01_preprocessing

December 18, 2024

1 Spectrum preprocessing and data exploration

All spectral data (both training and test data) needs to be preprocessed before feeding it into a model. This notebook guides through the preprocessing procedure, involving the following steps: 1. Rebinning: Match all energy calibrations 2. Concatenate all files of the same isotope and detector to one dataset 3. Concatenate all files from the same isotope, different detectors to one dataset 4. Limit the number of spectra

In addition, this notebook provides several data exploration plots that help to understand the general data trends: - Visualization of the mean spectra for all isotopes (detectors seperate) - Visualization of original spectra for all isotopes - Visualization of the cosine similarity between all isotopes and detectors (triangular matrix)

The python library used for preprocessing is `tools_preprocessing.py`. All functions called in this notebook and some predefined variables can be found there.

[1]:
```
from tools_preprocessing import *
from plotting import *
```

Detector names As this package specializes on one-dimensional spectral data from multiple detectors, you can set detector names.
You can choose multiple detectors for measurements of spectra as well as one detector for simulated spectra.
In this example, we have two detectors 'right' and 'left' as well as one simulated detector.

[2]:
```
dets_measured = ['left', 'right']
det_simulated = 'simulated'
GlobalVariables.dets_measured = dets_measured
GlobalVariables.det_simulated = det_simulated

all_detectors = dets_measured + [det_simulated]
GlobalVariables.all_detectors = all_detectors
```

1.1 Obtain Data

To demonstrate the function of this tool, we provide some example data in `example_data.zip`. It contains measured and simulated spectra of the isotopes Am241, Co60, Cs137 and Eu152 as well as measured background and multi-label spectra (containing more than one isotope).

Unzip those data either manually or by running `install.sh` as described in `README`.
The spectra will be stored in `data/numpy_raw`.

If you want to use your own measured or simulated data, please save them in the respective folders.

Data Format In our example dataset, spectral data are present in the folder `dir_numpy_raw` as list of dictionaries (format: .npy).
Each dictionary includes the spectrum, labels, energy calibration, detector name and information on absorbers.
Here is an example dictionary:

{'spectrum': [0, 0, 1, 2, …],
'labels': ['Co60', 'background'],
'calibration': {'offset': -10., 'slope': 8., 'quad': 0.001},
'detector': 'left'}

Hence, you can access properties of a list element by: - loading the list of spectral data: `spectra_list = np.load(PATH, allow_pickle=True)` - picking an element from the list, e.g. the first element: `first_element = spectra_list[0]` - extracting a property, e.g. the spectrum: `first_spectrum = first_element['spectrum']`

Folder structure Next, the correct folder structure for input and output files (as depicted in README) has to be ensured. Relative paths are used.

In addition, you need to manually write a text file `00_list_of_isotopes.txt` in the folder `data` to control - which files are included in preprocessing - by which key different files of one isotope should be mapped together.

Example: e.g. the filenames `240805_Am241_outside_right.npy` and `Am241_00morning_left.npy` will be concatenated into one file later on,
and we need to specify the key `Am241` in `00_list_of_isotopes.txt`. This also works for isotopes containing special characters, e.g. Ra226+.

[3]:
```
check_folder_structure()
```

```
All essential folders should now be available.
The file data/00_list_of_isotopes exists.
Seems like you have successfully edited the txt-file data/00_list_of_isotopes.
Well done!
```

Retrieve filenames and isotopes Next, a list of measured and simulated spectral data files found in the directory `dir_numpy_raw` is created.
If multiple detectors were used for measurements, one entry per detector is created, as indicated by the suffix (e.g. 'Co60_1_left', 'Co60_1_right').

In addition, the isotope names are read from `00_list_of_isotopes.txt` in the directory `data`.

[4]:
```
all_files_list = sorted([x for x in listdir(dir_numpy_raw) if '.npy' in x])
isotope_list = read_isotope_list()
GlobalVariables.all_isotopes = isotope_list
```

Clearing directories from old preprocessed data To avoid confusion between previously processed data, the directories specified in `paths_to_delete` are cleared before starting the preprocessing.

