coinc_lmfit module

This is a set of convenience functions written by Aaron Mueninghoff in July 2022 as part of his work on the Quantum Astrometry project, part of the Quantum Information Science research group at Brookhaven National Labs, working under Andrei Nomerotsky and Paul Stankus. Questions and comments can be sent to aaron.mueninghoff@stonybrook.edu.

These functions can be used to take a numpy array of channel pairs in the format produced by C++ code written by Denis Dolzhenko and extract information about rate of coincidence pairs.

`coinc_lmfit.`**`make_nice`**`(`*`arr`*`)`

Makes the array loaded in easier to work with.

This function takes channel pair data *arr* in the format provided by Denis, changes the dt (second) column to be in nanoseconds, changes the t (first) column to be in minutes, and changes the t column to start at zero.

Parameters: **arr** : *ndarray*

A $n \times 2$ numpy array in the the format of Denis' output, where the first column is the t value and the second column is dt value.

Returns: **arr** : *ndarray*

A $n \times 2$ numpy array of channel pairs in an easier to use format.

Notes

It is important that the array elements not be integers for the division.

Passing an array into `make_nice` a second time will make it not nice.

`coinc_lmfit.`**`create_barr`**`(`*`arrdt, lower=-20, upper=20, dtwidth=0.05, nbins=None`*`)`

Bins an array of dt values.

Given an array of dt values, this function separates the dt values in bins ranging from *lower* to *upper* of width *dtwidth* or across *nbins* number of bins. It is assumed that the dt quantities are in nanoseconds.

Parameters: **arrdt** : *array-like*

1d array-like object containing dt values to be binned.

Returns: **binmids** : *ndarray*

1d array of the dt values of the centers of the bins.

barrdt : *ndarray*

1d array of the bin values.

Other Parameters: **lower** : *float or int, optional*

The lower bound on the values to be binned (default is -20).

upper : *float or int, optional*

The upper bound on the values to be binned (default is 20).

dtwidth : *float, optional*

The width of the bins (default is 0.05). Ignored if *nbins* is given.

nbins : *int, optional*

The number of bins to bin *arrdt* into. An alternative to selecting the bin width using *dtwidth*.

`coinc_lmfit.`**`time_bin`**`(`*`arr, twidth`*`)`

Bins an array of t and dt values by the t values.

Takes a $n \times 2$ array and separates it into sections that are *twidth* wide (in seconds). Returns a 1d array of the center of the bins and a list of 1d arrays of dt values.

Parameters: **arr** : *ndarray*

A $n \times 2$ array with t values in the first column and dt values in the second column.

twidth : *int or float*

The width of the bins (in seconds).

Returns: **midlist** : *ndarray*

1d array containing the center values of the bins (in minutes).

arrdt_tlist : *list of ndarrays*

A list of arrays of the dt values in each bin.

`coinc_lmfit.`**`get_amp_unc`**`(`*`result, explicit_unc, i, maxiter, ci_verbose`*`)`

Tries to get the uncertainty on the amplitude of a peak fit.

Takes a ModelResult and tries to calculate or estimate the uncertainty on the best-fit value of the amplitude using `lmfit.conf_interval` or the covariance matrix, respectively.

Parameters: **result** : *LMFIT ModelResult*

A ModelResult from a fit to a peak-like model using the LMFIT package.

explicit_unc : *bool*

Whether to calculate the uncertainty explicitly (True, default) using `result.conf_interval()` or to estimate the uncertainty using the covariance matrix (False).

i : *int*

The index of the tbin that `coinc_sum` is at. Used for warning when explicit uncertainty calculation has reached *maxiter*.

maxiter : *int*

To be passed to `result.conf_interval`. The maximum number of iterations to do when calculating the uncertainty.

Returns: **amp_unc** : *float*

The uncertainty on the amplitude or np.nan if the function failed (see Notes).

Other Parameters: **ci_verbose** : *bool, optional*

Passed to `lmfit.conf_interval`. If *ci_verbose* is True (default is False) then `lmfit.conf_interval` will print out the result of every iteration in calculating the uncertainties.

