memo

Release 0.1.0

Arnaud Gaudry

Feb 18, 2022

CONTENTS:

CHAPTER

ONE

MEMO_MS PACKAGE

class memo_ms.**FeatureTable**(*path: str*, *software: str*)

Bases: object

Create a FeatureTable dataclass object from a feature table

Args: path (str): Path to a feature table file (.csv) software (str): One of [mzmine, xcms, msdial, memo]: the software used for feature detection.

Returns: self.feature_table (DataFrame): A cleaned feature quantification table

__init__(*path: str*, *software: str*) → None

export_matrix(*path*, *sep=','*)

Export a given matrix

Args: path (str): path to export sep (str): separator

Returns: None

filter(*use_samples_pattern=False*, *samples_pattern=''*, *max_occurence=None*, *min_rel_occurence=0*, *max_rel_occurence=1*)

Filter a feature table: remove samples matching samples_pattern AND remove features occuring in more than n = max_occurence samples matched by samples_pattern

Args: use_samples_pattern (bool): filter using a str pattern samples_pattern (str): the str pattern to match in samples to filter max_occurence (int): maximal number of occurence allowed in matched samples before removing a feature min_rel_occurence (float): remove features contained in less than (min_rel_occurence * 100) percent of the samples max_rel_occurence (float): remove features contained in more than (max_rel_occurence * 100) percent of the samples

Returns: self.filtered_feature_table (DataFrame): A filtered feature table

class memo_ms.**MemoMatrix**

Bases: object

Create an empty MemoMatrix dataclass object

__init__() → None

export_matrix(*path*, *sep=','*)

Export a given matrix

Args: path (str): path to export sep (str): separator

Returns: None

`filter`(*use_samples_pattern=False*, *samples_pattern=''*, *max_occurence=None*, *min_rel_occurence=0*, *max_rel_occurence=1*)

Filter a MEMO matrix: remove samples matching samples_pattern AND remove features occuring in more than n = max_occurence samples matched by samples_pattern

Args: use_samples_pattern (bool): filter using a str pattern samples_pattern (str): the str pattern to match in samples to filter max_occurence (int): maximal number of occurence allowed in matched samples before removing a word min_rel_occurence (float): remove words contained in less than (min_rel_occurence * 100 percent) of the samples max_rel_occurence (float): remove words contained in more than (max_rel_occurence * 100 percent) of the samples

Returns: self.memo_matrix (DataFrame): A filtered feature table matrix

`memo_from_aligned_samples`(*featuretable*, *spectradocuments*) → pandas.core.frame.DataFrame

Use a featuretable and a spectradocuments to generate a MEMO matrix. Returns a pd.DataFrame MEMO matrix.

Args: featuretable (FeatureTable): a FeatureTable dataclass object spectradocuments (SpectraDocuments): a SpectraDocuments dataclass oject

Returns: self.memo_matrix (DataFrame): A MEMO matrix

`memo_from_unaligned_samples`(*path_to_samples_dir*, *pattern_to_match='.mgf'*, *min_relative_intensity=0.01*, *max_relative_intensity=1.0*, *min_peaks_required=10*, *losses_from=10*, *losses_to=200*, *n_decimals=2*)

Generate a Memo matrix from a list of individual .mgf files

Args: path_to_samples_dir (str): Path to the directory where individual .mgf files are gathered. Subfolders will also be checked. pattern_to_match (str): Shared pattern between all spectra files to input. Will be removed in memo_matrix.index. min_relative_intensity (float): Minimal relative intensity to keep a peak max_relative_intensity (float): Maximal relative intensity to keep a peak min_peaks_required (int): Minimum number of peaks to keep a spectrum losses_from (int): minimal m/z value for losses losses_to (int): maximal m/z value for losses n_decimals (int): number of decimal when translating peaks/losses into words

Returns: self.memo_matrix (DataFrame): A MEMO matrix

`merge_memo`(*memomatrix_2*, *drop_not_in_common=False*)

Merge 2 MEMO matrix

Args: memocontainer2 (MemoContainer): MemoMatrix dataclass object containing the 2nd MEMO matrix to merge drop_not_in_common (bool): Drop peaks/losses not in common

Returns: MemoContainer (MemoContainer): A MemoMatrix dataclass object containing the merged MEMO matrix

`class memo_ms.SpectraDocuments`(*path: str*, *min_relative_intensity: float = 0.01*, *max_relative_intensity: float = 1.0*, *min_peaks_required: int = 10*, *losses_from: int = 10*, *losses_to: int = 200*, *n_decimals: int = 2*)

Bases: `object`

Create a SpectraDocuments dataclass object containing spectra documents and metadata from an MzMine 2 spectra file (.mgf)

