default: None) – Path to save yaml file in. Defaults to output_folder
- **fname** (str, default: 'operations.yaml') – Name of the YAML file. Defaults to operations.yaml

Return type
None

Returns
None

save_pipelines_as_yamls(*path=None*)

Saves Orchestrator configuration as a YAML file

Parameters
path (Optional[str], default: None) – Path to save yaml file in. Defaults to output_folder

Return type
None

Returns
None

`update_condition_map`(*condition_map={}*, *path=None*)

Adds conditions to each of the Pipelines in Orchestrator

Parameters

- **condition_map** (`dict`, default: `{}`) – Dictionoary that maps folder names too the condition in the experiment. i.e {A1-Site_0: control}
- **path** (`Optional[str]`, default: `None`) – If given, loads a condition map from the file found at path. Overwrites condition_map.

Return type
`None`

Returns
None

2.3 Operations

2.3.1 Process

`class celltk.process.Process`(*output='process'*, ***kwargs*)

Parameters
output (`str`, default: `'process'`) –

`align_by_cross_correlation`(*image=()*, *mask=()*, *align_with='image'*, *crop=True*, *normalization='phase'*)

Uses phase cross-correlation to shift the images to align them. Optionally can crop the images to align. Align with can be used to specify which of the inputs to use. Uses the first stack in the given list.

Parameters

- **image** (`Image`, default: `()`) – List of image stacks to be aligned.
- **mask** (`Mask`, default: `()`) – List of mask stacks to be aligned.
- **align_with** (`str`, default: `'image'`) – Can be one of 'image', 'mask', or 'track'. Defines which of the input stacks should be used for alignment.
- **crop** (`bool`, default: `True`) – If True, the aligned stacks are cropped based on the largest frame to frame shifts.
- **normalization** (`str`, default: `'phase'`) –

Return type
`Stack`

Returns
Aligned input stack.

Raises
AssertionError – If input stacks have different shapes.

apply_log_bias_field(*image*, *bias_field*)

Applies a log bias field (for example, calculated using N4 bias illumination correction) to the input image.

Parameters

- **image** (Image) –
- **bias_field** (Image) –

Return type
Image

binomial_blur(*image*, *iterations=7*)

Applies a binomial blur to the image.

Parameters

- **image** (Image) –
- **iterations** (int, default: 7) –

Return type
Image

crop_to_area(*images=()*, *masks=()*, *crop_factor=0.6*)

Crops provided images to a specifc area set by crop_factor.

Parameters

- **images** (Image[Optional], default: ()) –
- **masks** (Mask[Optional], default: ()) –
- **crop_factor** (float, default: 0.6) –

Return type
Stack

Returns

curvature_anisotropic_diffusion(*image*, *iterations=5*, *time_step=0.125*, *conductance=1.0*)

Applies curvature anisotropic diffusion blurring to the image. Useful for smoothing out noise, while preserving the edges of objects.

Parameters

- **image** (Image) –
- **iterations** (int, default: 5) –
- **time_step** (float, default: 0.125) –
- **conductance** (float, default: 1.0) –

Return type
Image

gaussian_filter(*image*, *sigma=2.5*, *dtype=<class 'numpy.float32'>*)

Applies a multidimensional Gaussian filter to the image.

Parameters

- **image** (Image) –
- **sigma** (float, default: 2.5) –
- **dtype** (type, default: <class 'numpy.float32'>) –

Return type
Image

Returns

gaussian_laplace_filter(*image*, *sigma=2.5*)

Multidimensional Laplace filter using Gaussian second derivatives.

Parameters

- **image** (Image) –
- **sigma** (float, default: 2.5) –

Return type
Image

histogram_matching(*image*, *bins=1000*, *match_pts=100*, *threshold=False*, *ref_frame=0*)

Rescales input image frames to match the intensity of a reference image. By default, the reference image is the first frame of the input image stack.

Parameters

- **image** (Image) –
- **bins** (int, default: 1000) –
- **match_pts** (int, default: 100) –
- **threshold** (bool, default: False) –
- **ref_frame** (int, default: 0) –

Return type
Image

inverse_gaussian_gradient(*image*, *alpha=100.0*, *sigma=5.0*)

Calculates gradients and inverts them on the range [0, 1], such that pixels close to borders have values close to 0, while all other pixels have values close to 1.

Parameters

- **image** (Image) –
- **alpha** (float, default: 100.0) –
- **sigma** (float, default: 5.0) –

Return type
Image

make_edge_potential_image(*image*, *method='sigmoid'*, *alpha=None*, *beta=None*, *k1=None*, *k2=None*)

Calculates an edge potential image from images with edges highlighted. An edge potential image has values close to 0 at edges, and values close to 1 else where. The quality of the edge potential image depends highly on the input image and the function/parameters used. The default function is 'sigmoid', which accepts two parameters to define the sigmoid function, alpha and beta. If you don't already know good values, heuristics can be used to estimate alpha and beta based on the minimum value along an edge (k1) and the average value away from an edge (k2). If no parameters are supplied, this function will attempt to guess.

Parameters

- **image** (Image) –
- **method** (str, default: 'sigmoid') –

- **alpha** (Optional[float], default: None) –
- **beta** (Optional[float], default: None) –
- **k1** (Optional[float], default: None) –
- **k2** (Optional[float], default: None) –

Return type
Image

make_maurer_distance_map(*image*, *value_range=None*, *inside_positive=False*, *use_euclidian=False*, *use_image_spacing=False*)

Applies a filter to calculate the distance map of a binary image with objects. The distance inside objects is negative.

Parameters

- **image** (Image) –
- **value_range** (Optional[Collection[float]], default: None) –
- **inside_positive** (bool, default: False) –
- **use_euclidean** –
- **use_image_spacing** (bool, default: False) –
- **use_euclidian** (bool, default: False) –

Return type
Image

n4_illumination_bias_correction(*image*, *mask=None*, *iterations=50*, *num_points=4*, *histogram_bins=200*, *spline_order=3*, *subsample_factor=1*, *save_bias_field=False*)

Applies N4 bias field correction to the image. Can optionally return the calculated log bias field, which can be applied to the image with Process.apply_log_bias_field.

Parameters

- **image** (Image) –
- **mask** (Optional[Mask], default: None) –
- **iterations** (Collection[int], default: 50) –
- **num_points** (Collection[int], default: 4) –
- **histogram_bins** (int, default: 200) –
- **spline_order** (int, default: 3) –
- **subsample_factor** (int, default: 1) – Amount to shrink image before calculating log_bias_field. Speeds up calculation.
- **save_bias_field** (bool, default: False) – If True, returns calculated log bias field instead of corrected image.

Return type
Image

recurssive_gauss_gradient(*image*, *sigma=1.0*, *use_direction=True*)

Applies recursive Gaussian filters to detect edges.

Parameters

- **image** (Image) –
- **sigma** (float, default: 1.0) –
- **use_direction** (bool, default: True) –

Return type
Image

recurssive_gauss_magnitude(*image*, *sigma=1.0*)

Applies recursive Gaussian filters to detect edges and returns the gradient magnitude at each pixel.

Parameters

- **image** (Image) –
- **sigma** (float, default: 1.0) –

Return type
Image

roberts_edge_detection(*image*)

Applies Roberts filter for edge detection.

Parameters
image (Image) –

Return type
Image

rolling_ball_background_subtract(*image*, *radius=100*, *kernel=None*, *nansafe=False*, *return_bg=False*)

Estimate background intensity by rolling/translating a kernel, and subtract from the input image.

Parameters

- **image** (Image) –
- **radius** (float, default: 100) –
- **kernel** (Optional[ndarray], default: None) –
- **nansafe** (bool, default: False) –
- **return_bg** (bool, default: False) – If True, returns background instead of image with background subtracted.

Return type
Image

sobel_edge_detection(*image*, *orientation='both'*)

Applies Sobel filter for edge detection. Can detect edges in only one dimension by using the orientation argument.

Parameters

- **image** (Image) –
- **orientation** (str, default: 'both') –

Return type
Image

sobel_edge_magnitude(*image*)

Similar to `Process.sobel_edge_detection`, but returns the magnitude of the gradient at each pixel, without regard for direction.

Parameters
image (`Image`) –

Return type
`Image`

tile_images(*image=()*, *mask=()*, *layout=None*, *border_value=0.0*)

Tiles image stacks side by side to produced a single image. Attempts to do some rescaling to match intensities first, but likely will not produce good results for images with large differences in intensity.

Parameters

- **image** (`Image[Optional]`, default: `()`) – List of image stacks to be tiled.
- **mask** (`Mask[Optional]`, default: `()`) – List of mask stacks to be tiled.
- **layout** (`Optional[Tuple[int]]`, default: `None`) –
- **border_value** (`Union[int, float]`, default: `0.0`) – Value of the default pixels.

Return type
`Image`

unet_predict(*image*, *weight_path*, *roi=2*, *batch=None*, *classes=3*)

Uses a UNet-based neural net to predict the label of each pixel in the input image. This function returns the probability of a specific region of interest, not a labeled mask.

Parameters

- **image** (`Image`) –
- **weight_path** (`str`) –
- **roi** (`Union[int, str]`, default: `2`) –
- **batch** (`Optional[int]`, default: `None`) –
- **classes** (`int`, default: `3`) –

Return type
`Image`

Returns

uniform_filter(*image*, *size=3*, *mode='reflect'*, *cval=0*)

Applies a multidimensional uniform filter to the input image.

Parameters

- **image** (`Image`) –
- **size** (`int`, default: `3`) –
- **mode** (`str`, default: `'reflect'`) –
- **cval** (`int`, default: `0`) –

Return type
`Image`

wavelet_background_subtract(*image*, *wavelet='db4'*, *mode='symmetric'*, *level=None*, *blur=False*, *return_bg=False*)

Uses discrete wavelet transformation to estimate and remove the background from an image.

Parameters

- **image** (Image) –
- **wavelet** (str, default: 'db4') –
- **mode** (str, default: 'symmetric') –
- **level** (Optional[int], default: None) –
- **blur** (bool, default: False) –
- **return_bg** (bool, default: False) – If True, returns estimated background, instead of image with background subtracted.

Return type
Image

wavelet_noise_subtract(*image*, *noise_level=1*, *thres=2*, *wavelet='db1'*, *mode='smooth'*, *level=None*)

Uses discrete wavelet transformation to estimate and remove noise from an image.

Parameters

- **image** (Image) –
- **noise_level** (int, default: 1) –
- **thres** (int, default: 2) –
- **wavelet** (str, default: 'db1') –
- **mode** (str, default: 'smooth') –
- **level** (Optional[int], default: None) –

Return type
Image

2.3.2 Segment

class celltk.segment.**Segment**(*output='mask'*, ***kwargs*)

Parameters
output (str, default: 'mask') –

adaptive_thres(*image*, *relative_thres=0.1*, *sigma=50*, *connectivity=2*)

Applies Gaussian blur to the image and marks pixels that are brighter than the blurred image by a specified threshold.

Parameters

- **image** (Image) –
- **relative_thres** (float, default: 0.1) –
- **sigma** (float, default: 50) –
- **connectivity** (int, default: 2) –

Return type
Mask[uint8]

Returns

agglomeration_segmentation(*image*, *seeds=None*, *agglom_min=0.7*, *agglom_max=None*, *compact=100*, *seed_thres=0.975*, *seed_min_size=12*, *steps=50*, *connectivity=2*)

Starts from a seed mask determined by a constant threshold. Then incrementally uses watershed to connect neighboring pixels to each seed.

Parameters

- **image** (Image) –
- **seeds** (Optional[Mask], default: None) –
- **agglom_min** (float, default: 0.7) –
- **agglom_max** (Optional[float], default: None) –
- **compact** (float, default: 100) –
- **seed_thres** (float, default: 0.975) –
- **seed_min_size** (float, default: 12) –
- **steps** (int, default: 50) –
- **connectivity** (int, default: 2) –

Return type
Mask

Returns

binary_fill_holes(*mask*, *fill_border=True*, *iterations=False*, *kernel_radius=4*, *max_length=45*, *in_place=True*, *method='ndi'*, ***kwargs*)

Fills holes in a binary image. Two algorithms are available, one from SimpleITK and one from scipy. Neither algorithm fills holes on the border of the image (due to ambiguity in defining a hole). This function can optionally fill holes on the border by guessing which holes are holes in a larger object.

Parameters

- **mask** (Mask) –
- **fill_border** (bool, default: True) – If True, attempts to fill holes on boundary of image. Note, there is ambiguity in defining a hole on the boundary.
- **iterations** (Union[bool, int], default: False) –
- **max_length** (int, default: 45) –
- **in_place** (bool, default: True) –
- **method** (str, default: 'ndi') –
- **kwargs** –

Parma kernel_radius
Return type
Mask

Returns

Parameters
kernel_radius (int, default: 4) –

canny_edge_segmentation(*image*, *sigma=1.0*, *low_thres=None*, *high_thres=None*, *use_quantiles=False*, *fill_holes=False*)

Uses a Canny filter to find edges in the image.

Parameters

- **image** (Image) –
- **sigma** (float, default: 1.0) –
- **low_thresh** –
- **high_thres** (Optional[float], default: None) –
- **use_quantiles** (bool, default: False) –
- **fill_holes** (bool, default: False) –
- **low_thres** (Optional[float], default: None) –

Return type
Mask

Returns

chan_vese_dense_levelset(*image*, *seeds*, *iterations=70*, *smoothing=0*, *curve_weight=1*, *area_weight=1*, *lambda1=1*, *lambda2=1*, *epsilon=1*)

Calculates the Chan-Vese level set from initial seeds. Similar to Segment.morphological_acwe, but more customizable.

Parameters

- **image** (Image) –
- **mask** –
- **iterations** (int, default: 70) –
- **smoothing** (float, default: 0) –
- **curve_weight** (float, default: 1) –
- **area_weight** (float, default: 1) –
- **lambda1** (float, default: 1) –
- **lambda2** (float, default: 1) –
- **epsilon** (float, default: 1) –
- **seeds** (Mask) –

Return type
Mask

Returns

clean_labels(*mask*, *min_radius=3*, *max_radius=20*, *open_size=3*, *clear_border=True*, *relabel=False*, *sequential=False*, *connectivity=2*)

Applies light cleaning intended for relatively small mammalian nuclei. Fills holes, removes small, large, and border-connected objectes, and then applies morphological opening.

Parameters

- **mask** (Mask) – Mask of uniquely labeled cell nuclei

- **min_radius** (float, default: 3) – Objects smaller than a circle with radius min_radius are removed.
- **max_radius** (float, default: 20) – Objects larger than a circle with radius max_radius are removed.
- **open_size** (int, default: 3) – Side length of the footprint used for the morphological opening operation.
- **clear_border** (Union[bool, int], default: True) – If True or int, adny object connected to the border is removed. If an integer is given, objects that many pixels from the border are also removed.
- **relabel** (bool, default: False) – If True, objects are relabeled after cleaning. This can help separate objects that were connected before cleaning, but are no longer connected.
- **sequential** (bool, default: False) – If True, objects are relabeled with sequential labels.
- **connectivity** (int, default: 2) – Determines the local neighborhood around a pixel for defining connected objects. Connectivity is defined as the number of orthogonal steps needed to reach a pixel.

Return type
Mask

Returns
Mask with cleaned labels

confidence_connected(*image*, *seeds*, *radius=1*, *multiplier=4.5*, *iterations=10*)

Uses mean and variance to connect pixels that are within a specified window (multiplier) of the values in the seed region.

Parameters

- **image** (Image) –
- **seeds** (Mask) –
- **radius** (int, default: 1) –
- **multiplier** (float, default: 4.5) –
- **iterations** (int, default: 10) –

Return type
Mask

Returns

constant_thres(*image*, *thres=1000*, *negative=False*, *connectivity=2*, *relative=False*)

Labels pixels above or below a threshold value.

Parameters

- **image** (Image) –
- **thres** (Union[int, float], default: 1000) – Threshold value to use
- **negative** (bool, default: False) – If True, labels pixels below thres
- **connectivity** (int, default: 2) –
- **relative** (bool, default: False) – If True, threshold is set relative to maximum pixel value in image.

Return type
Mask[uint8]

Returns
Labeled binary mask

convex_hull_object(*mask*, *connectivity=2*)

Computes the convex hull of each object found in a binary mask. If mask is not binary already, will be binarized.

Parameters

- **mask** (Mask) –
- **connectivity** (int, default: 2) –

Return type
Mask

Returns

expand_to_cytoring(*labels*, *image=None*, *distance=1*, *margin=0*, *thres=None*, *relative=True*, *mask=None*)

Expands labels in the given mask by a fixed distance.