[5]:
```
paths_to_delete = [dir_numpy_ready]
clear_preprocessed_data(paths_to_delete)
```

```
Deleting content of folder data/2_numpy_ready.
```

1.2 Preprocessing

1. Rebinning of all spectra to standard calibration The energy calibration of each spectrum is defined by the parameters `offset`, `slope` and `quad`, which establish the energy at each channel i:

```
energies = offset + slope * i + quad * i**2
```

To unify spectral data with varying energy calibrations, each spectrum is rebinned to a standard energy calibration `std_calib`.
You can specify the standard calibration in a dictionary, as shown below. Mathematically, the rebinning process interpolates each original spectrum from its specific energy calibration to the standard calibration.
Using linear interpolation, each spectrum's intensity values are recalculated to align with the channels in the standard energy calibration.
Undefined values are filled with a small baseline to prevent negative intensities.

Optionally, you can plot an original and rebinned example spectrum for each element in `all_files_list` by setting `show_plot=True`.

[6]:
```
std_calib = {'offset': -12., 'slope': 8.15, 'quad': 0.0012}
GlobalVariables.std_calib = std_calib

datalist_rebinned = rebinning(all_files_list, std_calib, show_plot=False)
```

```
 Starting rebinning of spectra…
```

2. Combine all datasets of the same isotope and detector in one As multiple datasets from an isotope and detector may exist, they are combined in this step.
To find out which datasets are combined together, filenames are screened for the isotopes defined in the text file `00_list_of_isotopes.txt`.

Example: 'Co60_1_right' and 'Co60_2_right' are concatenated to 'Co60_right' if `00_list_of_isotopes.txt` contains 'Co60', so only one dataset per isotope and detector remains.

[7]:
```
datalist_concat1 = concatenate_isotope_datasets(datalist_rebinned)
```

```
 Starting to concatenate files of same isotope (leave detectors separate)…
The files ['Co60-1_left.npy', 'Co60-2_left.npy'] are concatenated to
"Co60_left".
The files ['Co60-1_right.npy', 'Co60-2_right.npy'] are concatenated to
```

```
"Co60_right".
The files ['Cs137WenigBisKeinBlei_left.npy', 'Cs137inBlei_left.npy'] are
concatenated to "Cs137_left".
The files ['Cs137WenigBisKeinBlei_right.npy', 'Cs137inBlei_right.npy'] are
concatenated to "Cs137_right".
```

3. Combine all datasets of the of same isotopes, different detector in one In the next step, the datasets of different detectors (but same isotope) are concatenated.

Example: 'Co60_right' and 'Co60_left' are concatenated to 'Co60', so only one dataset per isotope remains.

[8]:
```
datalist_concat2 = concatenate_detector_datasets(datalist_concat1)
```

```
 Starting to concatenate files of same isotope for all detectors…
The datasets ['Am241_left', 'Am241_right', 'Am241_simulated'] are concatenated
to "Am241".
The datasets ['Co60_left', 'Co60_right', 'Co60_simulated'] are concatenated to
"Co60".
The datasets ['Cs137_left', 'Cs137_right', 'Cs137_simulated'] are concatenated
to "Cs137".
The datasets ['Eu152_left', 'Eu152_right', 'Eu152_simulated'] are concatenated
to "Eu152".
The datasets ['background_left', 'background_right'] are concatenated to
"background".
The datasets ['multi-class_left', 'multi-class_right'] are concatenated to
"multi-class".
```

4. Limit number of spectra At last, the number of spectra per isotope is limited to a number `n_max`.
You should choose `n_max` $>$ 100 to ensure a reasonable size of the test data later on.
For `n_max=None` no limit will be set and all data will be used.

The datasets are saved as .npy files in `dir_numpy_ready`.