Args: path (str): Path to spectra file (.mgf) min_relative_intensity (float): Minimal relative intensity to keep a peak max_relative_intensity (float): Maximal relative intensity to keep a peak min_peaks_required (int): Minimum number of peaks to keep a spectrum losses_from (int): minimal m/z value for losses losses_to (int): maximal m/z value for losses n_decimals (int): number of decimal when translating peaks/losses into words

Returns: self.document (DataFrame): A table containing spectra documents and metadata

`__init__`(*path: str*, *min_relative_intensity: float = 0.01*, *max_relative_intensity: float = 1.0*, *min_peaks_required: int = 10*, *losses_from: int = 10*, *losses_to: int = 200*, *n_decimals: int = 2*) → None

1.1 Submodules

1.1.1 memo_ms.classes module

`class memo_ms.classes.`**`FeatureTable`**(*path: str*, *software: str*)

Bases: `object`

Create a FeatureTable dataclass object from a feature table

Args: path (str): Path to a feature table file (.csv) software (str): One of [mzmine, xcms, msdial, memo]: the software used for feature detection.

Returns: self.feature_table (DataFrame): A cleaned feature quantification table

`__init__`(*path: str*, *software: str*) → None

`export_matrix`(*path*, *sep=','*)

Export a given matrix

Args: path (str): path to export sep (str): separator

Returns: None

`filter`(*use_samples_pattern=False*, *samples_pattern=''*, *max_occurence=None*, *min_rel_occurence=0*, *max_rel_occurence=1*)

Filter a feature table: remove samples matching samples_pattern AND remove features occuring in more than n = max_occurence samples matched by samples_pattern

Args: use_samples_pattern (bool): filter using a str pattern samples_pattern (str): the str pattern to match in samples to filter max_occurence (int): maximal number of occurence allowed in matched samples before removing a feature min_rel_occurence (float): remove features contained in less than (min_rel_occurence * 100) percent of the samples max_rel_occurence (float): remove features contained in more than (max_rel_occurence * 100) percent of the samples

Returns: self.filtered_feature_table (DataFrame): A filtered feature table

`class memo_ms.classes.`**`MemoMatrix`**

Bases: `object`

Create an empty MemoMatrix dataclass object

`__init__`() → None

`export_matrix`(*path*, *sep=','*)

Export a given matrix

Args: path (str): path to export sep (str): separator

Returns: None

`filter`(*use_samples_pattern=False*, *samples_pattern=''*, *max_occurence=None*, *min_rel_occurence=0*, *max_rel_occurence=1*)

Filter a MEMO matrix: remove samples matching samples_pattern AND remove features occuring in more than n = max_occurence samples matched by samples_pattern

Args: use_samples_pattern (bool): filter using a str pattern samples_pattern (str): the str pattern to match in samples to filter max_occurence (int): maximal number of occurence allowed in matched samples before removing a word min_rel_occurence (float): remove words contained in less than (min_rel_occurence * 100 percent) of the samples max_rel_occurence (float): remove words contained in more than (max_rel_occurence * 100 percent) of the samples

Returns: self.memo_matrix (DataFrame): A filtered feature table matrix

`memo_from_aligned_samples`(*featuretable*, *spectradocuments*) → pandas.core.frame.DataFrame

Use a featuretable and a spectradocuments to generate a MEMO matrix. Returns a pd.DataFrame MEMO matrix.

Args: featuretable (FeatureTable): a FeatureTable dataclass object spectradocuments (SpectraDocuments): a SpectraDocuments dataclass oject

Returns: self.memo_matrix (DataFrame): A MEMO matrix

`memo_from_unaligned_samples`(*path_to_samples_dir*, *pattern_to_match='.mgf'*, *min_relative_intensity=0.01*, *max_relative_intensity=1.0*, *min_peaks_required=10*, *losses_from=10*, *losses_to=200*, *n_decimals=2*)

Generate a Memo matrix from a list of individual .mgf files

Args: path_to_samples_dir (str): Path to the directory where individual .mgf files are gathered. Subfolders will also be checked. pattern_to_match (str): Shared pattern between all spectra files to input. Will be removed in memo_matrix.index. min_relative_intensity (float): Minimal relative intensity to keep a peak max_relative_intensity (float): Maximal relative intensity to keep a peak min_peaks_required (int): Minimum number of peaks to keep a spectrum losses_from (int): minimal m/z value for losses losses_to (int): maximal m/z value for losses n_decimals (int): number of decimal when translating peaks/losses into words

Returns: self.memo_matrix (DataFrame): A MEMO matrix

`merge_memo`(*memomatrix_2*, *drop_not_in_common=False*)

Merge 2 MEMO matrix

Args: memocontainer2 (MemoContainer): MemoMatrix dataclass object containing the 2nd MEMO matrix to merge drop_not_in_common (bool): Drop peaks/losses not in common