Parameters

- **labels** (Mask) –
- **distance** (float, default: 1) –
- **margin** (int, default: 0) –
- **thres** (Optional[float], default: None) –
- **relative** (bool, default: True) –
- **mask** (Optional[Mask], default: None) – If given, only expands labels into indices that are True in mask.
- **image** (Optional[Image], default: None) –

Return type
Mask

Returns

fast_marching_level_set(*image*, *seeds*, *n_points=0*)

Computes the distance from the seeds based on a fast marching algorithm.

Parameters

- **image** (Image) –
- **seeds** (Mask) –
- **n_points** (int, default: 0) –

Return type
Mask

Returns

filter_objects_by_props(*mask*, *properties*, *limits*, *image=None*)

Removes objects from a labeled image based on shape properties. Accepted properties are ones available for skimage.measure.regionprops.

Parameters

- **mask** (Mask) –
- **image** (Optional[Image], default: None) –
- **properties** (List[str]) –
- **limits** (Collection[Tuple[float]]) –

Return type
Mask

Returns

find_boundaries(*mask*, *connectivity=2*, *mode='thick'*, *keep_labels=True*)

Returns the outlines of the objects in the mask, optionally preserving the labels.

Parameters

- **mask** (Mask) –
- **connectivity** (int, default: 2) –
- **mode** (str, default: 'thick') –
- **keep_labels** (bool, default: True) –

Return type
Mask

Returns

label(*mask*, *connectivity=2*)

Uniquely labels connected pixels.

Parameters

- **mask** (Mask) – Mask of objects to be labeled. Can be binary or greyscale.
- **connectivity** (int, default: 2) – Determines the local neighborhood around a pixel. Defined as the number of orthogonal steps needed to reach a pixel.

Return type
Mask

Returns
Labeled mask.

label_by_voting(*masks*, *label_undecided=0*)

Applies a voting algorithm to determine the value of each pixel. Can be used to get the combined result from multiple masks.

Parameters

- **masks** (Mask) – List of masks to use in voting.
- **label_undecided** (int, default: 0) –

Return type
Mask

Returns

level_set_mask(*image*, *levelset='checkerboard'*, *size=None*, *center=None*, *label=False*)

Returns a binary level set of various shapes.

Parameters

- **image** (Image) –
- **levelset** (str, default: 'checkerboard') –
- **size** (Union[float, int, None], default: None) –
- **center** (Optional[Tuple[int]], default: None) –
- **label** (bool, default: False) –

Return type
Mask

Returns

li_thres(*image*, *inside_val=0*, *outside_val=1*)

Applies Li's thresholding method.

Parameters

- **image** (Image) –
- **inside_val** (int, default: 0) –
- **outside_val** (int, default: 1) –

Return type
Mask[uint8]

Returns

misic_predict(*image*, *model_path=None*, *weight_path=None*, *batch=None*)

Wrapper for implementation of Misic. Original paper from Panigrahi and colleagues.

Parameters

- **image** (Image) –
- **model_path** (Optional[str], default: None) –
- **weight_path** (Optional[str], default: None) –
- **batch** (Optional[int], default: None) –

Return type
Mask

Returns

morphological_acwe(*image*, *seeds='checkerboard'*, *iterations=10*, *smoothing=1*, *lambda1=1*, *lambda2=1*, *connectivity=1*, *thres=None*, *voronoi=False*, *keep_labels=True*, *clean_before_match=True*)

Uses morphological active contours without edges (ACWE) algorithm to segment objects. Optionally, applies a vornoi mask after segmentation to separate objects.

Parameters

- **image** (Image) –
- **seeds** (Mask[Optional], default: 'checkerboard') –
- **iterations** (int, default: 10) –

- **smoothing** (int, default: 1) –
- **lambda1** (float, default: 1) –
- **lambda2** (float, default: 1) –
- **connectivity** (int, default: 1) –
- **thres** (Optional[float], default: None) –
- **voronoi** (bool, default: False) –
- **keep_labels** (bool, default: True) –
- **clean_before_match** (bool, default: True) –

Return type
Mask

Returns

morphological_geodesic_active_contour(*image*, *seeds='checkerboard'*, *iterations=10*, *smoothing=1*, *threshold='auto'*, *balloon=0*)

Applies geodesic active contours with morphological operators to segment objects. Light wrapper for skimage.segmentation.morphological_geodesic_active_contour.

Parameters

- **image** (Image) –
- **seeds** (Mask[Optional], default: 'checkerboard') –
- **iterations** (int, default: 10) –
- **smoothing** (int, default: 1) –
- **threshold** (float, default: 'auto') –
- **balloon** (float, default: 0) –

Return type
Mask

Returns

morphological_watershed(*image*, *watershed_line=True*)

Runs morphological watershed on the image.

Parameters

- **image** (Image) –
- **watershed_line** (bool, default: True) –

Return type
Mask

Returns

multiotsu_thres(*image*, *classes=2*, *roi=None*, *nbins=256*, *hist=None*, *binarize=False*)

Applies Otsu's thresholding with multiple classes. By default, returns a mask with all classes included, but can be limited to only returning some of the classes.

Parameters

- **image** (Image) –
- **classes** (int, default: 2) – Which classes to be included in final image.

- **roi** (Union[int, Collection[int], None], default: None) –
- **nbins** (int, default: 256) –
- **hist** (Optional[ndarray], default: None) –
- **binarize** (bool, default: False) – If True, a binary image is returned.

Return type
Mask[uint8]

Returns

otsu_thres(*image*, *nbins=256*, *connectivity=2*, *buffer=0.0*, *fill_holes=False*)

Uses Otsu's method to determine the threshold and labels all pixels above the threshold.

Parameters

- **image** (Image) –
- **nbins** (int, default: 256) –
- **connectivity** (int, default: 2) –
- **buffer** (float, default: 0.0) – Adjust threshold value by a specific buffer.
- **fill_holes** (bool, default: False) – If True, applies binary fill hole operation after thresholding

Return type
Mask[uint8]

Returns

random_walk_segmentation(*image*, *seeds=None*, *seed_thres=0.99*, *seed_min_size=12*, *beta=80*, *tol=0.01*, *seg_thres=0.85*)

Uses random aniostropic diffusion from given seeds to assign each pixel in the image to a specific seed.

Parameters

- **image** (Image) –
- **seeds** (Optional[Mask], default: None) –
- **seed_thres** (float, default: 0.99) –
- **seed_min_size** (float, default: 12) –
- **beta** (float, default: 80) –
- **tol** (float, default: 0.01) –
- **seg_thres** (float, default: 0.85) –

Return type
Mask

Returns

regional_extrema(*image*, *find='minima'*, *fully_connected=True*, *thres=None*, *min_size=12*)

Finds regional minima/maxima within flat areas. Generally requires an image with little noise.

Parameters

- **image** (Image) –
- **find** (str, default: 'minima') –

- **fully_connected** (bool, default: True) –
- **thres** (Optional[float], default: None) –
- **min_size** (int, default: 12) –

Return type
Mask

Returns
Mask with regional extrema labeled

remove_nuc_from_cyto(*cyto_mask*, *nuc_mask*, *val_match=False*, *erosion=False*)

Removes nuclei from a cytoplasmic mask, optionally eroding the final mask.

Parameters

- **cyto_mask** (Mask) –
- **nuc_mask** (Mask) –
- **val_match** (bool, default: False) –
- **erosion** (bool, default: False) –

Return type
Mask

Returns

shape_detection_level_set(*edge_potential*, *initial_level_set*, *curvature=1*, *propagation=1*, *iterations=1000*)

Propagates an initial level set to edges found in an edge potential image. This function will likely not work well on an unmodified input image. Use the edge detection functions in Process to create an edge potential image.

Parameters

- **edge_potential** (Image) –
- **initial_level_set** (Mask) –
- **curvature** (float, default: 1) –
- **propagation** (float, default: 1) –
- **iterations** (int, default: 1000) –

Return type
Mask

Returns

sitk_filter_pipeline(*mask*, *filters*, *kwargs=[]*)

Applies arbitrary filters from the SimpleITK package. Any image filter in SimpleITK, and most arguments for those filters, can be used.

Parameters

- **mask** (Mask) –
- **filters** (Collection[str]) – List of filters from SimpleITK to apply.
- **kwargs** (Collection[dict], default: []) – List of dictionaries containing the kwargs for the selected filters. Length must equal length of filters.

Return type
Mask

Returns

sitk_label(*mask*, *min_size=None*)

Uniquely labels connected pixels. Also removes connected objects below a specified size.

Parameters

- **mask** (Mask) – Mask of objects to be labeled. Can be binary or greyscale.
- **min_size** (Optional[float], default: None) – If given, all objects smaller than min_size are removed.

Return type
Mask

Returns
Labeled mask

threshold_level_set(*image*, *initial_level_set*, *lower_thres=None*, *upper_thres=None*, *propagation=1*, *curvature=1*, *iterations=1000*, *max_rms_error=0.02*)

Level set segmentation method based on intensity values.

Parameters

- **image** (Image) –
- **initial_level_set** (Mask) –
- **lower_thres** (Optional[float], default: None) –
- **upper_thres** (Optional[float], default: None) –
- **propagation** (float, default: 1) –
- **curvature** (float, default: 1) –
- **iterations** (int, default: 1000) –
- **max_rms_error** (float, default: 0.02) –

Return type
Mask

Returns

unet_predict(*image*, *weight_path*, *roi=2*, *batch=None*, *classes=3*)

Uses a UNet-based neural net to predict the label of each pixel in the input image. This function returns the probability of a specific region of interest, not a labeled mask.

Parameters

- **image** (Image) –
- **weight_path** (str) –
- **roi** (Union[int, str], default: 2) –
- **batch** (Optional[int], default: None) –
- **classes** (int, default: 3) –

Return type
Image

Returns

watershed_from_markers(*image*, *mask*, *watershed_line=True*, *remove_large=True*)

Runs morphological watershed from given seeds.

Parameters

- **image** (Image) –
- **mask** (Mask) –
- **watershed_line** (bool, default: True) –
- **remove_large** (bool, default: True) –

Return type
Mask

Returns

watershed_ift_segmentation(*image*, *seeds=None*, *seed_thres=0.975*, *seed_min_size=12*, *connectivity=2*)

Applies watershed from the given seeds using the image foresting transform algorithm.

Parameters

- **image** (Image) –
- **seeds** (Optional[Mask], default: None) –
- **seed_thres** (float, default: 0.975) –
- **seed_min_size** (float, default: 12) –
- **connectivity** (int, default: 2) –

Return type
Mask

Returns

2.3.3 Track

class celltk.track.**Track**(*output='track'*, ***kwargs*)

Parameters
output (str, default: 'track') –

bayesian_tracker(*mask*, *config_path='celltk/config/bayes_config.json'*, *update_method='exact'*)

Wrapper for btrack, a Bayesian-based tracking algorithm.

Note:

- The underlying tracking algorithm was developed by `Ulicna

and colleagues`_. Please see: https://github.com/quantumjot/BayesianTracker

---

Parameters

- **mask** (Mask) – Mask with objects to be segmented

- **config_path** (str, default: 'celltk/config/bayes_config.json') – Path to configuration file. Must be JSON.
- **update_method** (str, default: 'exact') – Method to use when optimizing the solution. Options are 'exact' and 'approximate'. Use approximate if exact is taking too long or utilizing excessive resources.

Return type
Mask[int16]

Returns
Track with objects linked

detect_cell_division(*image*, *track*, *mask=None*, *displacement_thres=15*, *frame_thres=3*, *mass_thres=0.25*, *dist_thres=0.35*, *dot_thres=-0.7*, *angle_weight=0.5*)

Detects cells that have divided based on location, size, and intensity information. Daughter cells are expected to be approximately half the total intensity of the mother cell, and to have the mother cell roughly in line and between them.

Note:

- Any Mask passed to this function will have all other division events removed.

Parameters

- **image** – Image with intensity information.
- **track** – Mask of objects to detect division on.
- **mask** – Mask of objects to detect division on. Note that the objects must already be linked from frame to frame for the algorithm to work.
- **displacement_thres** – Maximum distance allowed from the centroid of a parent cell to the centroid of a daughter cell.
- **frame_thres** – Maximum number of frames from the division event to a daughter cell.
- **mass_thres** – Maximum error for total intensity of a daughter cell relataive to half of the mother cell's intensity.
- **dist_thres** – Maximum error allowed for location of mother cell relative to location of the daughter cells. Mother cell is expected to be equidistant from the daughter cells.
- **dot_thres** – Maximum value of the normalized dot product between the two vectors from each daughter cell to the mother cell. Essentially a measure of the angle between the daughter cells.
- **angle_weight** – Weight given to angle and distance information when assigning daughter cells. If multiple candidate daughters are found, they are assigned based on intensity information and angle information.

Returns
Mask with objects linked and cell division marked

kit_sch_ge_tracker(*image*, *mask*, *default_roi_size=2*, *delta_t=3*, *cut_off_distance=None*, *allow_cell_division=True*, *postprocessing_key=None*)

Tree-based tracking algorithm. First creates small tracklets within delta_t frames, then finds a globally optimal solution for linking the tracklets to form full tracks. Has built-in cell detection and can also combine objects.

Note:

- Objects can change in this algorithm, so the output is not guaranteed to have the exact same objects as the input mask.

Note:

- A current Gurobi license is required to use this algorithm. Personal licenses are free for academics.

Note:

- The underlying tracking algorithm was developed by Katharina Loffler and colleagues.

Parameters

- **image** (Image) – Image with intensity information
- **mask** (Mask) – Mask with objects to be tracked
- **default_roi_size** (int, default: 2) – Size of the region to look for connecting objects. Set relative to the mean size of the objects. i.e. 2 means a search area twice as large as the mean object.
- **delta_t** (int, default: 3) – Number of frames in each window for forming tracklets.
- **cut_off_distance** (Optional[Tuple], default: None) – Maximum distance between linked objects
- **allow_cell_division** (bool, default: True) – If True, attempt to locate and mark dividing cells.
- **postprocessing_key** (Optional[str], default: None) – TODO. See KIT-Sch-GE documentation

Return type
Mask[int16]

lineage_masks(*track*)

Creates a mask where all daughter cells have the label of their parent cell.

Parameters
track (Mask) – Track with parent and daughter cells.

Return type
Mask[int16]

Returns

linear_tracker_w_properties(*image*, *mask*, *properties=[]*, *weights=None*, *thresholds=None*, *displacement_thres=30*, *mass_thres=0.15*)

Tracks objects from frame to frame by optimizing the cost defined by an arbitrary list of properties. By default, includes thresholds on the distance and difference in intensity for two linked objects.

Parameters

- **image** (Image) –

- **mask** (Mask) –
- **properties** (Collection[str], default: []) –
- **weights** (Optional[Collection[float]], default: None) –
- **thresholds** (Optional[Collection[float]], default: None) –
- **displacement_thres** (float, default: 30) –
- **mass_thres** (float, default: 0.15) –

Return type
Mask[int16]

simple_linear_tracker(*mask*, *voronoi_split=True*)

Tracks objects by optimizing the area overlap from frame to frame.

Parameters

- **mask** (Mask) – Mask with the objects to be tracked uniquely labeled.
- **voronoi_split** (bool, default: True) – If True, creates a voronoi map from the objects in mask. Uses that map to keep objects separated in each frame. Slows down computation.

Return type
Mask[int16]

Returns
Mask with objects linked

simple_watershed_tracker(*mask*, *connectivity=2*, *watershed_line=True*, *keep_seeds=False*)

Uses watershed to track from objects in one frame to objects in the next. Useful for objects that grow in size, but don't move much, such as bacterial colonies.

Parameters

- **mask** (Mask) – Mask with the objects to be tracked uniquely labeled.
- **connectivity** (int, default: 2) – Determines the local neighborhood around a pixel. Defined as the number of orthogonal steps needed to reach a pixel.
- **watershed_line** (bool, default: True) – If True, a one-pixel wide line separates the regions obtained by the watershed algorithm. The line is labeled 0.
- **keep_seeds** (bool, default: False) – If True, seeds from previous frame are always kept. This is more robust for segmentation that is incomplete or fragmented in some frames. However, the mask will be monotonically increasing in size. Not appropriate for images with drift or for moving objects.

Return type
Mask[int16]

Returns
Mask with objects linked

2.3.4 Extract

class celltk.extract.**Extract**(*images=[]*, *masks=[]*, *channels=[]*, *regions=[]*, *lineages=[]*, *time=None*, *condition='condition'*, *position_id=0*, *min_trace_length=0*, *remove_parent=True*, *output='data_frame'*, *save=True*, *force_rerun=True*, *skip_frames=()*, *_output_id=None*, ***kwargs*)

Parameters

- **images** (Collection[str], default: []) –
- **masks** (Collection[str], default: []) –
- **channels** (Collection[str], default: []) –
- **regions** (Collection[str], default: []) –
- **lineages** (Collection[ndarray], default: []) –
- **time** (Optional[float], default: None) –
- **condition** (str, default: 'condition') –
- **position_id** (int, default: 0) –
- **min_trace_length** (int, default: 0) –
- **remove_parent** (bool, default: True) –
- **output** (str, default: 'data_frame') –
- **save** (bool, default: True) –
- **force_rerun** (bool, default: True) –
- **skip_frames** (Tuple[int], default: ()) –
- **_output_id** (Optional[Tuple[str]], default: None) –

add_derived_metric(*metric_name*, *keys*, *func='sum'*, *inverse=False*, *propagate=False*, *frame_rng=None*, **args*, ***kwargs*)

Calculate additional metrics based on information already in array

Parameters

- **metric_name** (str) – Key to save the metric under
- **keys** (Collection[Tuple[str]]) – One or multiple keys to calculate with. Each key will be used to index an array, ConditionArray[key]. Each key should produce a 2D array when indexed.
- **func** (str, default: 'sum') – Name of numpy function to apply, e.g. 'sum'
- **inverse** (bool, default: False) – If True, repeats the calculation as if the keys were passed in the opposite order and saves in the other keys.
- **propagate** ((str, bool), default: False) – If True, propagates the results of the calculation to the other keys in the array.
- **frame_rng** (Union[int, Tuple[int], None], default: None) – Frames to use in calculation. If int, takes that many frames from start of trace. If tuple, uses passed frames.

Return type
None

add_extra_metric(*name*, *func=None*)

Add custom metrics or metrics from regionprops to array. If function is none, value will just be nan.

Parameters

- **name** (str) – key for the metric in the ConditionArray
- **func** (Optional[Callable], default: None) – If str, name of the metric from skiamge.regionprops. if Callable, function that calculates the metric. Cannot be used if Operation must be saved as YAML before running.

Return type
None

add_filter(*filter_name*, *metric*, *region=0*, *channel=0*, *frame_rng=None*, **args*, ***kwargs*)

Remove cells from array that do not match the filter.

Parameters

- **filter_name** (str) – Options are 'outside', 'inside', 'outside_percentile', 'inside_percentile'.
- **metric** (str) – Name of metric to use. Can be any key in the array.
- **region** (Union[str, int], default: 0) – Name of region to calculate the filter in.
- **channel** (Union[str, int], default: 0) – Name of channel to calculate filter in.
- **frame_rng** (Union[int, Tuple[int], None], default: None) – Frames to use in calculation. If int, takes that many frames from start of trace. If tuple, uses passed frames.

Return type
None

extract_data_from_image(*images*, *masks*, *channels=[]*, *regions=[]*, *lineages=[]*, *time=None*, *condition='default'*, *position_id=None*, *min_trace_length=0*, *skip_frames=()*, *remove_parent=True*, *parent_track=0*)

Extracts data from stacks of images and constructs a ConditionArray.

Parameters

- **images** (Image) – Images to extract data from.
- **masks** (Mask) – Masks to segment images with.
- **channels** (Collection[str], default: []) – Names of channels corresponding to
- **regions** (Collection[str], default: []) – Names of segmented regions corresponding, tracks and masks, in that order.
- **lineages** (Collection[ndarray], default: []) – Lineage files corresponding to masks if provided.
- **time** (Union[float, ndarray, None], default: None) – If int or float, designates time between frames. If array, marks the frame time points.
- **condition** (str, default: 'default') – Name of the condition
- **position_id** (Optional[int], default: None) – Unique identifier if multiple ConditionArrays will share the same condition
- **min_trace_length** (int, default: 0) – All cells with shorter traces will be deleted from the final array.

- **skip_frames** (Tuple[int], default: ()) – Use to specify frames to be skipped. If provided to Pipeline, does not need to be provided again, but must match.
- **remove_parent** (bool, default: True) – If true, parents of cells are not kept in the final ConditionArray.
- **parent_track** (int, default: 0) – If multiple tracks are provided, designates the one to use for lineage tracking

Returns
ConditionArray with data from the images.

Return type
ConditionArray

The following axes dimensions are used: ax 0 - cell locations (nuc, cyto, population, etc.) ax 1 - channels (TRITC, FITC, etc.) ax 2 - metrics (median_int, etc.) ax 3 - cells ax 4 - frames

set_metric_list(*metrics*)

Sets the list of metrics to get. For a possible list, see skimage.regionprops or Extract._possible_metrics.

Parameters
metrics (Collection[str]) – List of metrics to measure from images

Return type
None

Returns
None

Note:

- CellTK can only use the scalar metrics in regionprops.

2.3.5 Evaluate

class celltk.evaluate.**Evaluate**(*output='evaluate'*, ***kwargs*)

Parameters
output (str, default: 'evaluate') –

make_single_cell_stack(*image*, *array*, *cell_id*, *position_id=None*, *window_size=(40, 40)*, *region=None*, *channel=None*)

Crops a window around the coordinates of a single cell in array.

Parameters

- **image** (Image) –
- **array** (Array) –
- **cell_id** (int) –
- **position_id** (Optional[int], default: None) –
- **window_size** (Tuple[int], default: (40, 40)) –
- **region** (Optional[str], default: None) –
- **channel** (Optional[str], default: None) –

Return type
Image

overlay_tracks(*image*, *track*, *boundaries=False*, *mode='inner'*)

Overlays the labels of objects over the reference image.

Parameters

- **image** (Image) –
- **track** (Mask) –
- **boundaries** (bool, default: False) –
- **mode** (str, default: 'inner') –

Return type
Image

save_kept_cells(*track*, *array*)

Creates a mask from the input track that only includes the cells that are present in array.

Parameters

- **track** (Mask) –
- **array** (Array) – ConditionArray containing lineage information

Return type
Mask

Operation is the parent class for all the operations above.

class celltk.core.operation.**Operation**(*images=[]*, *masks=[]*, *arrays=[]*, *output='out'*, *save=True*, *force_rerun=False*, *_output_id=None*, *_split_key='&'*)

Base class for all other Operations.

Parameters

- **images** (Collection[str], default: []) – Names of images to use in this operation
- **masks** (Collection[str], default: []) – Names of masks to use in this operation
- **arrays** (Collection[str], default: []) – Names of arrays to use in this operation
- **output** (str, default: 'out') – Name to save the output stack
- **save** (bool, default: True) – If False, the final result will not be saved to disk.
- **force_rerun** (bool, default: False) – If True, all functions are run even if the output stack is already loaded.
- **_output_id** (Optional[Tuple[str]], default: None) – Private attribute used to track output from the Operation.
- **_split_key** (str, default: '&') – Used to specify outputs from functions that return multiple outputs. For example, if you align two channels the outputs will be saved as 'align&channel000' and 'align&channel001' for _split_key = '&'

__call__(*images=[]*, *masks=[]*, *arrays=[]*, *_return_keys=False*)

__call__ runs operation independently of Pipeline class

Parameters

- **images** (Collection[Image], default: []) –

- **masks** (Collection[Mask], default: []) –
- **arrays** (Collection[Array], default: []) –
- **_return_keys** (bool, default: False) –

Return type
Union[Image, Mask, Array]

add_function(*func*, *, *save_as=None*, *output_type=None*, ***kwargs*)

Adds a function to be run by the Operation

Parameters

- **func** (str) – Name of the function. Must exist in the Operation class.
- **save_as** (Optional[str], default: None) – Name to save the result as. If not set, will not be saved unless it is the last function and Operation.save is True.
- **output_type** (Optional[str], default: None) – If given, changes the default output type to another CellTK type (i.e iimage, mask, track)

Pararm kwargs
All other kwargs are passed to the function. Note that they must be provided as kwargs.

Return type
None

Returns
None

apply_mask(*image*, *mask=None*, *mask_name=None*, **args*, ***kwargs*)

Applies a boolean mask to an image. User can supply a boolean mask or the name of a function from filter_utils.

Parameters

- **image** (Image) – Image to be masked
- **mask** (Optional[Mask], default: None) – Boolean mask of same shape as image
- **mask_name** (Optional[str], default: None) – Name of a function in filter_utils to use to create a boolean mask. If given, any input mask is ignored
- **args** – Passed to mask_name function
- **kwargs** – Passed to mask_name function.

Return type
Stack

Returns
Masked image

binary_threshold(*image*, *lower=None*, *upper=None*, *inside=None*, *outside=None*)

This function is too much. Write a simpler binarize function This is more like constant_thres w/o label as is.

Parameters

- **image** (Image) –
- **lower** (Optional[float], default: None) –
- **upper** (Optional[float], default: None) –

- **inside** (Optional[float], default: None) –
- **outside** (Optional[float], default: None) –

Return type
Image

get_inputs_and_outputs()

Returns all possible inputs and outputs expected by the Operation. Not all will be made or used. Typically this function is used by Pipeline to determine which files to load.

Note:

- This function is expected to change soon in a new version.

Return type
List[List[tuple]]

image_to_mask(*image*)

Convert an Image stack to a Mask stack.

Parameters
image (Image) – Image stack to be typed as a Mask

Return type
Mask

Returns
Input stack with Mask designation.

invert(*image*, *signed_float=False*)

Inverts the intensity of the given image. Wrapper for skimage.util.invert.

Parameters

- **image** (Image) – Image to be inverted.
- **signed_float** (bool, default: False) – Set to True if _____

Return type
Image

make_boolean_mask(*image*, *mask_name='outside'*, **args*, ***kwargs*)

Generates a mask using a function from filter_utils.

Parameters

- **image** (Image) – Image to use for generating mask
- **mask_name** (str, default: 'outside') – Name of the function in filter_utils
- **args** – Passed to mask_name function
- **kwargs** – Passed to mask_name function.

Return type
Mask

Returns
Boolean mask with same shape as input image

mask_to_image(*mask*)

Convert a Mask stack to a Image stack.

Parameters
mask (Mask) – Mask stack to be typed as a Image

Return type
Image

Returns
Input stack with Image designation.

mask_to_seeds(*mask*)

Create seed points at the centroid of each object in mask.

Parameters
mask (Mask) – Mask to serve as template

Return type
Mask

Returns
Mask of same shape as input mask with the pixel corresponding to the centroid of each object labeled.

match_labels_linear(*dest*, *source*)

Transfers labels from source mask to dest mask based on maximizing the area overlap. Objects with no overlap are given a new label starting at max_label_value + 1.

Parameters

- **dest** (Mask) – Mask with objects to be labeled
- **source** (Mask) – Mask with labeled objects to use as template

Return type
Mask

Returns
Labeled mask

regular_seeds(*image*, *n_points=25*, *dtype=<class 'numpy.uint8'>*)

Labels an arbitrary number of approximately evenly-spaced pixels with unique integers.

Parameters

- **image** (Image) – Images to serve as template for making seed mask
- **n_points** (int, default: 25) – Number of pixels to label
- **dtype** (type, default: <class 'numpy.uint8'>) – Data type of the output array.

Return type
Mask

Returns
Mask of same shape as image with n_points pixels labeled with a unique integer.

run_operation(*inputs*, *_return_inputs=True*)

Sets up a generator to run the operation and return results for each function in turn. This function also handles naming and typing the outputs of each function. Typically used by Pipeline. To use an Operation on arrays and get arrays in return, preferably use the __call__ method instead.

Parameters

- **inputs** (ImageContainer) – ImageContainer with all of the inputs required by the Operation.
- **_return_inputs** (bool, default: True) –

Return type
Generator

Returns
A signle generator for the results of each function.

set_logger(*logger*)

Add a custom logger object to log Pipeline operation.

Parameters
logger (Logger) – Custom logging object. Must be type logging.Logger.

Return type
None

Returns
None

2.4 Arrays

class celltk.core.arrays.**ConditionArray**(*regions=['nuc']*, *channels=['TRITC']*, *metrics=['label']*, *cells=[0]*, *frames=[0]*, *name='default'*, *time=None*, *pos_id=0*)

For now, this class is only built by Extract. It's not meant to be built by the user yet. Stores the results from a single condition in an experiment.

Parameters

- **regions** (List[str], default: ['nuc']) –
- **channels** (List[str], default: ['TRITC']) –
- **metrics** (List[str], default: ['label']) –
- **cells** (List[int], default: [0]) –
- **frames** (List[int], default: [0]) –
- **name** (str, default: 'default') –
- **time** (Union[float, ndarray, None], default: None) –
- **pos_id** (int, default: 0) –

add_metric_slots(*name*)

Expands the ConditionArry to make room for more metrics.

Note:

- This method must be used before attempting to add new metrics to the ConditionArray.

Parameters
name (List[str]) – List of names of the metrics to add

Return type
None

Returns
None

`property channels`

Names of the channels in the data array.

`property condition`

Name of the data array

`property coordinate_dimensions`

Dictionary of with coordinate names as keys and coordinate length as values

`property coordinates`

Names of the coordinates (axes) in the data array

`property dtype`

Data type of the data array. Usually double.

`filter_cells`(*mask=None*, *key=None*, *delete=True*, **args*, ***kwargs*)

Uses an arbitrary mask or a saved mask (key) to filter cells from the data. If delete, the underlying structure is changed, otherwise, the data are only returned.

Parameters

- **mask** (Optional[ndarray], default: None) – A boolean mask to filter cells with. Can be 1D, 2D or 5D.
- **key** (Optional[str], default: None) – Name of a saved mask to use for filtering cells. Overwrites mask if provided.
- **delete** (bool, default: True) – If True, cells are removed in the base array. Otherwise they are only removed in the array that is returned.

Returns
array with cells designated by maks or key removed.

Return type
np.ndarray

`filter_peaks`(*value_key*, *metrics*, *thresholds*, *kwargs=[{}]*, *peak_key=None*, *propagate=True*)

Removes segmented peaks based on arbitrary peak criteria. See celltk.utils.peak_utils for more information about possible metrics to use

Parameters

- **value_key** (Tuple[int, str]) – Key to the traces used to calculate the peak metrics.
- **metrics** (Collection[str]) – Names of the metrics to use for segmentation. See PeakHelper in celltk.utils.peak_utils.
- **thresholds** (Collection[str]) – Lower threshold for the metrics given. If a peak metric is less than the threshold, it will be removed. Must be the same length as metrics.
- **kwargs** (Collection[dict], default: [{}]) – Collection of dictionaries containing kwargs for the metrics given. If given, should be same length as metrics.
- **peak_key** (Optional[Tuple[int, str]], default: None) – Key to the peak labels. If not given, will attempt to find peak labels based on the value key

- **propagate** (bool, default: True) – If True, propagates filtered peak labels to the other keys in ConditionArray

Return type
None

Returns
None

generate_mask(*function*, *metric*, *region=0*, *channel=0*, *frame_rng=None*, *key=None*, **args*, ***kwargs*)

Generate a mask to remove cells using an arbitrary filter.

Parameters

- **function** – If str, name of function in filter_utils. Otherwise, should be a Callable that inputs a 2D array and returns a 2D boolean array.
- **metric** – Name of metric to use. Can be any key in the array.
- **region** – Name of region to calculate the filter in.
- **channel** – Name of channel to calculate filter in.
- **frame_rng** – Frames to use in calculation. If int, takes that many frames from start of trace. If tuple, uses passed frames.
- **key** – If given, saves the mask in ConditionArray as key.
- **args** – passed to function
- **kwargs** – passed to function

Returns
2D boolean array that masks cells outside filter

get_mask(*key*)

Returns a saved mask. Masks are generated and saved using ConditionArray().generate_mask()

Parameters
key (str) – Name of mask to retreive

Returns
The saved boolean mask

Return type
np.ndarray

interpolate_nans(*keys=None*)

Linear interpolation of nans for each cell. Modification is done in-place.

Parameters
keys (Optional[Collection[tuple]], default: None) – keys that will have nans removed. Each key should be a tuple of strings with length=3

Return type
None

Returns
None

property keys

All the possible keys that can be used to index the data array.

classmethod load(*path*)

Load an hdf5 file and convert to a ConditionArray.

Parameters
path (str) – Path to the hdf5 file to be loaded.

Returns
Loaded ConditionArray

Return type
ConditionArray

mark_active_cells(*key*, *thres=1*, *propagate=True*)

Uses peak labels to mark in what frames cells are active

Parameters

- **key** (Tuple[int, str]) – Key defining peak labels
- **thres** (float, default: 1) – Leave as 1, not currently used
- **propagate** (bool, default: True) – if True, propagate active marks to other keys in ConditionArray

Return type
None

Returns
None

property metrics

Names of the metrics in the data array.

property ncells

Number of cells present in array

property ndim

Number of dimensions in the data array. Should equal 5.

predict_peaks(*key*, *model=None*, *weight_path=None*, *propagate=True*, *segment=True*, ***kwargs*)

Uses a UNet-based neural net to predict peaks in the traces defined by key. Adds two keys to ConditionArray, 'slope_prob' and 'plateau_prob'. If *segment* is True, also adds a 'peaks' key. 'slope_prob' is the probability that a point is on the upward or downward slope of a peak. 'plateau_prob' is the probability that a point is at the top of a peak.

