EMIT SpectralUnmixing Package

<u>EndmemberLibrary.jl</u>

-SpectralLibrary

- The SpectralLibrary struct manages spectral data from a file, supporting class labels, scaling, and wavelength filtering.
- **file_name:** Data file name.
- **class_header_name:** Column for class labels in data file.
- **spectral_starting_column:** Start of spectral data (default: 2).
- **truncate_end_columns:** Number of columns to ignore at the end (default: 0).
- **class_valid_keys:** List of valid class labels (default: nothing).
- **scale_factor:** Scaling factor (default: 1.0).
- **wavelength_regions_ignore:** Wavelength regions to ignore (default: [0, 440, 1310, 1490, 1770, 2050, 2440,2880]).
- **spectra:** Spectral data matrix.
- **classes:** Class labels array.
- **good_bands:** Mask for valid wavelengths.
- **wavelengths:** Array of wavelengths.

  Constructor: SpectralLibrary(file_name, class_header_name, spectral_starting_column = 2, truncate_end_columns = 0, class_valid_keys = nothing, scale_factor = 1.0, wavelength_regions_ignore = [0,440,1310,1490,1770,2050,2440,2880])

-load_data!(library::SpectralLibrary)

- Loads the data from library.file_name into a DataFrame.
- Processes wavelength regions to ignore, creating paired ranges (Ex: [start, end]).
- Extracts spectral data based on the specified start and end columns.
- Retrieves class labels from the column specified by library.class_header_name.
- Extracts wavelengths from the DataFrame and ensures they are numerical and sorted.
- Creates a mask (good_bands) to filter out wavelengths within ignored regions.
- Ensures valid class keys by assigning unique class labels if class_valid_keys is not provided.
- Converts wavelength units from microns to nanometers if needed for consistency.
- Returns a SpectralLibrary

-filter_by_class!(library::SpectralLibrary)

- Checks if library.class_valid_keys is provided. If not, logs a message and skips filtering.
- Creates a Boolean mask (valid_classes) that marks which spectra belong to the valid classes.
- Filters the spectra and class labels to include only those matching the valid classes.
- Returns a SpectralLibrary

-interpolate_library_to_new_wavelengths!(library::SpectralLibrary, new_wavelengths::Array{Float64})

- Copies the original spectra to preserve the current state.
- Uses linear interpolation to resample each spectrum from the old wavelength values to the new_wavelengths.
- Updates the library.spectra matrix to the resampled spectra.
- Updates the library.wavelengths to new_wavelengths.
- Recalculates the "good bands" mask based on the new wavelengths and ignored regions.
- No Return Object.

-remove_wavelength_region_inplace!(library::SpectralLibrary, set_as_nans::Bool=false)

- If set_as_nans is true, the unwanted wavelengths (as defined by good_bands) in both spectra and wavelengths are replaced with NaN.
- If set_as_nans is false, the unwanted wavelengths are removed entirely from the spectra and wavelength arrays, and the good_bands mask is updated accordingly.
- No Return Object

-split_library(library::SpectralLibrary, split_fraction::Float64)

- Randomly permutes the indices of the spectra in the library.
- Divides the indices into two subsets based on the split_fraction.
- Creates deep copies of the original library and assigns the spectra and class labels to each subset.
- Returns two new spectral libraries: one containing the first subset of spectra, and the other containing the remaining spectra.

-get_good_bands_mask(wavelengths::Array{Float64}, wavelength_pairs)

- Creates a boolean array (good_bands), initialized to true for all wavelengths.
- Loops over each pair of wavelengths, where each pair specifies a range of wavelengths to exclude.
- For each pair, finds the index of the closest wavelength that is greater than or equal to the lower bound of the pair.
- Finds the index of the closest wavelength that is less than or equal to the upper bound of the pair.
- Updates the good_bands array, setting the range of wavelengths between the lower and upper bounds to false, indicating that these wavelengths should be excluded.
- Returns the good_bands mask

<u>Plotting.jl</u>

-plot_endmembers_individually(endmember_library::SpectralLibrary, output_name::String (optional))

- Loops through each class in the spectral library (class_valid_keys).
- Selects spectra corresponding to each class and replaces NaN values with 0.
- Plots the reflectance of each class across the wavelengths. If only the "good bands" are present, those are used; otherwise, all wavelengths are plotted.
- Customizes x-axis ticks and plot size, with a legend placed in the top-right corner.
- Saves the plot to the specified file if output_name is provided.
- No Return Object

<u>SpectralUnmixing.jl</u>
-simulate_pixel(library::SpectralLibrary, max_components::Int64)

- Sets the random seed
- Retrieves class-based index combinations using prepare_combinations.
- Selects endmembers using get_sma_permutation(), based on the combination_type and max_components.
- Randomly generates the proportions for each selected endmember to form a weighted mixture.
- Creates an output distribution to track which spectra contribute to the simulated pixel.
- Calculates the class-wise distribution of the pixel based on the contributing spectra.
- Computes the simulated reflectance as the weighted sum of the selected spectra.
- Returns simulated_rfl, output_distribution, output_distribution_classes representing simulated pixel's reflectance values, vector indicating the proportion of each individual spectrum in the mix, and the proportion of each class in the mix.