Returns: MemoContainer (MemoContainer): A MemoMatrix dataclass object containing the merged MEMO matrix

`class memo_ms.classes.SpectraDocuments`(*path: str*, *min_relative_intensity: float = 0.01*, *max_relative_intensity: float = 1.0*, *min_peaks_required: int = 10*, *losses_from: int = 10*, *losses_to: int = 200*, *n_decimals: int = 2*)

Bases: `object`

Create a SpectraDocuments dataclass object containing spectra documents and metadata from an MzMine 2 spectra file (.mgf)

Args: path (str): Path to spectra file (.mgf) min_relative_intensity (float): Minimal relative intensity to keep a peak max_relative_intensity (float): Maximal relative intensity to keep a peak min_peaks_required (int): Minimum number of peaks to keep a spectrum losses_from (int): minimal m/z value for losses losses_to (int): maximal m/z value for losses n_decimals (int): number of decimal when translating peaks/losses into words

Returns: self.document (DataFrame): A table containing spectra documents and metadata

`__init__`(*path: str*, *min_relative_intensity: float = 0.01*, *max_relative_intensity: float = 1.0*, *min_peaks_required: int = 10*, *losses_from: int = 10*, *losses_to: int = 200*, *n_decimals: int = 2*) → None

1.1.2 memo_ms.import_data module

`memo_ms.import_data.`**`import_memo_quant_table`**(*path*) → pandas.core.frame.DataFrame

Import feature quantification table memo ready

Args: path (str): Path to a MEMO ready feature quantification table: a csv file (sep = “,”) with feature as rows and samples as columns.

The first column must contain feature’s ID and the header must be “feature_id”.

Returns: quant_table (DataFrame): A cleaned feature quantification table

`memo_ms.import_data.`**`import_msdial_quant_table`**(*path*) → pandas.core.frame.DataFrame

Import feature quantification table generated from MS-DIAL and clean it

Args: path (str): Path to feature quantification table

Returns: quant_table (DataFrame): A cleaned MS-DIAL feature quantification table

`memo_ms.import_data.`**`import_mzmine2_quant_table`**(*path*) → pandas.core.frame.DataFrame

Import feature quantification table generated from MzMine 2 and clean it

Args: path (str): Path to feature quantification table

Returns: quant_table (DataFrame): A cleaned MzMine2 feature quantification table

`memo_ms.import_data.`**`import_xcms_quant_table`**(*path*) → pandas.core.frame.DataFrame

Import feature quantification table generated from XCMS and clean it

Args: path (str): Path to feature quantification table

Returns: quant_table (DataFrame): A cleaned XCMS feature quantification table

`memo_ms.import_data.`**`load_and_filter_from_mgf`**(*path*, *min_relative_intensity*, *max_relative_intensity*, *loss_mz_from*, *loss_mz_to*, *n_required*) → list

Load and filter spectra from mgf file to prepare for MEMO matrix generation

Returns: spectrums (list of matchms.spectrum): a list of matchms.spectrum objects

1.1.3 memo_ms.visualization module

`memo_ms.visualization.`**`plot_hca`**(*matrix*, *df_metadata*, *filename_col*, *group_col*, *plotly_discrete_cm=['#636EFA', '#EF553B', '#00CC96', '#AB63FA', '#FFA15A', '#19D3F3', '#FF6692', '#B6E880', '#FF97FF', '#FECB52']*, *linkage_method='ward'*, *linkage_metric='euclidean'*, *norm=False*, *scaling=False*)

Simple HCA plot of a MEMO matrix / Feature table using matplotlib

Args: matrix (DataFrame): A Table in the MemoMatrix.memo_matrix or FeatureTable.feature_table format df_metadata (DataFrame): Metadata of the MEMO matrix samples filename_col (str): Column name in df_metadata to match memo_matrix index group_col (str): Column name in df_metadata to use as groups for plotting plotly_discrete_cm ([type], optional): Plotly discrete colormap to use for groups. Defaults to px.colors.qualitative.Plotly. linkage_method (str, optional): Linkage method to use. Defaults to ‘ward’. linkage_metric (str, optional): Linkage metric to use. Defaults to ‘euclidean’. norm (bool, optional): Apply samples normalization. Defaults to False. scaling (bool, optional): Apply pareto scaling to MEMO matrix columns. Defaults to False.