Parameters

- **key** (Tuple[int, str]) – Key to the traces to predict peaks with. Must return a 2D array.
- **model** (Optional[UPeakModel], default: None) – An instantiated UPeakModel to use
- **weight_path** (Optional[str], default: None) – Path to the model weights to use for a new UPeakModel instance
- **propagate** (bool, default: True) – If True, propagates peak probabilities to the other keys in ConditionArray
- **segment** (bool, default: True) – If True, uses a watershed-based segmentation to label peaks based on the predictions.
- **kwargs** – Passed to segmentation function. See utils.peak_utils.segment_peaks_agglomeration.

Return type
None

Returns
None

Raises
ValueError – If neither model or weights are provided.

propagate_values(*key*, *prop_to='both'*)

Propagates metric value to other keys in ConditionArray.

Parameters

- **key** (Tuple[str]) – Key to metric containing the values to propagate
- **prop_to** (str, default: 'both') – Define the keys to propagate values to. Options are 'channel', 'region', or 'both'.

Return type
None

Returns
None

property regions

Names of the regions in the data array.

remove_parents(*parent_daughter*, *cell_index*)

Returns 1D boolean mask to remove traces of cells that divided. Daughter cell traces are kept. Use with reshape_mask to remove parent cells. Typically called by Extract.

Parameters

- **parent_daughter** (Dict) – Dictionary of parent_label : daughter_label
- **cell_index** (Dict) – Dictionary of cell_label : cell_index_in_array

Returns
Boolean mask with False in the rows of parent cells

Return type
np.ndarray

remove_short_traces(*min_trace_length*)

Removes cells with less than min_trace_length non-nan values. Uses label as the metric to determine non-nan values. Typically called by Extract.

Parameters
min_trace_length (int) – Shortest trace that should not be deleted

Returns
array with short traces removed

Return type
np.ndarray

reshape_mask(*mask*)

Takes in a 1D, 2D, or 5D boolean mask and casts to tuple of 5-dimensional indices. Use this to apply a 1D or 2D mask to self._arr.

Note:

- Always assumes that filtering is to happen in cell axis.

Parameters
mask (ndarray) – Boolean mask to be used as filter.

Returns
Indices that can be used to index ConditionArray

Return type
Tuple

save(*path*)

Saves ConditionArray to an hdf5 file.

Parameters
path (str) – Absolute path to save the file.

Return type
None

Returns
None

set_condition(*condition*)

Updates name of the ConditionArray.

Parameters
condition (str) – New name of the ConditionArray

Return type
None

Returns
None

set_position_id(*pos=None*)

Adds unique identifiers to the cells in ConditionArray. Typically called by Pipeline or ExperimentArray.

Parameters
pos (Optional[int], default: None) – Integer or string identifying the position

Return type
None

Returns
None

set_time(*time*)

Define the time axis. Time can be given as a frame interval or an array specifying the time for each frame.

Parameters
time (Union[float, ndarray]) – If int or float, designates time between frames. If array, marks the frame time points.

Return type
None

Returns
None

property shape

Shape of the data array

class celltk.core.arrays.**ExperimentArray**(*arrays=None*, *name=None*, *time=None*)

Base class to create arrays that can store an almost arbitrary number of ConditionArrays. Typically made by Orchestrator.build_experiment_file()

Parameters

- **arrays** (Optional[List[*ConditionArray*]], default: None) –
- **name** (Optional[str], default: None) –
- **time** (Optional[float], default: None) –

add_metric_slots(*name*)

Expands each ConditionArry to make room for more metrics.

Note:

- This method must be used before attempting to add new metrics to the ConditionArray.

Parameters
name (List[str]) – List of names of the metrics to add

Return type
None

Returns
None

property channels

Returns list of the names of the channels in each ConditionArray.

property conditions: List

Returns list of the name of each ConditionArray.

Return type
List

property coordinates

Returns the names of the coordinates of each ConditionArray.

property dtype

Returns list of the data type of each ConditionArray.

filter_cells(*mask=None*, *key=None*, *delete=True*, **args*, ***kwargs*)

Uses an arbitrary mask or a saved mask (key) to filter cells from each ConditionArray. If delete, the underlying data are changed. Otherwise, the filtered data are only returned.

Parameters

- **mask** (Optional[List[ndarray]], default: None) – A boolean mask to filter cells with. Can be 1D, 2D or 5D.
- **key** (Optional[str], default: None) – Name of a saved mask to use for filtering cells. Overwrites mask if provided.
- **delete** (bool, default: True) – If True, cells are removed in the base array. Otherwise they are only removed in the array that is returned.

- **args** – Passed to filtering function.
- **kwargs** – Passed to filtering function.

Returns
array with cells designated by maks or key removed.

Return type
np.ndarray

filter_peaks(*value_key*, *metrics*, *thresholds*, *kwargs=[{}]*, *peak_key=None*, *propagate=True*)

Removes segmented peaks based on arbitrary peak criteria. See celltk.utils.peak_utils for more information about possible metrics to use

Parameters

- **value_key** (Tuple[int, str]) – Key to the traces used to calculate the peak metrics.
- **metrics** (Collection[str]) – Names of the metrics to use for segmentation. See PeakHelper in celltk.utils.peak_utils.
- **thresholds** (Collection[str]) – Lower threshold for the metrics given. If a peak metric is less than the threshold, it will be removed. Must be the same length as metrics.
- **kwargs** (Collection[dict], default: [{}]) – Collection of dictionaries containing kwargs for the metrics given. If given, should be same length as metrics.
- **peak_key** (Optional[Tuple[int, str]], default: None) – Key to the peak labels. If not given, will attempt to find peak labels based on the value key
- **propagate** (bool, default: True) – If True, propagates filtered peak labels to the other keys in ConditionArray

Return type
None

Returns
None

generate_mask(*function*, *metric*, *region=0*, *channel=0*, *frame_rng=None*, *key=None*, *individual=True*, **args*, ***kwargs*)

Generates a boolean mask for each ConditionArray based on an arbitrary filter.

Parameters

- **function** – If str, name of function in filter_utils. Otherwise, should be a Callable that inputs a 2D array and returns a 2D boolean array.
- **metric** – Name of metric to use. Can be any key in the array.
- **region** – Name of region to calculate the filter in.
- **channel** – Name of channel to calculate filter in.
- **frame_rng** – Frames to use in calculation. If int, takes that many frames from start of trace. If tuple, uses passed frames.
- **key** – If given, saves the mask in ConditionArray as key.
- **individual** – If true, the filter is calculated for each ConditionArray independently. Otherwise, calculated on the whole data set, then applied to ConditionArrays.
- **args** – passed to function
- **kwargs** – passed to function

Returns
List of 2D boolean array to masks cells outside filter

`interpolate_nans`(*keys=None*)

Linear interpolation of nans for each cell in each ConditionArray.
Modification is done in-place.

Parameters
keys (`Optional[Collection[tuple]]`, default: `None`) – keys that will have nans removed. Each key should be a tuple of strings with length=3

Return type
`None`

Returns
None

`items`()
Use to iterate through the key and array for each ConditionArray.

`keys`()
Use to iterate through all the keys in ExperimentArray.

`classmethod load`(*path*)
Load an ExperimentArray from an hdf5 file.

Parameters
path (`str`) – Path to the hdf5 file

Returns
ExperimentArray

Return type
ExperimentArray

`load_condition`(*array*, *name=None*, *pos_id=None*)
Adds a ConditionArray to the ExperimentArray from an hdf5 file. The new ConditionArray gets saved as name + pos_id if provided, otherwise uses the name saved in the hdf5 file.

Parameters

- **array** (`Union[str, ConditionArray]`) – ConditionArray or path to the hdf5 file with ConditionArray
- **name** (`Optional[str]`, default: `None`) – Name of the ConditionArray to be loaded.
- **pos_id** (`Optional[int]`, default: `None`) – Unique identifier for the ConditionArray.

Return type
`None`

Returns
None

`mark_active_cells`(*key*, *thres=1*, *propagate=True*)
Uses peak labels to mark in what frames cells are active in each ConditionArray.

Parameters

- **key** (`Tuple[int, str]`) – Key defining peak labels
- **thres** (`float`, default: 1) – Leave as 1, not currently used

- **propagate** (bool, default: True) – if True, propagate active marks to other keys in ConditionArray

Return type
None

Returns
None

merge_conditions()

Concatenate ConditionArrays with matching conditions. If no arrays have matching conditions, nothing is done. If matching conditions are found, looks for position map to label each uniquely, or will just number them in the order that they were saved in the ExperimentArray. Arrays are concatenated along the cell axis.

Return type
None

Returns
None

Note:

- Any masks that have been saved in the individual ConditionArrays will be lost.

---

property metrics

Returns list of the names of the metrics in each ConditionArray.

property ncells

Returns list of the number of cells in each ConditionArray.

property ndim

Returns list of the number of dimensoins of each ConditionArray.

predict_peaks(*key*, *weight_path*, *propagate=True*, *segment=True*, ***kwargs*)

Uses a UNet-based neural net to predict peaks in the traces defined by key in each ConditionArray. Adds two keys to ConditionArrays, 'slope_prob' and 'plateau_prob'. If *segment* is True, also adds a 'peaks' key. 'slope_prob' is the probability that a point is on the upward or downward slope of a peak. 'plateau_prob' is the probability that a point is at the top of a peak.

Parameters

- **key** (Tuple[int, str]) – Key to the traces to predict peaks with. Must return a 2D array.
- **weight_path** (str) – Path to the model weights to use for a new UPeakModel instance
- **propagate** (bool, default: True) – If True, propagates peak probabilities to the other keys in ConditionArray
- **segment** (bool, default: True) – If True, uses a watershed-based segmentation to label peaks based on the predictions.
- **kwargs** – Passed to segmentation function.

Return type
None

Returns
None

property regions

Returns list of the names of the regions in each ConditionArray.

remove_empty_sites()

Removes all sites that have one or more empty dimensions.

Return type
None

Returns
None

remove_short_traces(*min_trace_length=0*)

Applies a filter to each condition to remove cells with fewer non-nan frames than min_trace_length. The 'label' metric is used for determining non-nan frames.

Parameters
min_trace_length (int, default: 0) – Minimum number of non-nan frames allowed

Return type
None

Returns
None

save(*path*)

Saves all the Conditions in Experiment to an hdf5 file.

Loads the hdf5 file for each condition and then saves them in a single hdf5 file at path. Runs merge_conditions() first to ensure data doesn't get overwritten.

Parameters
path (str) – Path to the location where file should be saved.

Return type
None

Returns
None

Raise
ValueError if any cell or frame is greater than 2 ** 16.

set_conditions(*condition_map={}*)

Updates names of all of the ConditionArrays

Parameters

- **condition** – Dict of current_name : new_name for each ConditionArray
- **condition_map** (Dict[str, str], default: {}) –

Return type
None

Returns
None

set_time(*time=None*)

Define the time axis in each ConditionArray. Time can be given as a frame interval or an array specifying the time for each frame.

Parameters
time (Optional[float], default: None) – If int or float, designates time between frames. If array, marks the frame time points.

Return type
None

Returns
None

property shape

Returns list of the shape of each ConditionArray.

property time

Returns list of the time axis of each ConditionArray.

values()

Use to iterate through all of the ConditionArrays.

2.5 Plotting

class celltk.utils.plot_utils.**PlotHelper**

Helper class for making plots. Most functions are simple wrappers around Plotly functions.

bar_plot(*arrays*, *keys=[]*, *estimator=None*, *err_estimator=None*, *ax_labels=None*, *colors=None*, *alpha=1.0*, *orientation='v'*, *barmode='group'*, *add_cell_numbers=True*, *legend=True*, *figure=None*, *figsize=(None, None)*, *margin='auto'*, *title=None*, *x_label=None*, *y_label=None*, *x_limit=None*, *y_limit=None*, *tick_size=None*, *axis_label_size=None*, *widget=False*, *row=None*, *col=None*, ***kwargs*)

Builds a Plotly Figure object plotting bars from the given arrays. Each array is interpreted as a separate condition and is plotted in a different color.

Parameters

- **arrays** (Collection[ndarray]) – List of arrays to plot. Assumed structure is n_cells x n_features. Each feature is plotted as a separate bar group. Arrays must be two-dimensional, so if only one sample, use np.newaxis or np.expand_dims.
- **keys** (Collection[str], default: []) – Names corresponding to the data arrays. If not provided, the keys will be integers.
- **estimator** (Union[Callable, str, partial, None], default: None) – Function for aggregating observations from multiple cells. For example, if estimator is np.mean, the mean of all of the cells will be plotted instead of a bar for each cell. Can be a function, name of numpy function, name of function in estimator_utils, or a functools.partial object. If a function or functools.partial object, should input a 2D array and return a 1D array.
- **err_estimator** (Union[Callable, str, partial, None], default: None) – Function for estimating error bars. Can be a function, name of numpy function, name of function in estimator_utils, or a functools.partial object. If a function or functools.partial object, should input a 2D array and return a 1D or 2D array. If output is 1D, errors will be symmetric If output is 2D, the first dimension is used for the high error and second dimension is used for the low error.
- **colors** (Union[str, Collection[str], None], default: None) – Name of a color palette or map to use. Searches first in seaborn/matplotlib, then in Plotly to find the color map.

If not provided, the color map will be glasbey. Can also be list of named CSS colors or hexadecimal or RGBA strings.

- **alpha** (float, default: 1.0) – Opacity of the marker fill colors.
- **ax_labels** (Optional[Collection[str]], default: None) – Labels for the categorical axis.
- **orientation** (str, default: 'v') – Orientation of the bar plot.
- **barmode** (str, default: 'group') – Keyword argument describing how to group the bars. Options are 'group', 'overlay', 'relative', and 'stack'.
- **add_cell_numbers** (bool, default: True) – If True, adds the number of cells to each key in keys.
- **legend** (bool, default: True) – If False, no legend is made on the plot.
- **figure** (Union[Figure, FigureWidget, None], default: None) – If a go.Figure object is given, will be used to make the plot instead of a blank figure.
- **figsize** (Tuple[int], default: (None, None)) – Height and width of the plot in pixels. Must be a tuple of length two. To leave height or width unchanged, set as None.
- **margin** (str, default: 'auto') – Set the size of the margins. If 'auto', all margins are set to defualt values. If 'zero' or 'tight', margins are removed. If a number is given, sets all margins to that number.
- **title** (Optional[str], default: None) – Title to add to the plot
- **x_label** (Optional[str], default: None) – Label of the x-axis
- **y_label** (Optional[str], default: None) – Label of the y-axis
- **x_limit** (Optional[Tuple[float]], default: None) – Initial limits for the x-axis. Can be changed if the plot is saved as an HTML object. Only applies if orientation is horizontal.
- **y_limit** (Optional[Tuple[float]], default: None) – Initial limits for the y-axis. Can be changed if the plot is saved as an HTML object. Only applies if orientation is vertical.
- **tick_size** (Optional[float], default: None) – Size of the font of the axis tick labels.
- **axis_label_size** (Optional[float], default: None) – Size of the font of the axis label.
- **widget** (bool, default: False) – If True, returns a go.FigureWidget object instead of a go.Figure object.
- **row** (Optional[int], default: None) – If Figure has multiple subplots, specifies which row to use for the plot. Indexing starts at 1. Note that some formatting args (such as figsize) may still be applied to all subplots. Both row and col must be provided together.
- **col** (Optional[int], default: None) – If Figure has multiple subplots, specifies which col to use for the plot. Indexing starts at 1. Note that some formatting args (such as figsize) may still be applied to all subplots. Both row and col must be provided together.
- **kwargs** – Depending on name, passed to go.Bar or to go.Figure.update_traces(). The following kwargs are passed to go.Bar: 'hoverinfo', 'marker', 'width'.

Return type
Union[Figure, FigureWidget]

Returns
Figure object

Raises

- **AssertionError** – If not all items in arrays are np.ndarray.
- **AssertionError** – If orientation is a disallowed value.
- **AssertionError** – If figsize is not a tuple of length two.

box_plot(*arrays*, *keys=[]*, *colors=None*, *line_colors=None*, *alpha=1.0*, *orientation='v'*, *show_points=False*, *jitter=None*, *show_mean=False*, *width=0*, *add_cell_numbers=True*, *legend=True*, *figure=None*, *figsize=(None, None)*, *margin='auto'*, *title=None*, *x_label=None*, *y_label=None*, *x_limit=None*, *y_limit=None*, *tick_size=None*, *axis_label_size=None*, *widget=False*, *row=None*, *col=None*, ***kwargs*)

Builds a Plotly go.Figure object with box and whisker distributions for each of the arrays given.

Parameters

- **arrays** (Collection[ndarray]) – List of arrays to plot. Arrays are assumed to be one dimensional.
- **keys** (Collection[str], default: []) – Names corresponding to the data arrays. If not provided, the keys will be integers.
- **colors** (Union[str, Collection[str], None], default: None) – Name of a color palette or map to use to fill boxes. Searches first in seaborn/matplotlib, then in Plotly to find the color map. If not provided, the color map will be glasbey. Can also be list of named CSS colors or hexadecimal or RGBA strings.
- **line_colors** (Union[str, Collection[str], None], default: None) – Name of a color palette or map to use for outlines of boxes. Searches first in seaborn/matplotlib, then in Plotly to find the color map. If not provided, will use same colormap as colors. Can also be list of named CSS colors or hexadecimal or RGBA strings.
- **alpha** (float, default: 1.0) – Opacity of the fill color of the box plots as a float in the range [0, 1].
- **orientation** (str, default: 'v') – Orientation of the box plot.
- **show_points** (Union[str, bool], default: False) – If True, individual data points are overlaid over the box plot. If 'outliers' or 'suspectedoutliers', only points that lie outside a certain distance from the interquartile range are shown.
- **jitter** (Optional[float], default: None) – Sets the jitter in the sample points drawn. Ranges from 0-1. If 0, points are aligned on distribution axis, if 1, points are drawn in random jitter the width of the boxes.
- **show_mean** (bool, default: False) – If True, dashed line is plotted at the mean value.
- **width** (float, default: 0) – Sets the width of the boxes. If 0, width is automatically set.
- **add_cell_numbers** (bool, default: True) – If True, adds the number of cells to each key in keys.
- **legend** (bool, default: True) – If False, no legend is made on the plot.
- **figure** (Union[Figure, FigureWidget, None], default: None) – If a go.Figure object is given, will be used to make the plot instead of a blank figure.
- **figsize** (Tuple[int], default: (None, None)) – Height and width of the plot in pixels. Must be a tuple of length two. To leave height or width unchanged, set as None.
- **margin** (str, default: 'auto') – Set the size of the margins. If 'auto', all margins are set to defualt values. If 'zero' or 'tight', margins are removed. If a number is given, sets all margins to that number.
- **title** (Optional[str], default: None) – Title to add to the plot

- **x_label** (Optional[str], default: None) – Label of the x-axis
- **y_label** (Optional[str], default: None) – Label of the y-axis
- **x_limit** (Optional[Tuple[float]], default: None) – Initial limits for the x-axis. Can be changed if the plot is saved as an HTML object. Only applies if orientation is horizontal.
- **y_limit** (Optional[Tuple[float]], default: None) – Initial limits for the y-axis. Can be changed if the plot is saved as an HTML object. Only applies if orientation is veritcal.
- **tick_size** (Optional[float], default: None) – Size of the font of the axis tick labels.
- **axis_label_size** (Optional[float], default: None) – Size of the font of the axis label.
- **widget** (bool, default: False) – If True, returns a go.FigureWidget object instead of a go.Figure object.
- **row** (Optional[int], default: None) – If Figure has multiple subplots, specifies which row to use for the plot. Indexing starts at 1. Note that some formatting args (such as figsize) may still be applied to all subplots. Both row and col must be provided together.
- **col** (Optional[int], default: None) – If Figure has multiple subplots, specifies which col to use for the plot. Indexing starts at 1. Note that some formatting args (such as figsize) may still be applied to all subplots. Both row and col must be provided together.
- **kwargs** – Depending on name, passed to go.Box or to go.Figure.update_traces(). The following kwargs are passed to go.Box: 'mean', 'lowerfence', 'upperfence', 'notched', 'notchspan', 'notchwidth', 'q1', 'q3', 'quartilemethod', 'sd', 'whiskerwidth'.

Return type
Union[Figure, FigureWidget]

Returns
Figure object

Raises

- **AssertionError** – If not all entries in arrays or neg_arrays are np.ndarray
- **AssertionError** – If any entry in arrays or neg_arrays have more than one dimension.
- **AssertionError** – If neg_arrays is given, and len(arrays) != len(neg_arrays)
- **AssertionError** – If figsize is not a tuple of length two.

contour2d_plot(*x_array*, *y_array*, *colorscale='viridis'*, *fill=True*, *zmin=None*, *zmid=None*, *zmax=None*, *robust_z=False*, *width=0.5*, *xbinsize=None*, *ybinsize=None*, *histfunc='count'*, *histnorm=''*, *figure=None*, *figsize=(None, None)*, *margin='auto'*, *title=None*, *x_label=None*, *y_label=None*, *x_limit=None*, *y_limit=None*, *tick_size=None*, *axis_label_size=None*, *widget=False*, *row=None*, *col=None*, ***kwargs*)

Builds a Plotly go.Figure object with a two dimensional density contour heatmap of the provided arrays. This function is just a very thin wrapper around go.Heatmap2dContour.

Parameters

- **x_array** (ndarray) – Array containing observations for the data on the x-axis. Expected to be one dimensional.
- **y_array** (ndarray) – Array containing observations for the data on the y-axis. Expected to be one dimensional.
- **colorscale** (str, default: 'viridis') – The colorscale to make the heatmap in. Options are more limited than the options for colors. Options include: "Blackbody", "Bluered",

"Blues", "Cividis", "Earth", "Electric", "Greens", "Greys", "Hot", "Jet", "Picnic", "Portland", "Rainbow", "RdBu", "Reds", "Viridis", "YlGnBu", and "YlOrRd".

- **fill** (bool, default: True) – If True, space between contour lines is filled with color, otherwise, only the lines are colored.
- **zmin** (Optional[float], default: None) – Sets the lower bound of the color domain. If given, zmax must also be given.
- **zmid** (Optional[float], default: None) – Sets the midpoint of the color domain by setting zmin and zmax to be equidistant from this point.
- **zmax** (Optional[float], default: None) – Sets the upper bound of the color domain. If given, zmin must also be given.
- **robust_z** (bool, default: False) – If True, uses percentiles to set zmin and zmax instead of extremes of the dataset.
- **width** (float, default: 0.5) – Width of the contour lines.
- **xbinsize** (Optional[float], default: None) – Size of the bins along the x-axis.
- **ybinsize** (Optional[float], default: None) – Size of the bins along the y-axis.
- **histfunc** (str, default: 'count') – Specifies the binning function used for this histogram trace. If "count", the histogram values are computed by counting the number of values lying inside each bin. Can also be "sum", "avg", "min", "max".
- **histnorm** (str, default: '') – Specifies the type of normalization used for this histogram trace. If "", the span of each bar corresponds to the number of occurrences If "percent" / "probability", the span of each bar corresponds to the percentage / fraction of occurrences with respect to the total number of sample points (here, the sum of all bin HEIGHTS equals 100% / 1). If "density", the span of each bar corresponds to the number of occurrences in a bin divided by the size of the bin interval. If probability density, the area of each bar corresponds to the probability that an event will fall into the corresponding bin (here, the sum of all bin AREAS equals 1).
- **figure** (Union[Figure, FigureWidget, None], default: None) – If a go.Figure object is given, will be used to make the plot instead of a blank figure.
- **figsize** (Tuple[int], default: (None, None)) – Height and width of the plot in pixels. Must be a tuple of length two. To leave height or width unchanged, set as None.
- **margin** (str, default: 'auto') – Set the size of the margins. If 'auto', all margins are set to defualt values. If 'zero' or 'tight', margins are removed. If a number is given, sets all margins to that number.
- **title** (Optional[str], default: None) – Title to add to the plot
- **x_label** (Optional[str], default: None) – Label of the x-axis
- **y_label** (Optional[str], default: None) – Label of the y-axis
- **x_limit** (Optional[Tuple[float]], default: None) – Initial limits for the x-axis. Can be changed if the plot is saved as an HTML object. Only applies if orientation is horizontal.
- **y_limit** (Optional[Tuple[float]], default: None) – Initial limits for the y-axis. Can be changed if the plot is saved as an HTML object. Only applies if orientation is veritcal.
- **tick_size** (Optional[float], default: None) – Size of the font of the axis tick labels.
- **axis_label_size** (Optional[float], default: None) – Size of the font of the axis label.

- **widget** (bool, default: False) – If True, returns a go.FigureWidget object instead of a go.Figure object.
- **row** (Optional[int], default: None) – If Figure has multiple subplots, specifies which row to use for the plot. Indexing starts at 1. Note that some formatting args (such as figsize) may still be applied to all subplots. Both row and col must be provided together.
- **col** (Optional[int], default: None) – If Figure has multiple subplots, specifies which col to use for the plot. Indexing starts at 1. Note that some formatting args (such as figsize) may still be applied to all subplots. Both row and col must be provided together.
- **kwargs** – Passed to go.Heatmap2dContour

Return type
Union[Figure, FigureWidget]

Returns
Figure object

Raises

- **AssertionError** – If x_array or y_array are more than one dimensional.
- **AssertionError** – If figsize is not a tuple of length two.

heatmap2d_plot(*x_array*, *y_array*, *colorscale='viridis'*, *zmin=None*, *zmid=None*, *zmax=None*, *robust_z=False*, *xbinsize=None*, *ybinsize=None*, *histfunc='count'*, *histnorm=''*, *figure=None*, *figsize=(None, None)*, *margin='auto'*, *title=None*, *x_label=None*, *y_label=None*, *x_limit=None*, *y_limit=None*, *tick_size=None*, *axis_label_size=None*, *widget=False*, *row=None*, *col=None*, ***kwargs*)

Builds a Plotly go.Figure object with a two dimensional density heatmap of the provided arrays. This function is just a very thin wrapper around go.Heatmap2d.

Parameters

- **x_array** (ndarray) – Array containing observations for the data on the x-axis. Expected to be one dimensional.
- **y_array** (ndarray) – Array containing observations for the data on the y-axis. Expected to be one dimensional.
- **colorscale** (str, default: 'viridis') – The colorscale to make the heatmap in. Options are more limited than the options for colors. Options include: “Blackbody”, “Bluered”, “Blues”, “Cividis”, “Earth”, “Electric”, “Greens”, “Greys”, “Hot”, “Jet”, “Picnic”, “Portland”, “Rainbow”, “RdBu”, “Reds”, “Viridis”, “YlGnBu”, and “YlOrRd”.
- **zmin** (Optional[float], default: None) – Sets the lower bound of the color domain. If given, zmax must also be given.
- **zmid** (Optional[float], default: None) – Sets the midpoint of the color domain by setting zmin and zmax to be equidistant from this point.
- **zmax** (Optional[float], default: None) – Sets the upper bound of the color domain. If given, zmin must also be given.
- **robust_z** (bool, default: False) – If True, uses percentiles to set zmin and zmax instead of extremes of the dataset.
- **xbinsize** (Optional[float], default: None) – Size of the bins along the x-axis.
- **ybinsize** (Optional[float], default: None) – Size of the bins along the y-axis.

- **histfunc** (str, default: 'count') – Specifies the binning function used for this histogram trace. If "count", the histogram values are computed by counting the number of values lying inside each bin. Can also be "sum", "avg", "min", "max".
- **histnorm** (str, default: '') – Specifies the type of normalization used for this histogram trace. If "", the span of each bar corresponds to the number of occurrences If "percent" / "probability", the span of each bar corresponds to the percentage / fraction of occurrences with respect to the total number of sample points (here, the sum of all bin HEIGHTS equals 100% / 1). If "density", the span of each bar corresponds to the number of occurrences in a bin divided by the size of the bin interval. If probability density, the area of each bar corresponds to the probability that an event will fall into the corresponding bin (here, the sum of all bin AREAS equals 1).
- **figure** (Union[Figure, FigureWidget, None], default: None) – If a go.Figure object is given, will be used to make the plot instead of a blank figure.
- **figsize** (Tuple[int], default: (None, None)) – Height and width of the plot in pixels. Must be a tuple of length two. To leave height or width unchanged, set as None.
- **margin** (str, default: 'auto') – Set the size of the margins. If 'auto', all margins are set to defualt values. If 'zero' or 'tight', margins are removed. If a number is given, sets all margins to that number.
- **title** (Optional[str], default: None) – Title to add to the plot.
- **x_label** (Optional[str], default: None) – Label of the x-axis
- **y_label** (Optional[str], default: None) – Label of the y-axis
- **x_limit** (Optional[Tuple[float]], default: None) – Initial limits for the x-axis. Can be changed if the plot is saved as an HTML object. Only applies if orientation is horizontal.
- **y_limit** (Optional[Tuple[float]], default: None) – Initial limits for the y-axis. Can be changed if the plot is saved as an HTML object. Only applies if orientation is veritcal.
- **tick_size** (Optional[float], default: None) – Size of the font of the axis tick labels.
- **axis_label_size** (Optional[float], default: None) – Size of the font of the axis label.
- **widget** (bool, default: False) – If True, returns a go.FigureWidget object instead of a go.Figure object.
- **row** (Optional[int], default: None) – If Figure has multiple subplots, specifies which row to use for the plot. Indexing starts at 1. Note that some formatting args (such as figsize) may still be applied to all subplots. Both row and col must be provided together.
- **col** (Optional[int], default: None) – If Figure has multiple subplots, specifies which col to use for the plot. Indexing starts at 1. Note that some formatting args (such as figsize) may still be applied to all subplots. Both row and col must be provided together.
- **kwargs** – Passed to go.Histogram2d.

Return type
Union[Figure, FigureWidget]

Returns
Figure object

Raises

- **AssertionError** – If x_array or y_array are more than one dimensional.
- **AssertionError** – If figsize is not a tuple of length two.

heatmap_plot(*array*, *colorscale='viridis'*, *zmin=None*, *zmid=None*, *zmax=None*, *robust_z=False*, *reverse=False*, *sort=None*, *figure=None*, *figsize=(None, None)*, *margin='auto'*, *title=None*, *x_label=None*, *y_label=None*, *x_limit=None*, *y_limit=None*, *tick_size=None*, *axis_label_size=None*, *widget=False*, *row=None*, *col=None*, ***kwargs*)

Builds a Plotly go.Figure object with a heatmap of the provided array. This function is just a very thin wrapper around go.Heatmap.

Parameters

- **array** (ndarray) – Array from which to make the heatmap.
- **colorscale** (str, default: 'viridis') – The colorscale to make the heatmap in. Options are more limited than the options for colors. Options include: "Blackbody", "Bluered", "Blues", "Cividis", "Earth", "Electric", "Greens", "Greys", "Hot", "Jet", "Picnic", "Portland", "Rainbow", "RdBu", "Reds", "Viridis", "YlGnBu", and "YlOrRd".
- **zmin** (Optional[float], default: None) – Sets the lower bound of the color domain. If given, zmax must also be given.
- **zmid** (Optional[float], default: None) – Sets the midpoint of the color domain by setting zmin and zmax to be equidistant from this point.
- **zmax** (Optional[float], default: None) – Sets the upper bound of the color domain. If given, zmin must also be given.
- **robust_z** (bool, default: False) – If True, uses percentiles to set zmin and zmax instead of extremes of the dataset.
- **reverse** (bool, default: False) – If True, the color mapping is reversed.
- **sort** (Optional[str], default: None) – If the name of a distance metric, will sort the array according to that metric before producing the heatmap. Valid values are 'cityblock', 'cosine', 'euclidean', 'l1', 'l2', 'manhattan', 'braycurtis', 'canberra', 'chebyshev', 'correlation', 'dice', 'hamming', 'jaccard', 'kulsinski', 'mahalanobis', 'minkowski', 'rogerstanimoto', 'russellrao', 'seuclidean', 'sokalmichener', 'sokalsneath', 'sqeuclidean', and 'yule'.
- **figure** (Union[Figure, FigureWidget, None], default: None) – If a go.Figure object is given, will be used to make the plot instead of a blank figure.
- **figsize** (Tuple[int], default: (None, None)) – Height and width of the plot in pixels. Must be a tuple of length two. To leave height or width unchanged, set as None.
- **margin** (str, default: 'auto') – Set the size of the margins. If 'auto', all margins are set to defualt values. If 'zero' or 'tight', margins are removed. If a number is given, sets all margins to that number.
- **title** (Optional[str], default: None) – Title to add to the plot
- **x_label** (Optional[str], default: None) – Label of the x-axis
- **y_label** (Optional[str], default: None) – Label of the y-axis
- **x_limit** (Optional[Tuple[float]], default: None) – Initial limits for the x-axis. Can be changed if the plot is saved as an HTML object. Only applies if orientation is horizontal.
- **y_limit** (Optional[Tuple[float]], default: None) – Initial limits for the y-axis. Can be changed if the plot is saved as an HTML object. Only applies if orientation is veritcal.
- **tick_size** (Optional[float], default: None) – Size of the font of the axis tick labels.
- **axis_label_size** (Optional[float], default: None) – Size of the font of the axis label.
- **widget** (bool, default: False) – If True, returns a go.FigureWidget object instead of a go.Figure object.