[9]:
```
limit_spectra_per_isotope(datalist_concat2, n_max=None)  # e.g. n_max=10000
```

```
 You chose not to limit the number of spectra per dataset.
Am241: 2106 spectra originally
Co60: 3616 spectra originally
Cs137: 3680 spectra originally
Eu152: 2483 spectra originally
background: 1226 spectra originally
multi-class: 1284 spectra originally
```

1.3 Data exploration

In the following, the preprocessed data are visualized in different plots.
This also helps to check if the preprocessing was successful, i.e., if the results appear consistent.

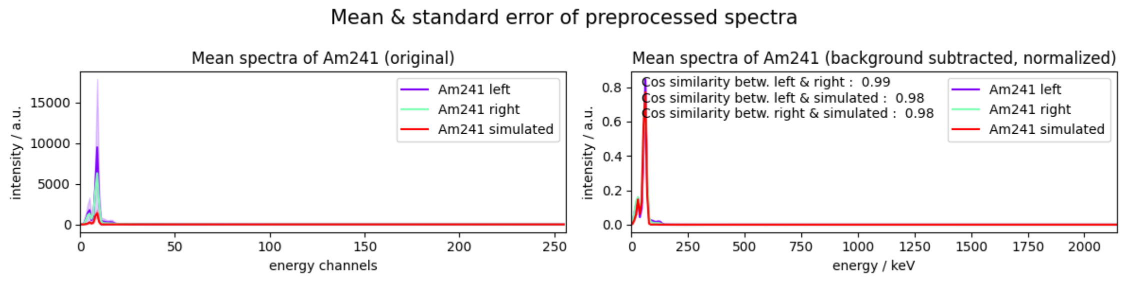
Comparison: Mean spectra from different detectors First, let's take a look at the mean spectra. The mean spectra of each isotope are plotted in individual plots, color-coded by the different detectors.
Choose `zoom_in=True` to take a closer look at the lower part of the spectrum and set `save_plots=True` if you want to save the plots.

On the left, the original mean spectra are shown whereas on the right, background is subtracted from measured data and all means are normalized.
Preferably, the (pure) measured background spectra found in the respective file (e.g. in `Am241.npy`) are used for background subtraction.
If none are found, the mean background spectrum is calculated from the separate file `background.npy`.

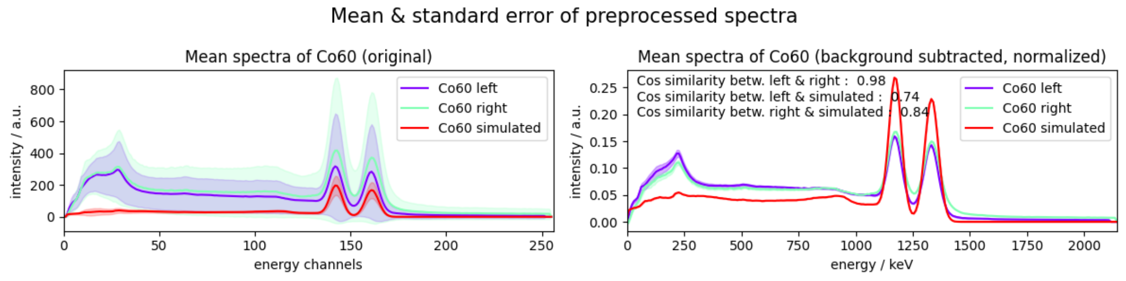
In addition, the cosine similarity (ranging from -1 to 1) between the different detectors is calculated to compare them quantitatively.
They should be close to 1 to allow for model inference between different detectors.

[10]:
```
plot_mean_spectra_by_isotope_and_detector(isotope_list, zoom_in=False,␣
 ↪save_plots=False)
```

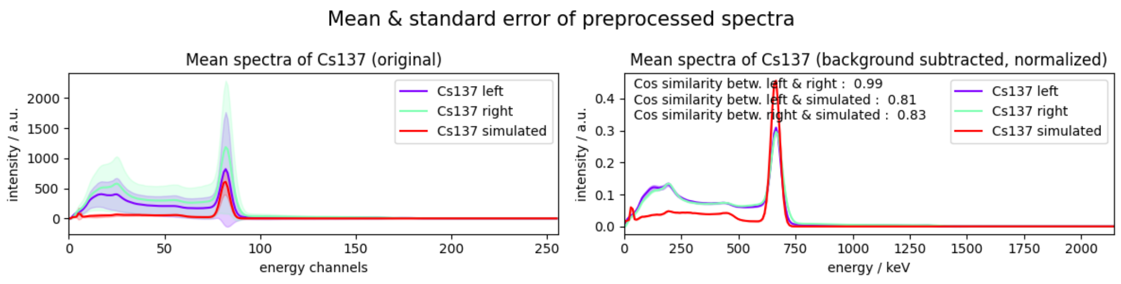
```
No pure background spectra found for Am241, detectors=['simulated'], using
background.npy instead.
```



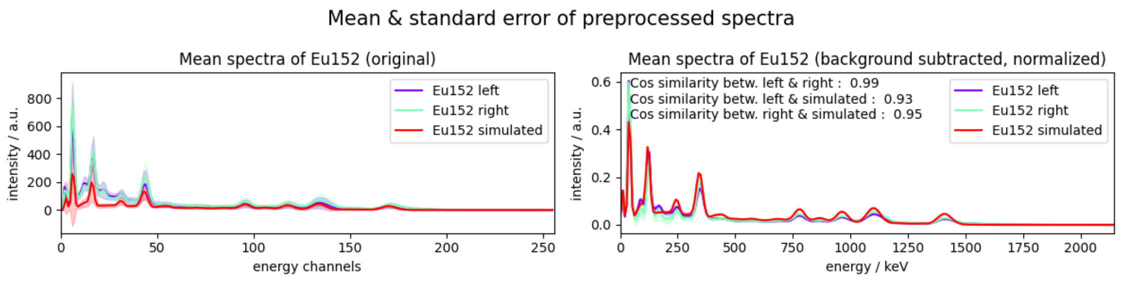
```
No pure background spectra found for Co60, detectors=['simulated'], using
background.npy instead.
```

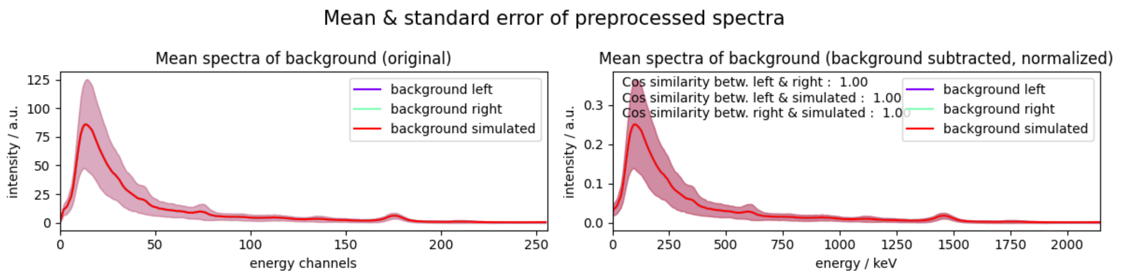


```
No pure background spectra found for Cs137, detectors=['simulated'], using
background.npy instead.
```



```
No pure background spectra found for Eu152, detectors=['simulated'], using
background.npy instead.
```





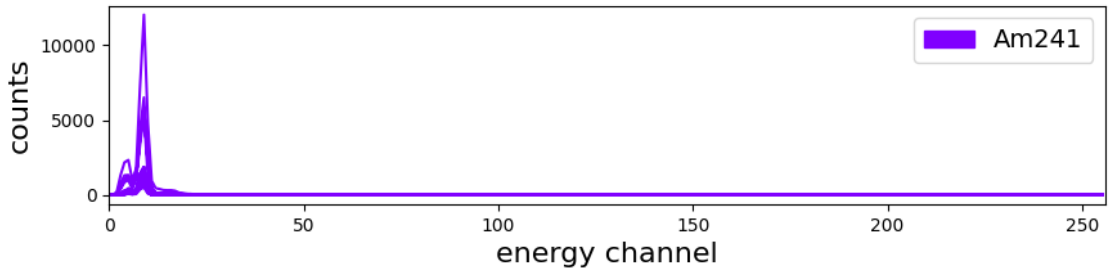
```
No isotope spectra were found for multi-class, ['left']. Returning array of
zeros.
No isotope spectra were found for multi-class, ['right']. Returning array of
zeros.
No isotope spectra were found for multi-class, ['simulated']. Returning array of
zeros.
```

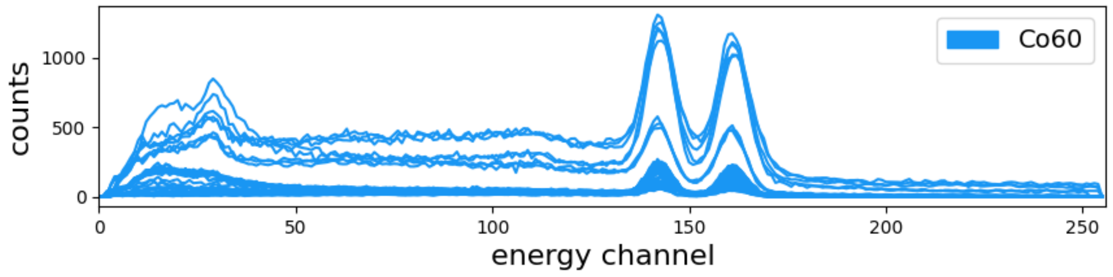
```
No pure background spectra found for multi-class, detectors=['simulated'], using
background.npy instead.
```

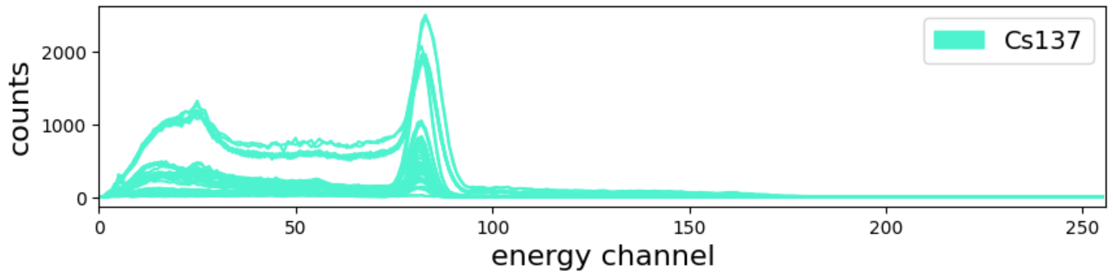
```
<Figure size 640x480 with 0 Axes>
```

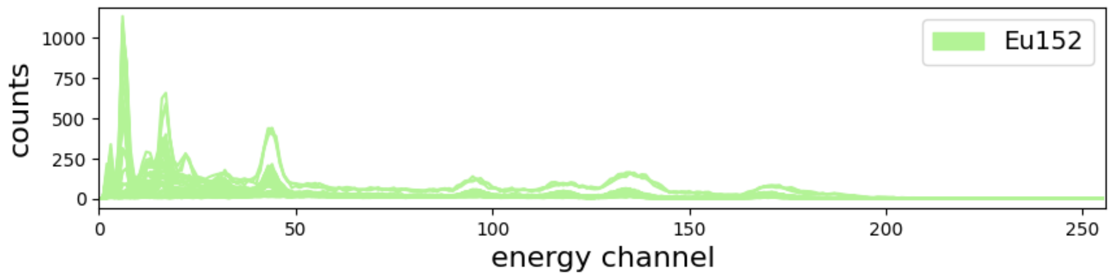
Example spectra for all isotopes Below, example spectra of the preprocessed data are shown for all isotopes in colorful plots.
You can choose how many spectra are to be shown and whether you want to save the plot.