-get_sma_permuatation(class_idx, num_endmembers::Vector{Int64}, comination_type::String, library_length::Int64)

- For "class-even":
  - Shuffles the indices for each class.
  - Selects one endmember from each class until the required number of endmembers is selected.
  - Removes indices from the shuffled list once selected.
- For other types:
  - Randomly selects the required number of endmembers from the entire library.
- If num_endmembers[1] == -1, it returns all indices from the library.
- Returns a vector of selected indices representing the permutation of spectra

-prepare_combinations(library::SpectralLibrary, combination_type::String)

- For "class-even":
  - Iterates over the valid class keys in library.class_valid_keys.
  - For each class, it extracts the indices of spectra that belong to that class and stores them in class_idx.
- Returns the list of indices grouped by class

-prepare_options(library::SpectralLibrary, combination_type::String, num_endmembers::Vector{Int64}, class_idx::Vector{Vector{Int64}})

- For "class-even":
  - Generates all combinations where one endmember is selected from each class.

- For "all":
    - Generates all possible combinations of spectra based on the specified num_endmembers.
- Raises an error if an invalid combination type is provided.
- Returns list of combinations where each is a list of indices of selected spectra.

-scale_data(refl::Matrix{Float64}, wavelengths::Vector{Float64}, criteria::String)

- For "none":
    - Returns refl unchanged.
- For "brightness":
    - Excludes noisy wavelength regions [1300, 1500] and [1800, 2000]. Computes normalization by taking the square root of the mean squared reflectance over the good bands.
- Else:
    - Normalizes each pixel or spectrum so that the reflectance at the specified wavelength equals 0.5.
- Returns normalized reflectance matrix based on the specified criteria.

-unmix_pixel(library::SpectralLibrary, img_dat_input::Array{Float64}, unc_dat, class_idx, options, mode::String, n_mc::Int64, num_endmembers::Vector{Int64}, normalization::String, optimization::String, max_combinations::Int64, combination_type::String)

- Ensures img_dat_input is in a 2D format.
- Monte Carlo Loop:
    - Performs n_mc Monte Carlo iterations
    - Chooses between "pinv", "qr", and "default" for the least squares solution method based on the optimization parameter.
    - Unmixing for "sma" and "sma-best":
        - Selects endmembers using permutations, solves the least squares problem, and applies optimization methods (bvls, ldsqp, or inverse)
    - Unmixing for "mesma" and "mesma-best":
        - Evaluates multiple endmember combinations, selects the best solution, and calculates the component fractions.
- Normalization and Aggregation:
    - Normalizes the component fractions and aggregates the results to per-class values.
- Returns:
    - output_mixture: Final aggregated proportion results.
    - output_mixture_var: Variance of the mixture results.

    - output_comp_frac: Individual component proportions for each Monte Carlo iteration.
    - output_comp_frac_var: Variance of the component proportions.

-results_from_mc(results::Matrix{Float64}, cost::Vector{Float64}, mode::String)

- If there’s only one Monte Carlo iteration, output_var is set to nothing (no variance to compute).
- For multiple iterations, calculates the standard deviation across iterations for each component
- If the mode contains "best", selects the result with the lowest cost. Otherwise, it will select the average.
- Returns the selected result and the variance

__Solvers.jl__
- dolsq(A, b; method::String="default")
- For "default":
  - Solves the system using the backslash operator (A \ b).
- For "pinv":
  - Uses the pseudoinverse (pinv(A) * b).
- For "qr":
  - Solves the system using the QR decomposition (qr(A) \ b).
- Returns solution vector x

- compute_kkt_optimality(g::Vector{Float64}, on_bound::Vector)
- Calculates g_kkt by multiplying gradient g and on_bound.
- Sets the values of g_kkt for free variables (on_bound == 0) to the absolute value of their gradient.
- Returns the largest value in g_kkt, representing the biggest KKT condition violation.

-bvls(A, b, x_lsq, lb, ub, tol::Float64, max_iter::Int64, verbose::Int64, inverse_method::String)
- Sets the solution x equal to the least squares estimate x_lsq.
- Edits x to stay within the lower (lb) and upper (ub) bounds.
- on_bound tracks which components are at their lower (-1), upper (1), or free (0) bounds.
- Free set and Active set: Variables that represent which components are not at bounds and at bounds, respectively.
- First Loop (While Free Variables Exist):
  - Solves the least squares problem for the free variables using the method specified
  - Identifies variables that violate bounds and edits them to their respective lower or upper bounds.
  - Updates x for variables within bounds and recalculates key metrics like residuals, cost, and gradient.
  - The loop continues until all bound violations are fixed.
- Second Loop (For Loop):
  - Runs for up to max_iter iterations.
  - Checks if the solution meets the KKT optimality conditions; if optimality is achieved, the loop stops.
  - Frees the variable with the highest potential for improvement and adjusts the solution in an inner loop.
  - Solves the least squares problem for newly freed variables, checks for bound violations, and updates x.

    - Continues adjusting the solution until a feasible result is found or optimality is achieved.
    - If the cost change is small enough, the loop ends.
- Sets very small values in x (less than 1e-5) to zero
- Determines how the function stopped:
    - 1 if optimality conditions are met.
    - 2 if the cost stopped changing significantly.
    - 0 if the maximum number of iterations was reached.
- Returns the solution x, and the final cost (sum of squared residuals)

-opt_solve(A::Matrix{Float64}, b::Vector{Float64}, x0::Vector{Float64}, lb::Vector{Float64}, ub::Vector{Float64})

- Initializes the optimization model using NLopt.Optimizer with the :LD_SLSQP algorithm for constrained optimization.
- Edits x0 values to the range [0, 1].
- Defines decision variables x with bounds lb[i] <= x[i] <= ub[i] and initializes them with values from x0.
- Minimizes the mean squared error (MSE) between b and the predicted values Ax:
- Solves the optimization problem using JuMP.optimize!(mle).
- Returns x, solution vector, and the exponential of the minimized MSE