Raises: UserWarning

Raises a UserWarning if the *maxiter* is reached. Does not raise UserWarning's from `result.conf_interval` about convergence, because they are handled.

Notes

Calculating the uncertainties explicitly takes much longer than estimating them.

Sometimes the fit cannot calculate the covariance matrix, and sometimes *conf_interval* cannot calculate the uncertainty. These usually happen when the amplitude is very small. If one of these methods fails, the function will try to get the uncertainty using the other method. If both fail, then *get_amp_unc* returns np.nan so that it can be fixed later by `fix_amp_unc`.

This function uses a context manager to temporarily change the way Warnings are filtered. This is so that warnings from `conf_interval` about convergence are ignored but warnings about hitting the maximum iteration are raised as exceptions. The maxiter warning is raised as an exception so it can be caught with a `try except` clause, modified, and then raised as a warning.

`coinc_lmfit.`**`fix_amp_unc`**`(`*`coinc_arr, unc_arr`*`)`

Estimates amplitude uncertainty where `get_amp_unc` couldn't.

Sometimes, `get_amp_unc` will fail, outputing `np.nan`. This function estimates the uncertainties where `get_amp_unc` could not, by taking the average of the uncertainties of the closest 10% of amplitudes by best-fit value.

Parameters: **coinc_arr** : *ndarray*

1d array of the best-fit amplitudes.

unc_arr : *ndarray*

1d array of the uncertainties on the best-fit ampltides, where some values may be `np.nan`.

Returns: **unc_arr** : *ndarray*

1d array of the uncertainties on the best-fit amplitudes containing no `np.nan`'s.

`coinc_lmfit.`**`coinc_sum`**`(`*`arr, sigma_width=1.5, tbinwidth=10, dtbinwidth_full=0.005, dtbinwidth=0.05, peakmodel=None, explicit_unc=True, maxiter=2000, ci_verbose=False`*`)`

Calculates the number of coincident pairs as function of time.

Takes an array in the format that `make_nice` outputs. Fits a single gaussian to all of the dt values, then separates the array into bins by time and fits a gaussian with the same width and center as the gaussian fit to the full array. Then calculates the area under each gaussian (without the background) and the uncertainty on the area.

The gaussian fitting function is $F(x; A, \mu, \sigma, c) = \frac{A}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2} + c$ where A, μ, σ, c are the amplitude, mean, sigma, and constant background, respectively.

Parameters: **arr** : *ndarray*

$n \times 2$ array with t values (in seconds) in the first column and dt values (in nanoseconds) in the second column.

sigma_width : *float, optional*

The distance in sigma units from the center of the gaussian to integrate over. That is $\int_{\mu-w_\sigma\sigma}^{\mu+w_\sigma\sigma} G(x)\,dx$ where μ, σ, w_σ, G are the center, the sigma, the *sigma_width*, and the gaussian function, respectively.

tbinwidth : *float, optional*

The width in time (seconds) in which to bin *arr*. A gaussian will be fit to the dt values in each of these bins.

Returns:

tbinmids : *ndarray*

1d array of the center values of the bins (in minutes).

coinc_arr : *ndarry*

1d array of the areas under the gaussian within *sigma_width* in each time bin.

unc_arr : *ndarray*

1d array of the uncertainties on the areas in *coinc_arr*.

Other Parameters:

dtbinwidth_full : *float, optional*

The width (in nanoseconds) of the bins for the dt values of the entirety of *arr*, to be used for the initial gaussian fit. Default is 0.005.

dtbinwidth : *float, optional*

The width (in nanoseconds) of the bins for the dt values of each time bin. This should be larger than *dtbinwidth_full* because there will be fewer values to bin. Default is 0.05.

peakmodel : *LMFIT peak-like built-in model, optional*

The peak-like model to use to fit the HBT peaks in the data (default is `GaussianModel`). Must be a model that has a `guess` function (all built-in models have this).

explicit_unc : *bool, optional*

Whether or not to calculate the uncertainies on the amplitude explicitly. Default is True.

maxiter : *int, optional*

Passed to `get_amp_unc` which passes it to `lmfit.conf_interval`. The maximum number of iterations (default is 2000) for `lmfit.conf_interval` to use when calculating explicitly. If a UserWarning is raised about reaching the maxiter, then the maxiter should be raised. Setting maxiter to `np.inf` is probably a bad idea.

ci_verbose : *bool, optional*

Passed to `get_amp_enc`, which passes it to `lmfit.conf_interval`. If *ci_verbose* is True, then `lmfit.conf_interval` will print out the result of every iteration in calculating the uncertainties.

Raises:

UserWarning

Calls `get_amp_unc` which may raise a UserWarning about `lmfit.conf_interval` reaching maxiter. If this happens, *maxiter* should be raised.

Notes

The expression for the area is derived here. Let w_{dt} be the width of the dt bins, and let w_σ be the width of the integration window in terms of sigma. Then the fitting equation is written as $f(x; A, \mu, \sigma, c) = \frac{A}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2} c$, where the parameters are defined as above.

$$\begin{aligned}
Area &= \int_{x=\mu-w_\sigma\sigma}^{x=\mu+w_\sigma\sigma} \frac{A}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2}\,dx \\
u = x-\mu, &\quad \Longrightarrow \quad du = dx \quad \Longrightarrow \quad dx = du \\
&= \frac{2A}{\sigma\sqrt{2\pi}} \int_{u=0}^{u=w_\sigma\sigma} e^{-u^2/2\sigma^2}\,du \\
v^2 = \frac{u^2}{2\sigma^2} &\quad \Longrightarrow \quad v = \frac{u}{\sqrt{2}\,\sigma} \quad \Longrightarrow \quad dv = \frac{du}{\sqrt{2}\,\sigma} \quad \Longrightarrow \quad du = \sqrt{2}\,\sigma dv \\
&= \frac{2A}{\sqrt{\pi}} \int_{v=0}^{v=w_\sigma/\sqrt{2}} e^{-v^2}\,dv \\
&= A \cdot \frac{2}{\sqrt{\pi}} \int_0^{w_\sigma/\sqrt{2}} e^{-v^2}\,dv \\
&= A \cdot \mathrm{erf}\left(\frac{w_\sigma}{\sqrt{2}}\right)
\end{aligned}$$

This value is then divided by $w_d t$ to get the total number of counts under the gaussian (unitless).

`coinc_lmfit.`**`make_slices`**`(`*`bounds_list, tbinmids`*`)`

Creates a list of slices using the centers of time bins.

Parameters: **bounds_list** : *array-like*

array or list of 2-tuples, where first entry in each tuple is the beginning of the region (in minutes) and the second entry is the end (in minutes).

tbinmids : *ndarray*

1d array of the center values of time bins (in minutes).

Returns: **slice_list** : *list*

List of slices in the same order as *bounds_list*.

`coinc_lmfit.`**`fit_sin`**`(`*`tbinmids, coinc_arr, unc_arr=None, init_vals=None, maxiter=200, ci_verbose=False, suppress_warnings=False`*`)`

Fits a sine function to the result of `coinc_sum`.

Takes the results of `coinc_sum` and fits it to $F(x; A, f, \phi, c) = A\sin(fx+\phi)+c$, where A, ϕ, f, c are the amplitude, frequency, phase shift, and background constant, respectively. Can explicitly calculate the uncertainties on the best-fit values.

Parameters: **tbinmids** : *ndarray*

1d array of the center values of time bins (in minutes).

coinc_arr : *ndarray*

1d array of the area under the gaussians fitted in each bin by `coinc_sum`.

unc_arr : *ndarray, optional*

1d array of the uncertainties on *coinc_arr*; to be used for weighting in the fit.

Returns: **res** : *LMFIT ModelResult*

A `lmfit.ModelResult` object for the fit

Other Parameters: **init_vals** : *array-like, optional*

An iterable with 4 float elements. Used as the initial values in the fit for the parameters in this order: amplitude, frequency, phase shift, constant.

Note that the `lmfit.guess` function for `lmfit.SineModel` seems to work very well. You probably dont need to specify the initial values.

maxiter : *int, optional*

To be passed to `lmfit.conf_interval`; the maximum number of iterations to be used. Default is 200.

ci_verbose : *bool, optional*

Passed to `lmfit.conf_interval`. If *ci_verbose* is True (default is False), then `lmfit.conf_interval` will print out the result of every iteration in calculating the uncertainties.

suppress_warnings : *bool, optional*

Whether to suppress warnings from `conf_interval`. Default is False.

Raises: UserWarning

`lmfit.conf_interval` may raise UserWarnings about bad convergence or reaching maxiter.

Notes

It is often easiest to give the first three arguments by unpacking (an array in the shape of) the output of `coinc_sum`.

`coinc_lmfit.`**`make_label`**`(`*`result`*`)`

Makes a legend label given the ModelResult of a SineModel fit.

Parameters: **result** : *LMFIT ModelResult*

The `lmfit.ModelResult` object, usually from `fit_sin`.

Returns: str

The legend label as a single string.

Notes

The fitting function is not in the preferred form. $F_{\text{fit}}(x; A, f, \phi, c) = A\sin(fx+\phi)+c, \quad F_{\text{plot}}(x; A, \nu, \delta, c) = A\sin(2\pi(x/\nu+\delta))+c$ This function does the conversion of both the values and uncertanties, and converts the time-units to seconds.

This function uses `sigfig.round` to handle significant figures and uncertanties properly.

`coinc_lmfit.`**`plot_chpairs`**`(`*`coinc_lists, res_lists=None, slice_lists=None, color_list=None, sin_pointnum=2000, ylabel_binwidth=None, figure_kwargs=None, errorbar_kwargs=None, sin_kwargs=None, legend_kwargs=None`*`)`

Plots the results of `coinc_sum` with or without `fit_sin`.

Takes one or multiple results of `coinc_sum` and plots it. Can also plot the results of one or multiple sin fits on top. Written with customization in mind.

Parameters: **coinc_lists** : *array-like*

A list of 3-tuples, where the elements of the 3-tuples are the results from a single call of `coinc_sum` (tbinmids, coinc_arr, unc_arr).

res_lists : *array-like containing LMFIT ModelResults, optional*

A list of lists of ModelResults from `sin_fit`. The length of *res_lists* should match the length of *coinc_list*, and the length of each list in *res_lists* should match the length of the corresponding list in *slice_lists*.

slice_lists : *array-like containing slice, optional*

A list of lists of slices (easily made with `make_slices`). The length of `slice_lists` should match the length of `coinc_lists`, and the length of each list in `slice_lists` should match the length of the corresponding list in `res_lists`.

Returns:

fig : *Matplotlib figure object*

axs : *ndarray of Matplotlib AxesSubplot*

ndrray of axes that the errorbars (and sin fits) are plotted on.

extra : *Matplotlib AxesSubplot*

The axis used for having common axis labels.

Other Parameters:

color_list : *list of strings, optional*

A list of the colors the sine plot lines should be. Passed to `ax.plot`, look at MatPlotLib documentation for acceptable color strings. Default is the MatPlotLib base colors (except black and white).

sin_pointnum : *int, optional*

The number of points used to plot the sine fits.

ylabel_binwidth : *int or str, optional*

If specified, then the common ylabel is automatically created.

figure_kwargs : *dict, optional*

A dict of keyword arguments to pass to `plt.figure`. Only overrides default arguments that are specified.

errorbar_kwargs : *dict, optional*

A dict of keyword arguments to pass to `ax.errorbar`. Only overrides default arguments that are specified.

sin_kwargs : *dict, optional*

A dict of keyword arguments to pass to `ax.plot` for plotting the sin fits. The color of the line should be specified with *color_list*, not in this dict. Only overrides default arguments that are specified.

legend_kwargs : *dict, optional*

A dict of keyword arguments to pass to `ax.legend`. Only overrides default arguments that are specified.

Notes

Look at the example file to see how to properly enter the required argument(s). Since this function returns the figure and the axes, additional formatting can be done on the returned figure (such as changing axis limits or labels, and saving the figure).