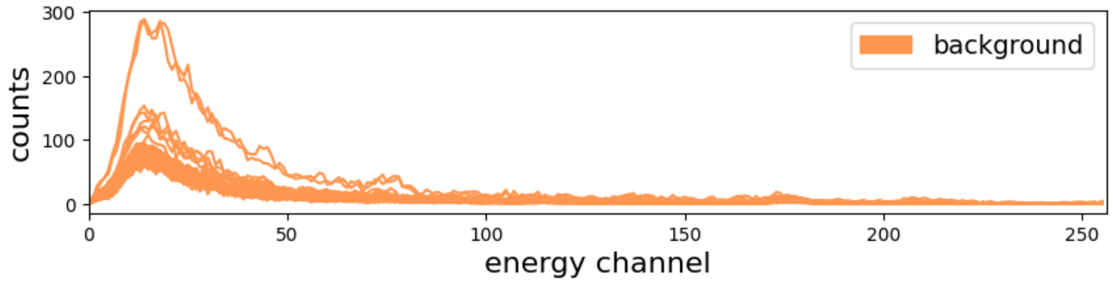
```
[11]: plot_example_spectra_by_isotopes(isotope_list, n_spectra=80, save_plot=False)
```











```
No spectra labelled as multi-class found in multi-class.npy, cannot be plotted.
```

Correlation between spectra of different isotopes and detectors: Cosine similarity matrix One-dimensional spectra can be interpreted as vectors and their similarity can be quantified by the cosine similarity.
A cosine similarity of 1 means very similar spectra (modulo a scalar) and 0 means orthogonal, i.e., very different.

While it is favourable to have high cosine similarities (close to 1) between spectra of the same isotope,
spectra from different isotopes should not be too similar, thus have low cosine similarities close to 0.
This will help the model to distinguish different isotopes and avoid confusion.

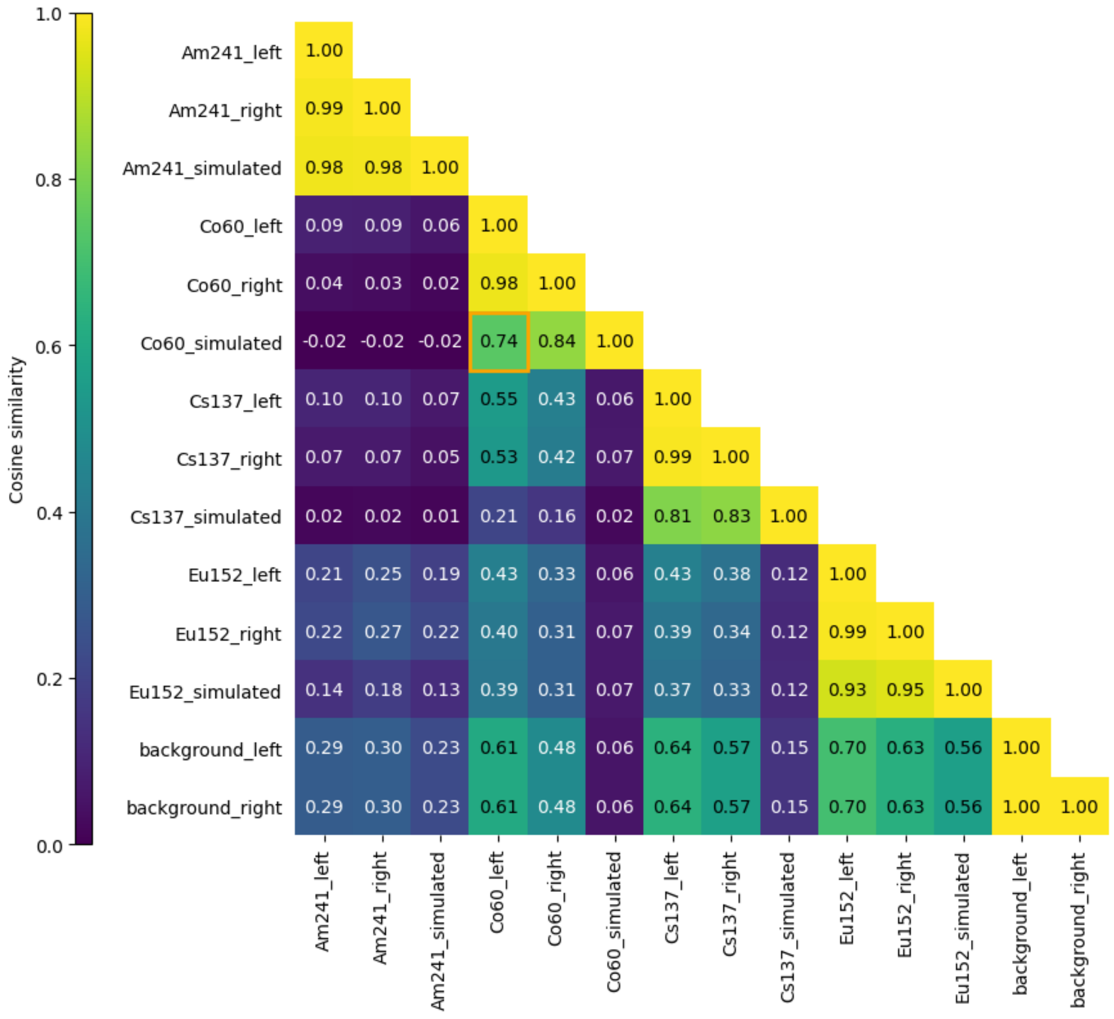
Below, the cosine similarity between all isotopes and detectors is calculated.
It serves as a first assessment if the data are suited for this model.
Critical values that are too low/high are rimmed and you can set `threshold` manually to a value between 0 and 1:
- Orange-rimmed: means of the same isotope that are not similar enough (cos_sim < threshold)
- Red-rimmed: means of different isotopes that are too similar (cos_sim >= threshold)

For the example dataset, `threshold=0.8` has proven to be a reasonable and helpful estimate.

[12]:
```
cossim_mat, names = calc_cos_sim_matrix_means(isotope_list)
plot_cos_sim_matrix_means(cossim_mat, names, threshold=0.8)
```

```
No pure background spectra found for Am241, detectors=['simulated'], using
background.npy instead.
No pure background spectra found for Co60, detectors=['simulated'], using
background.npy instead.
No pure background spectra found for Cs137, detectors=['simulated'], using
background.npy instead.
No pure background spectra found for Eu152, detectors=['simulated'], using
background.npy instead.
No spectra labelled as multi-class found in multi-class.npy, cannot be included
in the cosine similarity matrix!

Not similar enough: Co60_simulated and Co60_left, cosine similarity=0.743 in row
5, column 3
```


Cosine similarity
Am241_left 1.00
Am241_right 0.99 1.00
Am241_simulated 0.98 0.98 1.00
Co60_left 0.09 0.09 0.06 1.00
Co60_right 0.04 0.03 0.02 0.98 1.00
Co60_simulated -0.02 -0.02 -0.02 0.74 0.84 1.00
Cs137_left 0.10 0.10 0.07 0.55 0.43 0.06 1.00
Cs137_right 0.07 0.07 0.05 0.53 0.42 0.07 0.99 1.00
Cs137_simulated 0.02 0.02 0.01 0.21 0.16 0.02 0.81 0.83 1.00
Eu152_left 0.21 0.25 0.19 0.43 0.33 0.06 0.43 0.38 0.12 1.00
Eu152_right 0.22 0.27 0.22 0.40 0.31 0.07 0.39 0.34 0.12 0.99 1.00
Eu152_simulated 0.14 0.18 0.13 0.39 0.31 0.07 0.37 0.33 0.12 0.93 0.95 1.00
background_left 0.29 0.30 0.23 0.61 0.48 0.06 0.64 0.57 0.15 0.70 0.63 0.56 1.00
background_right 0.29 0.30 0.23 0.61 0.48 0.06 0.64 0.57 0.15 0.70 0.63 0.56 1.00 1.00
Am241_left
Am241_right
Am241_simulated
Co60_left
Co60_right
Co60_simulated
Cs137_left
Cs137_right
Cs137_simulated
Eu152_left
Eu152_right
Eu152_simulated
background_left
background_right