- **row** (Optional[int], default: None) – If Figure has multiple subplots, specifies which row to use for the plot. Indexing starts at 1. Note that some formatting args (such as figsize) may still be applied to all subplots. Both row and col must be provided together.
- **col** (Optional[int], default: None) – If Figure has multiple subplots, specifies which col to use for the plot. Indexing starts at 1. Note that some formatting args (such as figsize) may still be applied to all subplots. Both row and col must be provided together.
- **kwargs** – Passed to go.Heatmap.

Return type
Union[Figure, FigureWidget]

Returns
Figure object.

Raises

- **AssertionError** – If array dtype is object.
- **AssertionError** – If array is not two-dimensional.
- **AssertionError** – If figsize is not a tuple of length two.

histogram_plot(*arrays*, *keys=[]*, *histfunc='count'*, *histnorm=''*, *normalizer=None*, *nbins=None*, *binsize=None*, *bargap=None*, *bargroupgap=None*, *cumulative=False*, *include_histogram=True*, *include_kde=False*, *bandwidth=None*, *extend_kde=0*, *fill_kde=False*, *colors=None*, *alpha=1.0*, *orientation='v'*, *barmode='group'*, *add_cell_numbers=True*, *legend=True*, *figure=None*, *figsize=(None, None)*, *margin='auto'*, *title=None*, *x_label=None*, *y_label=None*, *x_limit=None*, *y_limit=None*, *tick_size=None*, *axis_label_size=None*, *widget=False*, *row=None*, *col=None*, ***kwargs*)

Builds a Plotly Figure object plotting a histogram of each of the given arrays. Each array is interpreted as a separate condition and is plotted in a different color.

Parameters

- **arrays** (Collection[ndarray]) – List of arrays to plot histograms.
- **keys** (Collection[str], default: []) – Names corresponding to the data arrays. If not provided, the keys will be integers.
- **histfunc** (str, default: 'count') – Specifies the binning function used for this histogram trace. If "count", the histogram values are computed by counting the number of values lying inside each bin. Can also be "sum", "avg", "min", "max".
- **histnorm** (str, default: '') – Specifies the type of normalization used for this histogram trace. If "", the span of each bar corresponds to the number of occurrences If "percent" / "probability", the span of each bar corresponds to the percentage / fraction of occurrences with respect to the total number of sample points (here, the sum of all bin HEIGHTS equals 100% / 1). If "density", the span of each bar corresponds to the number of occurrences in a bin divided by the size of the bin interval. If "probability density", the area of each bar corresponds to the probability that an event will fall into the corresponding bin (here, the sum of all bin AREAS equals 1).
- **normalizer** (Union[Callable, str, None], default: None) – If given, used to normalize the data after applying the estimators. Normalizes the error as well. Can be 'minmax' or 'maxabs', or a callable that inputs an array and outputs an array of the same shape.
- **nbins** (Optional[int], default: None) – Approximate number of bins to use. Ignored if binsize is set.
- **binsize** (Optional[float], default: None) – Size of each bin.

- **bargap** (Optional[float], default: None) – Gap between bars in adjacent locations.
- **bargroupgap** (Optional[float], default: None) – Gap between bars in the same location.
- **cumulative** (bool, default: False) – If True, the histogram will plot cumulative occurances.
- **include_histogram** (bool, default: True) – If True, plot the histogram as a series of bars.
- **include_kde** (bool, default: False) – If True, calculate and plot a Gaussian kernel density estimate of the data. NOTE: Not currently normalized, so only plots the probability density function.
- **bandwidth** (Union[str, float, None], default: None) – Value of the bandwidth or name of method to estimate a good value of the bandwidth. Method options are 'scott' and 'silverman'. If None, uses 'scott'.
- **extend_kde** (Union[float, bool], default: 0) – Boolean or number of bandwidth lengths to extend the plot of the kernel density estimate. False or value of 0 means the kernel density estimate will only be plotted for the values of the data provided. If True, the default value is 3.
- **fill_kde** (Union[bool, str], default: False) – If True, fills in the area of the kernel density estimate. By default, fills to a value of 0 on the axis. Can also be a string to specify a different fill method. These options are from Plotly and include: 'tozerox', 'tozeroy', 'tonextx', 'tonexty', 'tonext', 'toself'.
- **colors** (Union[str, Collection[str], None], default: None) – Name of a color palette or map to use. Searches first in seaborn/matplotlib, then in Plotly to find the color map. If not provided, the color map will be glasbey. Can also be list of named CSS colors or hexadecimal or RGBA strings.
- **alpha** (float, default: 1.0) – Opacity of the fill color of the histogram as a float in the range [0, 1].
- **orientation** (str, default: 'v') – Orientation of the bar plot.
- **barmode** (str, default: 'group') – Keyword argument describing how to group the bars. Options are 'group', 'overlay', 'stack', and 'relative'.
- **add_cell_numbers** (bool, default: True) – If True, adds the number of cells to each key in keys.
- **legend** (bool, default: True) – If False, no legend is made on the plot.
- **figure** (Union[Figure, FigureWidget, None], default: None) – If a go.Figure object is given, will be used to make the plot instead of a blank figure.
- **figsize** (Tuple[int], default: (None, None)) – Height and width of the plot in pixels. Must be a tuple of length two. To leave height or width unchanged, set as None.
- **margin** (str, default: 'auto') – Set the size of the margins. If 'auto', all margins are set to defualt values. If 'zero' or 'tight', margins are removed. If a number is given, sets all margins to that number.
- **title** (Optional[str], default: None) – Title to add to the plot
- **x_label** (Optional[str], default: None) – Label of the x-axis
- **y_label** (Optional[str], default: None) – Label of the y-axis

- **x_limit** (Optional[Tuple[float]], default: None) – Initial limits for the x-axis. Can be changed if the plot is saved as an HTML object. Only applies if orientation is horizontal.
- **y_limit** (Optional[Tuple[float]], default: None) – Initial limits for the y-axis. Can be changed if the plot is saved as an HTML object. Only applies if orientation is vertical.
- **tick_size** (Optional[float], default: None) – Size of the font of the axis tick labels.
- **axis_label_size** (Optional[float], default: None) – Size of the font of the axis label.
- **widget** (bool, default: False) – If True, returns a go.FigureWidget object instead of a go.Figure object.
- **row** (Optional[int], default: None) – If Figure has multiple subplots, specifies which row to use for the plot. Indexing starts at 1. Note that some formatting args (such as figsize) may still be applied to all subplots. Both row and col must be provided together.
- **col** (Optional[int], default: None) – If Figure has multiple subplots, specifies which col to use for the plot. Indexing starts at 1. Note that some formatting args (such as figsize) may still be applied to all subplots. Both row and col must be provided together.
- **kwargs** – Depending on name, kwargs are passed to either go.Histogram or the line kwarg of go.Scatter if a kernel density estimate is included. The following kwargs are passed to "line": 'color', 'dash', 'shape', 'simplify', 'smoothing', 'width', 'hoverinfo'.

Return type
Union[Figure, FigureWidget]

Returns
Figure object

Raises

- **AssertionError** – If not all items in arrays are np.ndarray.
- **AssertionError** – If orientation is a disallowed value.
- **AssertionError** – If figsize is not a tuple of length two.

line_plot(*arrays*, *keys=[]*, *estimator=None*, *err_estimator=None*, *normalizer=None*, *colors=None*, *alpha=1.0*, *time=None*, *add_cell_numbers=True*, *legend=True*, *figure=None*, *figsize=(None, None)*, *margin='auto'*, *title=None*, *x_label=None*, *y_label=None*, *x_limit=None*, *y_limit=None*, *tick_size=None*, *axis_label_size=None*, *widget=False*, *gl=False*, *row=None*, *col=None*, ***kwargs*)

Builds a Plotly Figure object plotting lines of the given arrays. Each array is interpreted as a separate condition and is plotted in a different color.

Parameters

- **arrays** (Collection[ndarray]) – List of arrays to plot. Assumed structure is n_cells x n_features. Arrays must be two-dimensional, so if only one sample, use np.newaxis or np.expand_dims.
- **keys** (Collection[str], default: []) – Names corresponding to the data arrays. If not provided, the keys will be integers.
- **estimator** (Union[Callable, str, partial, None], default: None) – Function for aggregating observations from multiple cells. For example, if estimator is np.mean, the mean of all of the cells will be plotted instead of a trace for each cell. Can be a function, name of numpy function, name of function in estimator_utils, or a functools.partial object. If a function or functools.partial object, should input a 2D array and return a 1D array.

- **err_estimator** (Union[Callable, str, partial, None], default: None) – Function for estimating error bars from multiple cells. Can be a function, name of numpy function, name of function in estimator_utils, or a functools.partial object. If a function or functools.partial object, should input a 2D array and return a 1D or 2D array. If output is 1D, errors will be symmetric If output is 2D, the first dimension is used for the high error and second dimension is used for the low error.
- **normalizer** (Union[Callable, str, None], default: None) – If given, used to normalize the data after applying the estimators. Normalizes the error as well. Can be 'minmax', 'maxabs', 'standard', 'robust', or a callable that inputs an array and outputs an array of the same shape.
- **colors** (Union[str, Collection[str], None], default: None) – Name of a color palette or map to use. Searches first in seaborn/matplotlib, then in Plotly to find the color map. If not provided, the color map will be glasbey.
- **alpha** (float, default: 1.0) – Opacity of the line colors.
- **time** (Union[Collection[ndarray], ndarray, None], default: None) – Time axis for the plot. Must be the same size as the second dimension of arrays.
- **add_cell_numbers** (bool, default: True) – If True, adds the number of cells to each key in keys.
- **legend** (bool, default: True) – If False, no legend is made on the plot.
- **figure** (Union[Figure, FigureWidget, None], default: None) – If a go.Figure object is given, will be used to make the plot instead of a blank figure.
- **figsize** (Tuple[int], default: (None, None)) – Height and width of the plot in pixels. Must be a tuple of length two. To leave height or width unchanged, set as None.
- **margin** (str, default: 'auto') – Set the size of the margins. If 'auto', all margins are set to defualt values. If 'zero' or 'tight', margins are removed. If a number is given, sets all margins to that number.
- **title** (Optional[str], default: None) – Title to add to the plot
- **x_label** (Optional[str], default: None) – Label of the x-axis
- **y_label** (Optional[str], default: None) – Label of the y-axis
- **x_limit** (Optional[Tuple[float]], default: None) – Initial limits for the x-axis. Can be changed if the plot is saved as an HTML object.
- **y_limit** (Optional[Tuple[float]], default: None) – Initial limits for the y-axis. Can be changed if the plot is saved as an HTML object.
- **tick_size** (Optional[float], default: None) – Size of the font of the axis tick labels.
- **axis_label_size** (Optional[float], default: None) – Size of the font of the axis label.
- **widget** (bool, default: False) – If True, returns a go.FigureWidget object instead of a go.Figure object.
- **gl** (bool, default: False) – If True, switches to using a WebGL backend. Much faster for large datasets, but some features may not be available. May not work in all contexts.
- **row** (Optional[int], default: None) – If Figure has multiple subplots, specifies which row to use for the plot. Indexing starts at 1. Note that some formatting args (such as figsize) may still be applied to all subplots. Both row and col must be provided together.

- **col** (Optional[int], default: None) – If Figure has multiple subplots, specifies which col to use for the plot. Indexing starts at 1. Note that some formatting args (such as figsize) may still be applied to all subplots. Both row and col must be provided together.
- **kwargs** – Depending on name, passed to the "line" keyword argument of go.Scatter or as keyword arguments for go.Scatter. The following kwargs are passed to "line": 'color', 'dash', 'shape', 'simplify', 'smoothing', 'width', 'hoverinfo'

Return type
Union[Figure, FigureWidget]

Returns
Figure object

Raises

- **AssertionError** – If any item in arrays is not two dimensional.
- **AssertionError** – If figsize is not a tuple of length two.
- **TypeError** – If time is not an np.ndarray or collection of np.ndarray.

ridgeline_plot(*arrays*, *keys=[]*, *colors=None*, *spanmode='hard'*, *overlap=3*, *show_box=False*, *show_points=False*, *show_mean=True*, *add_cell_numbers=True*, *legend=True*, *figure=None*, *figsize=(None, None)*, *margin='auto'*, *title=None*, *x_label=None*, *y_label=None*, *x_limit=None*, *y_limit=None*, *tick_size=None*, *axis_label_size=None*, *widget=False*, *row=None*, *col=None*, ***kwargs*)

Builds a Plotly go.Figure object with partially overlapping distributions for each of the arrays given. Similar to a violin plot. See the section on ridgeline plots here for more information.

Parameters

- **arrays** (Collection[ndarray]) – List of arrays to create distributions from. Arrays are assumed to be one dimensional.
- **keys** (Collection[str], default: []) – Names corresponding to the data arrays. If not provided, the keys will be integers.
- **colors** (Union[str, Collection[str], None], default: None) – Name of a color palette or map to use. Searches first in seaborn/matplotlib, then in Plotly to find the color map. If not provided, the color map will be glasbey. Can also be list of named CSS colors or hexadecimal or RGBA strings.
- **spanmode** (str, default: 'hard') – Determines how far the tails of the violin plot are extended. If 'hard', the plot spans as far as the data. If 'soft', the tails are extended.
- **overlap** (float, default: 3) – Sets the amount of overlap between adjacent distributions. Larger values means more overlap.
- **show_box** (bool, default: False) – If True, a box plot is made and overlaid over the distribution.
- **show_points** (Union[str, bool], default: False) – If True, individual data points are overlaid over the distribution.
- **side** – Side to plot the distribution. By default, the distribution is plotted on both sides, but can be 'positive' or 'negative' to plot on only one side.
- **add_cell_numbers** (bool, default: True) – If True, adds the number of cells to each key in keys.
- **legend** (bool, default: True) – If False, no legend is made on the plot.

- **figure** (Union[Figure, FigureWidget, None], default: None) – If a go.Figure object is given, will be used to make the plot instead of a blank figure.
- **figsize** (Tuple[int], default: (None, None)) – Height and width of the plot in pixels. Must be a tuple of length two. To leave height or width unchanged, set as None.
- **margin** (str, default: 'auto') – Set the size of the margins. If 'auto', all margins are set to defualt values. If 'zero' or 'tight', margins are removed. If a number is given, sets all margins to that number.
- **title** (Optional[str], default: None) – Title to add to the plot
- **x_label** (Optional[str], default: None) – Label of the x-axis
- **y_label** (Optional[str], default: None) – Label of the y-axis
- **x_limit** (Optional[Tuple[float]], default: None) – Initial limits for the x-axis. Can be changed if the plot is saved as an HTML object. Only applies if orientation is horizontal.
- **y_limit** (Optional[Tuple[float]], default: None) – Initial limits for the y-axis. Can be changed if the plot is saved as an HTML object. Only applies if orientation is veritcal.
- **tick_size** (Optional[float], default: None) – Size of the font of the axis tick labels.
- **axis_label_size** (Optional[float], default: None) – Size of the font of the axis label.
- **widget** (bool, default: False) – If True, returns a go.FigureWidget object instead of a go.Figure object.
- **row** (Optional[int], default: None) – If Figure has multiple subplots, specifies which row to use for the plot. Indexing starts at 1. Note that some formatting args (such as figsize) may still be applied to all subplots. Both row and col must be provided together.
- **col** (Optional[int], default: None) – If Figure has multiple subplots, specifies which col to use for the plot. Indexing starts at 1. Note that some formatting args (such as figsize) may still be applied to all subplots. Both row and col must be provided together.
- **kwargs** – Depending on name, passed to go.Violin or to go.Figure.update_traces(). The following kwargs are passed to go.Violin: 'bandwidth', 'fillcolor', 'hoverinfo', 'jitter', 'line', 'marker', 'opacity', 'pointpos', 'points', 'span', 'width', 'hoverinfo'
- **show_mean** (bool, default: True) –

Return type
Union[Figure, FigureWidget]

Returns
Figure object

scatter_plot(*x_arrays=[]*, *y_arrays=[]*, *keys=[]*, *estimator=None*, *err_estimator=None*, *normalizer=None*, *scatter_mode='markers'*, *colors=None*, *alpha=1.0*, *symbols=None*, *add_cell_numbers=True*, *legend=True*, *figure=None*, *figsize=(None, None)*, *margin='auto'*, *title=None*, *x_label=None*, *y_label=None*, *x_limit=None*, *y_limit=None*, *tick_size=None*, *axis_label_size=None*, *widget=False*, *gl=False*, *row=None*, *col=None*, ***kwargs*)

Builds a Plotly Figure object containing a scatter plot of the given arrays. Each array is interpreted as a separate condition and is plotted in a different color or with a different marker.

Parameters

- **x_arrays** (Collection[ndarray], default: []) – List of arrays that set the x-coordinates to plot. Each array is assumed to be a different condition.
- **y_arrays** (Collection[ndarray], default: []) – List of arrays that set the y-coordinates to plot. Each array is assumed to be a different condition.

- **keys** (Collection[str], default: []) – Names corresponding to the data arrays. If not provided, the keys will be integers.
- **estimator** (Union[Callable, str, partial, None], default: None) – Function for aggregating observations from multiple cells. For example, if estimator is np.mean, the mean of all of the cells will be plotted instead of a point for each cell. Can be a function, name of numpy function, name of function in estimator_utils, or a functools.partial object. If a function or functools.partial object, should input a 2D array and return a 1D array.
- **err_estimator** (Union[Callable, str, partial, None], default: None) – Function for estimating vertical error bars. Can be a function, name of numpy function, name of function in estimator_utils, or a functools.partial object. If a function or functools.partial object, should input a 2D array and return a 1D or 2D array. If output is 1D, errors will be symmetric. If output is 2D, the first dimension is used for the high error and second dimension is used for the low error. Only applies to the y-dimension. Horizontal error bars not currrently implemented.
- **normalizer** (Union[Callable, str, None], default: None) – If given, used to normalize the data after applying the estimators. Normalizes the error as well. Can be 'minmax' or 'maxabs', or a callable that inputs an array and outputs an array of the same shape.
- **scatter_mode** (str, default: 'markers') – Drawing mode for the traces. Can be 'markers', 'lines', or 'lines+markers'.
- **colors** (Union[str, Collection[str], None], default: None) – Name of a color palette or map to use. Searches first in seaborn/matplotlib, then in Plotly to find the color map. If not provided, the color map will be glasbey. Can also be list of named CSS colors or hexadecimal or RGBA strings.
- **alpha** (float, default: 1.0) – Opacity of the marker fill colors.
- **add_cell_numbers** (bool, default: True) – If True, adds the number of cells to each key in keys.
- **legend** (bool, default: True) – If False, no legend is made on the plot.
- **figure** (Union[Figure, FigureWidget, None], default: None) – If a go.Figure object is given, will be used to make the plot instead of a blank figure.
- **figsize** (Tuple[int], default: (None, None)) – Height and width of the plot in pixels. Must be a tuple of length two. To leave height or width unchanged, set as None.
- **margin** (str, default: 'auto') – Set the size of the margins. If 'auto', all margins are set to defualt values. If 'zero' or 'tight', margins are removed. If a number is given, sets all margins to that number.
- **title** (Optional[str], default: None) – Title to add to the plot
- **x_label** (Optional[str], default: None) – Label of the x-axis
- **y_label** (Optional[str], default: None) – Label of the y-axis
- **x_limit** (Optional[Tuple[float]], default: None) – Initial limits for the x-axis. Can be changed if the plot is saved as an HTML object.
- **y_limit** (Optional[Tuple[float]], default: None) – Initial limits for the y-axis. Can be changed if the plot is saved as an HTML object.
- **tick_size** (Optional[float], default: None) – Size of the font of the axis tick labels.
- **axis_label_size** (Optional[float], default: None) – Size of the font of the axis label.

- **widget** (bool, default: False) – If True, returns a go.FigureWidget object instead of a go.Figure object.
- **gl** (bool, default: False) – If True, switches to using a WebGL backend. Much faster for large datasets, but some features may not be available. May not work in all contexts.
- **row** (Optional[int], default: None) – If Figure has multiple subplots, specifies which row to use for the plot. Indexing starts at 1. Note that some formatting args (such as figsize) may still be applied to all subplots. Both row and col must be provided together.
- **col** (Optional[int], default: None) – If Figure has multiple subplots, specifies which col to use for the plot. Indexing starts at 1. Note that some formatting args (such as figsize) may still be applied to all subplots. Both row and col must be provided together.
- **kwargs** – Depending on name, passed to the "marker" keyword argument of go.Scatter or as keyword arguments for go.Scatter. The following kwargs are passed to "marker": 'color', 'line', 'opacity', 'size', 'symbol'.

Return type
Union[Figure, FigureWidget]

Returns
Figure object

Raises

- **AssertionError** – If both x_arrays and y_arrays are given, but have different lengths.
- **AssertionError** – If not all items in either array are np.ndarray.
- **AssertionError** – If not all items in arrays have the same number of columns.
- **AssertionError** – If any item in arrays has more than 3 columns.
- **AssertionError** – If figsize is not a tuple of length two.

Parameters
symbols (Union[str, Collection[str], None], default: None) –

trace_color_plot(*trace_array*, *color_arrays=[]*, *color_thres=[]*, *colors=None*, *rows=6*, *cols=4*, *max_figures=None*, *time=None*, *figsize=(None, None)*, *title=None*, *x_label=None*, *y_label=None*, *x_limit=None*, *y_limit=None*, ***kwargs*)

Generates Plotly go.Figure objects with subplots for each individual trace in trace_array. Traces can be colored with discrete colors based on arbitrary criteria in color_arrays. For example, this function can be used to evaluate the success of peak segmentation by passing the traces to trace_array, and the peak segmentation to color_arrays.

Parameters

- **trace_array** (ndarray) – Array containing the values to be plotted. Assumed structure is two-dimensional of shape n_cells x n_features.
- **color_arrays** (Collection[ndarray], default: []) – Collection of arrays of the same shape as trace_array. Used with color_thres to determine what sections of the trace should be colored. There is no limit on the number of arrays passed, but color_thres must be the same length.
- **color_thres** (Collection[float], default: []) – Collection of thresholds associated with color_arrays. For each array and threshold, the trace will be colored wherever color_array >= color_thres.

- **colors** (Union[str, Collection[str], None], default: None) – Name of a color palette or map to use. Will use the first color as the base color of trace, and subsequent colors for each color_array. Searches first in seaborn/matplotlib, then in Plotly to find the color map. If not provided, the color map will be glasbey. Can also be list of named CSS colors or hexadecimal or RGBA strings.
- **rows** (int, default: 6) – Number of rows of subplots to make for each figure.
- **cols** (int, default: 4) – Number of columns of subplots to make for each figure.
- **max_figures** (Optional[int], default: None) – Maximum number of figures to produce. If None, makes enough figures to plot all of the traces.
- **time** (Optional[ndarray], default: None) – Time axis for the plot. Must be the same size as the second dimension of arrays.
- **title** (Optional[str], default: None) – Title to add to the plot
- **x_label** (Optional[str], default: None) – Label of the x-axis
- **y_label** (Optional[str], default: None) – Label of the y-axis
- **x_limit** (Optional[Tuple[float]], default: None) – Initial limits for the x-axis. Can be changed if the plot is saved as an HTML object.
- **y_limit** (Optional[Tuple[float]], default: None) – Initial limits for the y-axis. Can be changed if the plot is saved as an HTML object.
- **kwargs** – Depending on name, passed to the "line" keyword argument of go.Scatter or as keyword arguments for go.Scatter. The following kwargs are passed to "line": 'color', 'dash', 'shape', 'simplify', 'smoothing', 'width', 'hoverinfo'
- **figsize** (Tuple[float], default: (None, None)) –

Return type
Generator

Returns
Generator that produces go.Figure objects

Raises

- **AssertionError** – If any array in color_arrays does not have the same shape as trace array.
- **AssertionError** – If length of color_arrays does not equal length of color_thres.

violin_plot(*arrays*, *neg_arrays=[]*, *keys=[]*, *neg_keys=[]*, *colors=None*, *neg_colors=None*, *alpha=1.0*, *orientation='v'*, *show_box=False*, *show_points=False*, *show_mean=False*, *spanmode='soft'*, *side=None*, *add_cell_numbers=True*, *legend=True*, *figure=None*, *figsize=(None, None)*, *margin='auto'*, *title=None*, *x_label=None*, *y_label=None*, *x_limit=None*, *y_limit=None*, *tick_size=None*, *axis_label_size=None*, *widget=False*, *row=None*, *col=None*, ***kwargs*)

Builds a Plotly go.Figure object with violin distributions for each of the arrays given. If negative arrays are given, matches them with arrays and plots two distributions side by side.

Parameters

- **arrays** (Collection[ndarray]) – List of arrays to plot. Arrays are assumed to be one dimensional. If neg_arrays is given, arrays are plotted on the positive side.
- **keys** (Collection[str], default: []) – Names corresponding to the data arrays. If not provided, the keys will be integers.

- **neg_arrays** (Collection[ndarray], default: []) – List of arrays to plot. Arrays are assumed to be 1-dimensional. If neg_arrays is given, arrays are plotted on the positive side.
- **neg_keys** (Collection[str], default: []) – Names corresponding to the neative data arrays. If not provided, the keys will be integers.
- **colors** (Union[str, Collection[str], None], default: None) – Name of a color palette or map to use. Searches first in seaborn/matplotlib, then in Plotly to find the color map. If not provided, the color map will be glasbey. Can also be list of named CSS colors or hexadecimal or RGBA strings.
- **neg_colors** (Union[str, Collection[str], None], default: None) – Name of a color palette or map to use. Searches in seaborn/matplotlib first, then in Plotly to find the color map. If not provided, the color map will be glasbey. Can also be list of named CSS colors or hexadecimal or RGBA strings.
- **alpha** (float, default: 1.0) – Opacity of the fill color of the violin plots as a float in the range [0, 1].
- **orientation** (str, default: 'v') – Orientation of the violin plot.
- **show_box** (bool, default: False) – If True, a box plot is made and overlaid over the violin plot.
- **show_points** (Union[str, bool], default: False) – If True, individual data points are overlaid over the violin plot.
- **show_mean** (bool, default: False) – If True, dashed line is plotted at the mean value.
- **spanmode** (str, default: 'soft') – Determines how far the tails of the violin plot are extended. If 'hard', the plot spans as far as the data. If 'soft', the tails are extended.
- **side** (Optional[str], default: None) – Side to plot the distribution. By default, the distribution is plotted on both sides, but can be 'positive' or 'negative' to plot on only one side.
- **add_cell_numbers** (bool, default: True) – If True, adds the number of cells to each key in keys.
- **legend** (bool, default: True) – If False, no legend is made on the plot.
- **figure** (Union[Figure, FigureWidget, None], default: None) – If a go.Figure object is given, will be used to make the plot instead of a blank figure.
- **figsize** (Tuple[int], default: (None, None)) – Height and width of the plot in pixels. Must be a tuple of length two. To leave height or width unchanged, set as None.
- **margin** (str, default: 'auto') – Set the size of the margins. If 'auto', all margins are set to defualt values. If 'zero' or 'tight', margins are removed. If a number is given, sets all margins to that number.
- **title** (Optional[str], default: None) – Title to add to the plot
- **x_label** (Optional[str], default: None) – Label of the x-axis
- **y_label** (Optional[str], default: None) – Label of the y-axis
- **x_limit** (Optional[Tuple[float]], default: None) – Initial limits for the x-axis. Can be changed if the plot is saved as an HTML object. Only applies if orientation is horizontal.
- **y_limit** (Optional[Tuple[float]], default: None) – Initial limits for the y-axis. Can be changed if the plot is saved as an HTML object. Only applies if orientation is veritcal.

- **tick_size** (Optional[float], default: None) – Size of the font of the axis tick labels.
- **axis_label_size** (Optional[float], default: None) – Size of the font of the axis label.
- **widget** (bool, default: False) – If True, returns a go.FigureWidget object instead of a go.Figure object.
- **row** (Optional[int], default: None) – If Figure has multiple subplots, specifies which row to use for the plot. Indexing starts at 1. Note that some formatting args (such as figsize) may still be applied to all subplots. Both row and col must be provided together.
- **col** (Optional[int], default: None) – If Figure has multiple subplots, specifies which col to use for the plot. Indexing starts at 1. Note that some formatting args (such as figsize) may still be applied to all subplots. Both row and col must be provided together.
- **kwargs** – Depending on name, passed to go.Violin or to go.Figure.update_traces(). The following kwargs are passed to go.Violin: 'bandwidth', 'fillcolor', 'hoverinfo', 'jitter', 'line', 'marker', 'opacity', 'pointpos', 'points', 'span', 'width', 'hoverinfo'

Return type
Union[Figure, FigureWidget]

Returns
Figure object

Raises

- **AssertionError** – If not all entries in arrays or neg_arrays are np.ndarray
- **AssertionError** – If any entry in arrays or neg_arrays have more than one dimension.
- **AssertionError** – If neg_arrays is given, and len(arrays) != len(neg_arrays)
- **AssertionError** – If figsize is not a tuple of length two.

2.6 Utilities

2.6.1 General utilities

class celltk.utils.operation_utils.**PadHelper**(*target*, *axis=None*, *mode='constant'*, ***kwargs*)

Pads images to be even for wavelet reconstruction. In the future, may include more generalizeable padding.

celltk.utils.operation_utils.**cast_sitk**(*image*, *req_type*, *cast_up=False*)

Casts a SimpleITK image to a different pixel type.

Parameters

- **req_type** (str) – Desired SimpleITK pixel type.
- **cast_up** (bool, default: False) – If True, pixels can be cast to a type with higher precision (i.e. sitk.sitkInt16 -> sitk.sitkInt32)
- **image** (Image) –

Return type
Image

celltk.utils.operation_utils.**crop_array**(*array*, *crop_vals=None*)

Crops an image to the specified dimensions.

Parameters

- **array** (ndarray) –
- **crop_vals** (Optional[Tuple[int]], default: None) –

Return type
ndarray

celltk.utils.operation_utils.**data_from_regionprops_table**(*regionprops*, *metric*, *labels=None*, *frames=None*)

Given a list of regionprops data, return data for specified metrics at specific label, frame indices

Parameters

- **regionprops** (Dict[int, dict]) –
- **metric** (str) –
- **labels** (Optional[List[int]], default: None) –
- **frames** (Optional[List[int]], default: None) –

Return type
Union[ndarray, float]

celltk.utils.operation_utils.**dilate_sitk**(*labels*, *radius*)

Grayscale dilation of images.

Parameters

- **labels** (ndarray) –
- **radius** (int) –

Return type
ndarray

celltk.utils.operation_utils.**get_binary_footprint**(*rank=2*, *connectivity=1*)

Wrapper for ndi.generate_binary_structure

Parameters

- **rank** (int, default: 2) –
- **connectivity** (int, default: 1) –

Return type
ndarray

celltk.utils.operation_utils.**get_cell_index**(*cell_id*, *label_array*, *position_id=None*, *position_array=None*)

Returns the index of the centroid of a specific cell.

Parameters

- **cell_id** (int) –
- **label_array** (ndarray) –
- **position_id** (Optional[int], default: None) –
- **position_array** (Optional[ndarray], default: None) –

Return type
int

celltk.utils.operation_utils.**get_image_pixel_type**(*image*)

Returns the numpy data type or SimpleITK pixel type of the input image as a string.

Parameters
image (Union[ndarray, Image]) –

Return type
str

celltk.utils.operation_utils.**get_split_idxs**(*arrays*, *axis=0*)

Parameters

- **arrays** (Collection[ndarray]) –
- **axis** (int, default: 0) –

Return type
List[int]

celltk.utils.operation_utils.**gray_fill_holes**(*labels*)

Fills holes in a non-binary image.

Parameters
labels (ndarray) –

Return type
ndarray

celltk.utils.operation_utils.**label_by_parent**(*mask*, *lineage*)

Replaces daughter cell labels with their parent label.

Parameters

- **mask** (Mask) –
- **lineage** (ndarray) –

Return type
Mask

celltk.utils.operation_utils.**lineage_to_track**(*mask*, *lineage*)

Given a Mask and a cell lineage, marks pixels in the Mask to create the representative Mask.

Parameters

- **mask** (Mask) –
- **lineage** (ndarray) –

Return type
Mask

celltk.utils.operation_utils.**mask_to_seeds**(*mask*, *method='sitk'*, *output='mask'*, *binary=True*)

Find centroid of all objects and return, either as list of points or labeled mask. If binary, all seeds are 1, otherwise, preserves labels.

Parameters

- **mask** (ndarray) –
- **method** (str, default: 'sitk') –
- **output** (str, default: 'mask') –
- **binary** (bool, default: True) –

Return type
Union[ndarray, list]

celltk.utils.operation_utils.**match_labels_linear**(*source*, *dest*)

Transfers labels from source to dest based on area overlap between objects

Parameters

- **source** (ndarray) –
- **dest** (ndarray) –

Return type
ndarray

celltk.utils.operation_utils.**nan_helper_2d**(*arr*)

Linear interpolation of nans along rows in 2D array.

Parameters
arr (ndarray) –

Return type
ndarray

celltk.utils.operation_utils.**ndi_binary_fill_holes**(*labels*, *fill_border=True*, *kernel_radius=2*, *max_length=45*, *in_place=False*)

Fills holes in a binary image. If fill border is true, uses an iterative heuristic strategy to determine if border holes belong to an object and fills them if they do.

Parameters

- **labels** (ndarray) –
- **fill_border** (bool, default: True) –
- **kernel_radius** (int, default: 2) –
- **max_length** (int, default: 45) –
- **in_place** (bool, default: False) –

Return type
ndarray

celltk.utils.operation_utils.**paired_dot_distance**(*par_xy*, *dau_xy*)

Calculates error for normalized dot distance and error along line. Used by Tracker.detect_cell_division to evaluate possible parent, daughter combinations based on the final position of the cells.

Note:

- x, y are switched in this function relative to the image.

Parameters

- **par_xy** (ndarray) –
- **dau_xy** (ndarray) –

Return type
Tuple[ndarray]

celltk.utils.operation_utils.**parents_from_track**(*track*)

Returns dictionary of {daughter_id: parent_id} from the input Track.

Parameters
track (Mask) –

Return type
Dict[int, int]

celltk.utils.operation_utils.**shift_array**(*array*, *shift*, *fill=nan*)

Shifts an array and fills in the values or crops to size. See: https://stackoverflow.com/questions/30399534/

Parameters

- **array** (ndarray) –
- **shift** (tuple) –
- **fill** (float, default: nan) –

Return type
ndarray

celltk.utils.operation_utils.**sitk_binary_fill_holes**(*labels*, *fill_border=True*, *iterations=False*, *kernel_radius=4*, *max_length=45*, *in_place=True*, ***kwargs*)

Fills holes in a binary image. If fill border is true, uses an iterative heuristic strategy to determine if border holes belong to an object and fills them if they do.

Parameters

- **labels** (ndarray) –
- **fill_border** (bool, default: True) –
- **iterations** (Union[int, bool], default: False) –
- **kernel_radius** (int, default: 4) –
- **max_length** (int, default: 45) –
- **in_place** (bool, default: True) –

Return type
ndarray

celltk.utils.operation_utils.**skimage_level_set**(*shape*, *levelset='checkerboard'*, *size=None*, *center=None*)

Wrapper for levelset functions in skimage.segmentation

Params size
Refers to square_size for checkerboard or radius for disk

celltk.utils.operation_utils.**sliding_window_generator**(*arr*, *overlap=0*)

Creates a generator for slices from array with an arbitrary amount of overlap for successive slices.

Parameters
overlap (int, default: 0) – Number of slices to overlap in each returned array. e.g. overlap = 1: [0, 1], [1, 2], [2, 3], [3, 4], overlap = 2: [0, 1, 2], [1, 2, 3], [2, 3, 4]

Note:

- Overlaps get passed as a stack, not as separate args. i.e. if overlap = 1, image.shape = (2, h, w)

Note:

- If memory is an issue here, can probably manually count the indices and make a generator that way, but it will be much slower.

Parameters
arr (ndarray) –

Return type
Generator

celltk.utils.operation_utils.**split_array**(*array*, *split_idxs*, *axis=0*)

Parameters

- **array** (ndarray) –
- **split_idxs** (List[int]) –
- **axis** (int, default: 0) –

Return type
List[ndarray]

celltk.utils.operation_utils.**stack_pad**(*arrays*, *axis=0*, *pad_value=nan*)

Stacks arrays along axis, padding shorter arrays with pad_value.

Parameters

- **arrays** (Collection[ndarray]) –
- **axis** (int, default: 0) –
- **pad_value** (float, default: nan) –

Return type
ndarray

celltk.utils.operation_utils.**track_to_lineage**(*track*)

Given a set of track images, reconstruct all the cell lineages.

Parameters
track (Mask) –

Return type
ndarray

celltk.utils.operation_utils.**track_to_mask**(*track*, *idx=None*)

Gives Track with parent values filled in by closest neighbor.

Parameters

- **track** (Mask) –
- **idx** (Optional[ndarray], default: None) – locations of parent values to fill in

Return type
Mask

celltk.utils.operation_utils.**voronoi_boundaries**(*seed*, *thin=False*, *thick=False*)

Calculate voronoi boundaries for the given seed array and return as binary mask.

Parameters

- **seed** (ndarray) –
- **thin** (bool, default: False) –
- **thick** (bool, default: False) –

Return type
ndarray

celltk.utils.operation_utils.**wavelet_background_estimate**(*image*, *wavelet='db4'*, *mode='smooth'*, *level=None*, *blur=False*, *axes=(-2, -1)*)

Uses wavelet reconstruction of the image to estimate the background.

Parameters

- **image** (ndarray) –
- **wavelet** (str, default: 'db4') –
- **mode** (str, default: 'smooth') –
- **level** (Optional[int], default: None) –
- **blur** (bool, default: False) –
- **axes** (Tuple[int], default: (-2, -1)) –

Return type
ndarray

celltk.utils.operation_utils.**wavelet_noise_estimate**(*image*, *noise_level=1*, *wavelet='db1'*, *mode='smooth'*, *level=None*, *thres=2*, *axes=(-2, -1)*)

Uses wavelet reconstruction of the image to estimate noise.

Parameters

- **image** (ndarray) –
- **noise_level** (int, default: 1) –
- **wavelet** (str, default: 'db1') –
- **mode** (str, default: 'smooth') –
- **level** (Optional[int], default: None) –
- **thres** (int, default: 2) –
- **axes** (Tuple[int], default: (-2, -1)) –

Return type
ndarray

2.6.2 Peak finding utilities

class celltk.utils.peak_utils.**PeakHelper**

Helper class for getting data from traces and peak labels

amplitude(*traces*, *labels*)

Returns the maximum value in each peak for each trace. Each cell has an empty list if no peaks are labeled.

Parameters

- **traces** (ndarray) – Array of shape n_cells x n_features with values to use for calculating amplitude.
- **labels** (ndarray) – Array of same shape as traces with peaks labeled with unique integers in each cell trace.

Return type
List[List[float]]

Returns
List of amplitudes for each cell trace.

area_under_curve(*traces*, *labels*)

Parameters

- **traces** (ndarray) –
- **labels** (ndarray) –

Return type
List[List[float]]

detect_peak_tracts(*traces*, *labels*, *max_gap=8*)

Connects peaks that are close together into a single tract.

Parameters

- **traces** (ndarray) –
- **labels** (ndarray) –
- **max_gap** (int, default: 8) –

Return type
ndarray

filter_peaks(*traces*, *labels*, *metrics*, *thresholds*, *kwargs=[{}]*)

Parameters

- **traces** (ndarray) –
- **labels** (ndarray) –
- **metrics** (Collection[str]) –
- **thresholds** (Collection[float]) –
- **kwargs** (Collection[dict], default: [{}]) –

Return type
ndarray

first_time(*traces*, *labels*)

Returns the first time point belonging to each peak

Parameters

- **traces** (ndarray) –
- **labels** (ndarray) –

Return type
List[List[float]]

last_time(*traces*, *labels*)

Returns the last time point belonging to each peak

Parameters

- **traces** (ndarray) –
- **labels** (ndarray) –

Return type
List[List[float]]

length(*traces*, *labels*)

Returns the length of each peak.

Parameters

- **traces** (ndarray) –
- **labels** (ndarray) –

Return type
List[List[int]]

Returns

max_time(*traces*, *labels*)

Returns the time that a peak reaches it's maximum amplitude.

Parameters

- **traces** (ndarray) –
- **labels** (ndarray) –

Return type
List[List[float]]

nonlinearity(*traces*, *labels*)

Returns inverse of the absolute value of the Pearson's correlation coefficient. Higher values mean the peak is less linear.

Parameters

- **traces** (ndarray) –
- **labels** (ndarray) –

Return type
List[List[float]]

Returns

prominence(*traces*, *labels*, *tracts=[]*)

Returns the difference between the maximum value in a peak and the base of the peak. If tracts are provided adjusts the base of each peak in the tract to be the base of the tract.

Parameters

- **traces** (ndarray) –
- **labels** (ndarray) –
- **tracts** (List[List[int]], default: []) –

Return type
List[List[float]]

Returns

width(*traces*, *labels*, *tracts=[]*, *relative=0.5*, *absolute=None*)

Not yet complete.

Parameters

- **traces** (ndarray) –
- **labels** (ndarray) –
- **tracts** (List[List[int]], default: []) –
- **relative** (float, default: 0.5) –
- **absolute** (Optional[float], default: None) –

Return type
List[List[float]]

Returns

celltk.utils.peak_utils.**segment_peaks_agglomeration**(*traces*, *probabilities*, *steps=15*, *min_seed_prob=0.6*, *min_peak_prob=0.5*, *min_seed_length=2*, ***kwargs*)

Returns an array with peaks incrementally counted in each trace, i.e. the labels will be [0, 0, 1, 1,… 0, 2, 2, … 0, 3 ..].

Parameters

- **traces** (ndarray) –
- **probabilities** (ndarray) –
- **steps** (int, default: 15) –
- **min_seed_prob** (float, default: 0.6) –
- **min_peak_prob** (float, default: 0.5) –
- **min_seed_length** (int, default: 2) –

Return type
ndarray

2.6.3 Estimating error utilities

celltk.utils.estimator_utils.**bootstrap_estimator**(*arr*, *reps=1000*, *ci=0.95*, *axis=0*, *ignore_nans=True*, *function=<function nanmean>*)

Uses bootstrap resampling to estimate a confidence interval.

Parameters

- **arr** (ndarray) –
- **reps** (int, default: 1000) –
- **ci** (float, default: 0.95) –
- **axis** (int, default: 0) –
- **ignore_nans** (bool, default: True) –
- **function** (Callable, default: <function nanmean at 0x7fa5024f1e50>) –

Return type
Tuple[ndarray]

celltk.utils.estimator_utils.**confidence_interval**(*arr*, *ci=0.95*)

Calculates the confidence interval based on a t-distribution.

Note:

- Only works on axis 0 for now

Parameters

- **arr** (ndarray) –
- **ci** (float, default: 0.95) –

Return type
ndarray

celltk.utils.estimator_utils.**fraction_of_total**(*arr*, *ignore_nans=True*, *axis=0*)

Returns the fraction of entries that have non-zero values.

Parameters

- **arr** (ndarray) –
- **ignore_nans** (bool, default: True) –
- **axis** (int, default: 0) –

Return type
ndarray

celltk.utils.estimator_utils.**get_bootstrap_population**(*arr*, *boot_reps=1000*, *seed=69420*, *function=<function nanmean>*)

Parameters

- **arr** (ndarray) – response of cells in one condition, cells x response/times
- **boot_reps** (int, default: 1000) – Number of bootstrap replicates
- **seed** (int, default: 69420) –

- **function** (Callable, default: <function nanmean at 0x7fa5024f1e50>) –

Return type
ndarray

Returns
array boot_reps x response/times

celltk.utils.estimator_utils.**normal_approx**(*arr*, *ci=0.95*, *axis=0*)

Calculates the normal error for a binomial distribution.

Parameters

- **arr** (ndarray) –
- **ci** (float, default: 0.95) –
- **axis** (int, default: 0) –

Return type
ndarray

celltk.utils.estimator_utils.**wilson_score**(*arr*, *ci=0.95*, *axis=0*)

Calculates the Wilson score for a binomial distribution.

Parameters

- **arr** (ndarray) –
- **ci** (float, default: 0.95) –
- **axis** (int, default: 0) –

Return type
ndarray

2.6.4 Filtering utilities

celltk.utils.filter_utils.**any_nan**(*values*, *propagate=True*, *mask=None*)

Masks cells that include any np.nan

Parameters

- **values** (ndarray) –
- **propagate** (bool, default: True) –
- **mask** (Optional[ndarray], default: None) –

Return type
ndarray

celltk.utils.filter_utils.**any_negative**(*values*, *ignore_nans=False*, *propagate=True*, *mask=None*)

Masks cells that include any negative values

Parameters

- **values** (ndarray) –
- **ignore_nans** (bool, default: False) –
- **propagate** (bool, default: True) –
- **mask** (Optional[ndarray], default: None) –

Return type
ndarray

celltk.utils.filter_utils.**inside**(*values*, *lo=-inf*, *hi=inf*, *allow_equal=True*, *ignore_nans=False*, *propagate=True*, *mask=None*)

Masks cells with values that fall inside of the specified range

Parameters

- **values** (ndarray) –
- **lo** (float, default: -inf) –
- **hi** (float, default: inf) –
- **allow_equal** (bool, default: True) –
- **ignore_nans** (bool, default: False) –
- **propagate** (bool, default: True) –
- **mask** (Optional[ndarray], default: None) –

Return type
ndarray

celltk.utils.filter_utils.**inside_percentile**(*values*, *lo=0*, *hi=100*, *ignore_nans=False*, *propagate=True*, *mask=None*)

Masks cells with values that fall inside of the specified percentile range

Parameters

- **values** (ndarray) –
- **lo** (float, default: 0) –
- **hi** (float, default: 100) –
- **ignore_nans** (bool, default: False) –
- **propagate** (bool, default: True) –
- **mask** (Optional[ndarray], default: None) –

Return type
ndarray

celltk.utils.filter_utils.**outside**(*values*, *lo=-inf*, *hi=inf*, *allow_equal=True*, *ignore_nans=False*, *propagate=True*, *mask=None*)

Masks cells with values that fall outside of the specified range

Parameters

- **values** (ndarray) –
- **lo** (float, default: -inf) –
- **hi** (float, default: inf) –
- **allow_equal** (bool, default: True) –
- **ignore_nans** (bool, default: False) –
- **propagate** (bool, default: True) –
- **mask** (Optional[ndarray], default: None) –

Return type
ndarray

celltk.utils.filter_utils.**outside_percentile**(*values*, *lo=0*, *hi=100*, *ignore_nans=False*, *propagate=True*, *mask=None*)

Masks cells with values that fall outside of the specified percentile range

Parameters

- **values** (ndarray) –
- **lo** (float, default: 0) –
- **hi** (float, default: 100) –
- **ignore_nans** (bool, default: False) –
- **propagate** (bool, default: True) –
- **mask** (Optional[ndarray], default: None) –

Return type
ndarray

2.6.5 Metric utilities

celltk.utils.metric_utils.**intensity_stdev**(*mask*, *image*)

Returns standard deviation of all intensity values in region of interest.

Parameters

- **mask** (ndarray) –
- **image** (ndarray) –

Return type
float

celltk.utils.metric_utils.**intensity_variance**(*mask*, *image*)

Returns variance of all intensity values in region of interest.

Parameters

- **mask** (ndarray) –
- **image** (ndarray) –

Return type
float

celltk.utils.metric_utils.**median_intensity**(*mask*, *image*)

Returns median intensity in region of interest.

Parameters

- **mask** (ndarray) –
- **image** (ndarray) –

Return type
float

celltk.utils.metric_utils.**predict_peaks**(*array*, *weight_path='celltk/config/upeak_example_weights.tf'*, *segment=True*, *roi=(1, 2)*, *save_path=None*)

Predicts peaks in arbitrary data using UPeak.

Parameters

- **array** (ndarray) – Data of shape n_samples x n_timepoints. Each sample will have peaks predicted.
- **weight_path** (str, default: `'celltk/config/upeak_example_weights.tf'`) – Weights to use for predicting peaks. If not provided, uses default weights.
- **segment** (bool, default: `True`) – If True, segments peaks using a watershed based algorithm and returns segmennted peaks instead of peak probabilities.
- **roi** (Union[int, Tuple[int]], default: `(1, 2)`) – Regions of interest to return. 0 is background, 1 is slope, and 2 is plateau.
- **save_path** (Optional[str], default: `None`) – If provided, saves the output array at the supplied path.
- **save_plot** – If a string is provided, saves output figures using the given figure name. Do not provided extension in file name, figures can currently only be saved as png.
- **show_plot** – If True, plot will be shown to the user.

Return type
ndarray

celltk.utils.metric_utils.**total_intensity**(*mask*, *image*)

Returns total intensity in region of interest.

Parameters

- **mask** (ndarray) –
- **image** (ndarray) –

Return type
float

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

C

INDEX

Symbols

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W