Returns: None

`memo_ms.visualization.`**`plot_heatmap`**(*matrix*, *df_metadata*, *filename_col*, *group_col*, *plotly_discrete_cm=['#636EFA', '#EF553B', '#00CC96', '#AB63FA', '#FFA15A', '#19D3F3', '#FF6692', '#B6E880', '#FF97FF', '#FECB52']*, *linkage_method='ward'*, *linkage_metric='euclidean'*, *heatmap_metric='braycurtis'*, *norm=False*, *scaling=False*)

HCA and heatmap plot of a MEMO matrix / Feature table using Plotly

Args: matrix (DataFrame): A Table in the MemoMatrix.memo_matrix or FeatureTable.feature_table format df_metadata (DataFrame): Metadata of the MEMO matrix samples filename_col (str): Column name in df_metadata to match memo_matrix index group_col (str): Column name in df_metadata to use as groups for plotting plotly_discrete_cm ([type], optional): Plotly discrete colormap to use for groups. Defaults to px.colors.qualitative.Plotly. linkage_method (str, optional): Linkage method to use. Defaults to 'ward'. linkage_metric (str, optional): Linkage metric to use. Defaults to 'euclidean'. heatmap_metric (str, optional): Distance metric to use for heatmap. Defaults to 'braycurtis'. norm (bool, optional): Apply samples normalization. Defaults to False. scaling (bool, optional): Apply pareto scaling to MEMO matrix columns. Defaults to False.

Returns: None

`memo_ms.visualization.`**`plot_pcoa_2d`**(*matrix*, *df_metadata*, *filename_col*, *group_col*, *metric='braycurtis'*, *norm=False*, *scaling=False*, *pc_to_plot=(1, 2)*)

Simple 2D PCoA plot of a MEMO matrix / Feature table using Plotly

Args: matrix (DataFrame): A Table in the MemoMatrix.memo_matrix or FeatureTable.feature_table format df_metadata (DataFrame): Metadata of the MEMO matrix samples filename_col (str): Column name in df_metadata to match memo_matrix index group_col (str): Column name in df_metadata to use as groups for plotting metric (str, optional): Distance metric to use, see https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.distance.pdist.html. Defaults to 'braycurtis'. norm (bool, optional): Apply samples normalization. Defaults to False. scaling (bool, optional): Apply pareto scaling to MEMO matrix columns. Defaults to False. pc_to_plot (list of int, optional): PCs to plot. Defaults to [1,2].

Returns: None

`memo_ms.visualization.`**`plot_pcoa_3d`**(*matrix*, *df_metadata*, *filename_col*, *group_col*, *metric='braycurtis'*, *norm=False*, *scaling=False*, *pc_to_plot=(1, 2, 3)*)

Simple 2D PCoA plot of a MEMO matrix / Feature table using Plotly

Args: matrix (DataFrame): A Table in the MemoMatrix.memo_matrix or FeatureTable.feature_table format df_metadata (DataFrame): Metadata of the MEMO matrix samples filename_col (str): Column name in df_metadata to match memo_matrix index group_col (str): Column name in df_metadata to use as groups for plotting metric (str, optional): Distance metric to use, see https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.distance.pdist.html. Defaults to 'braycurtis'. norm (bool, optional): Apply samples normalization. Defaults to False. scaling (bool, optional): Apply pareto scaling to MEMO matrix columns. Defaults to False. pc_to_plot (list of int, optional): PCs to plot. Defaults to [1,2,3].

Returns: None

CHAPTER TWO

DESCRIPTION

MEMO is a method allowing a Retention Time (RT) agnostic alignment of metabolomics samples using the fragmentation spectra (MS2) of their constituents. The occurrence of MS2 peaks and neutral losses (to the precursor) in each sample is counted and used to generate an *MS2 fingerprint* of the sample. These fingerprints can in a second stage be aligned to compare different samples. Once obtained, different filtering (remove peaks/losses from blanks for example) and visualization techniques (MDS/PCoA, TMAP, Heatmap, . . .) can be used. MEMO suits particularly well to compare chemodiverse samples, *i.e.* with a poor features overlap, or to compare samples with a strong RT shift, acquired using different LC methods or even different mass spectrometers technology (MaXis Q-ToF vs Q-Exactive).

MEMO is mainly built on matchms and spec2vec packages for handling the MS2 spectra and converting them into documents. Huge thanks to them for the amazing work done with these packages!

CHAPTER

THREE

TO INSTALL IT:

First, make sure to have anaconda installed.

1. Create a new conda environment to avoid clashes:

```
conda create --name memo python=3.8
conda activate memo
```

2. Install with pip:

```
pip install numpy
pip install memo-ms
```

If you have an error, try installing scikit-bio from conda-forge before installing the package with pip:

```
conda install -c conda-forge scikit-bio
pip install memo-ms
```

You can clone the Github package repository to get the demo files and the tutorial!

CHAPTER FOUR

EXAMPLES

Different examples of application and comparison to other MS/MS-based metrics are available here and the corresponding notebooks are available on GitHub.

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX