

---

# **EM2Tools**

***Release 0.2.0***

**Frédéric PLEWNIAK**

**Sep 24, 2020**



**CONTENTS:**

**1   Getting started** **1**

    1.1   Requirements . . . . . 1

    1.2   Installation . . . . . 1

**2   The EM2Tools API reference** **3**

    2.1   The “utils” module . . . . . 3

    2.2   The “argparse\_em2” module . . . . . 3

    2.3   The “seq\_record” module . . . . . 4

    2.4   The “seq” module . . . . . 7

    2.5   The “seq\_feature” module . . . . . 8

    2.6   The “seq\_utils” module . . . . . 11

    2.7   The “table” module . . . . . 13

**3   Indices and tables** **19**

**Python Module Index** **21**

**Index** **23**



## GETTING STARTED

### 1.1 Requirements

The EM2Tools package requires at least Python 3.6 with Biopython, pandas and gffpandas packages.

### 1.2 Installation

#### 1.2.1 Linux

1. Download and unzip the master branch zip file:

```
wget https://github.com/fplewniak/EM2Tools/archive/master.zip
unzip master.zip
```

2. Install the EM2Tools Python package and the required Python packages with pip:

```
pip install EM2Tools-master/.
```



## THE EM2TOOLS API REFERENCE

### 2.1 The “utils” module

Basic utilities that can be shared between different modules of the API

`em2lib.utils.norm_sum_to_one(data)`

Normalization of values in a 1-D array so they sum to one

**Parameters** `data` – the 1-D array to normalize

**Returns** the normalized array

`em2lib.utils.norm_max(data)`

Normalization of values in a 1-D array relative to the max value

**Parameters** `data` – the 1-D array to normalize

**Returns** the normalized array

`em2lib.utils.norm_min_max(data)`

Normalization of values in a 1-D array to make them fall between 0 (min value) and 1 (max value)

**Parameters** `data` – the 1-D array to normalize

**Returns** the normalized array

`em2lib.utils.zscore(data)`

Computes the Z-score of values in a 1-D array

**Parameters** `data` – the 1-D array to normalize

**Returns** the normalized array

### 2.2 The “argparse\_em2” module

Extension module to the standard argparse module. Adding custom actions and argument verification methods.

**class** `em2lib.argparse_em2.GetList(option_strings, dest, nargs='+', **kwargs)`

An argparse custom action to return a list from an argument containing a list of elements and/or file names. Files are supposed to contain one element of the list per line. There can be more than one file and the argument may take a combination of elements and files. In all cases, the returned list will contain all the specified elements without any checking for redundancy. If you need a non redundant set instead of a list, then use GetSet action instead.

**static** `arg2list` (*values*)

This method converts the argument values containing elements and/or files containing elements into a list of elements.

**Parameters** **values** – argument values, this is supposed to be a list of arguments or None (returns an empty list)

**Returns** the list of elements or an empty list if the argument was None

**class** `em2lib.parse_em2.GetSet` (*option\_strings, dest, nargs='+', \*\*kwargs*)

An argparse custom action to return a set from an argument containing a list of elements and/or file names. Files are supposed to contain one element of the list per line. There can be more than one file and the argument may take a combination of elements and files. In all cases, the returned set will contain all the specified elements keeping only one copy of each element. If you do not want to remove redundancy, then use GetList action instead.

**static** `arg2set` (*values*)

This method converts the argument values containing elements and/or files containing elements into a set of elements.

**Parameters** **values** – argument values, this is supposed to be a list of arguments or None (returns an empty set)

**Returns** the set of elements or an empty set if the argument was None

## 2.3 The “seq\_record” module

Extension module to the Biopython Bio.SeqRecord module

**class** `em2lib.seq_record.SeqRecordEM2` (*seq, \*\*kwargs*)

Extension to Biopython SeqRecord class

**add\_feature** (*\*\*kwargs*)

Adds a feature to the current record according to arguments passed as *\*\*kwargs*.

**Parameters** **kwargs** – keyword arguments to pass to SeqFeatureEM2 class

**overlap** (*start, end=None, strand=0*)

Retrieves features that overlap a given position range.

**Parameters**

- **strand** – strand specification of features to be returned. If strand is 0, then features on both strands are returned. If feature.strand is 0, then all strands will match.
- **start** – start of range
- **end** – end of range, if None, then end=start

**Returns** a list of overlapping features

**feature\_after** (*position, strand=0, nearest=False*)

Retrieves the features immediately after (but not overlapping) the specified position, on one strand or both. If nearest is True, then only the nearest ones are returned.

**Parameters**

- **nearest** – if True, only the nearest features are returned. This only makes sense when strand is 0
- **strand** – strand specification of features to be returned. If strand is 0, then features on both strands are returned



- **position** – the position

**Returns** a list of features after the specified position

**feature\_before** (*position, strand=0, nearest=False*)

Retrieves the features immediately before (but not overlapping) the specified position, on one strand or both. If nearest is True, then only the nearest ones are returned.

**Parameters**

- **nearest** – if True, only the nearest features are returned. This only makes sense when strand is 0
- **strand** – strand specification of features to be returned. If strand is 0, then features on both strands are returned
- **position** – the position

**Returns** a list of features before the specified position

**surrounding\_features** (*position, strand=0, nearest=False*)

Retrieves all the features around a given position but not overlapping it. If nearest is True, then only the nearest features are returned.

**Parameters**

- **position** – the position
- **nearest** – if True, only the nearest features are returned.
- **strand** – strand specification of features to be returned. If strand is 0, then features on both strands are returned

**Returns** a list of features around the specified position

**join** (*other=None, offset=0, keepself=True*)

Joins two SeqRecordEM2 objects into a new one representing the resulting merged sequence

**Parameters**

- **keepself** – if True and overlapping subsequences are different, then keep sequence from self record, otherwise keep the sequence of other record.
- **other** – the other SeqRecordEM2 object
- **offset** – the offset of the two sequences. If the value is negative, then the two sequences overlap.

**Returns** the result of merging records as a new SeqRecordEM2 object

**stitch** (*other, fpos\_in\_self, fpos\_in\_other, feature\_length, orientation=1, \*\*kwargs*)

Stitches two records, that is, joins them according to an overlapping feature. The sequences may or may not overlap. If not, Ns or Xs are added to fill the gap. If they overlap, a warning is issued if sequences do not correspond exactly. The new record keeps track of the two original records as Features. By convention, the self record should contain the start position of the feature on the forward strand, the other contains the end position on the forward strand if orientation=1 or on the reverse strand if orientation=-1. If orientation is -1, then the other record is reversed/complemented before stitching and the position of the overlapping feature is modified accordingly. It is the user's responsibility to provide the records in the right order.

**Parameters**

- **other** – the other SeqRecordEM2 object to stitch
- **fpos\_in\_self** – feature position in self record (start position of feature)

- **fpos\_in\_other** – feature position in other record (end position of feature), according to FeatureLocation conventions for end position requiring that length = end - start, it is not included in the feature.
- **feature\_length** – feature length
- **orientation** – the orientation of the other record relative to the self, either 1 if it is in the same orientation, -1 if other needs to be reversed before stitching, 0 if stranded but unknown, None for proteins
- **kwargs** – any additional parameters that may be passed to the constructor of the stitching feature in the new record

**Returns** the stitched record as a new SeqRecordEM2 object

**reverse\_complement** (*id=False, name=False, description=False, features=True, annotations=False, letter\_annotations=True, dbxrefs=False*)

Reverse-complement the record adjusting features and their positions accordingly. The record id is conserved but if name is not specified 'reversed' is appended. All other arguments are passed and handled by the parent method. Note that the main goal for this method is to replace SeqRecord and Seq objects by their SeqRecordEM2 and SeqEM2 equivalents when reverse/complementing.

#### Parameters

- **id** – the id for the reversed record
- **name** – the name for the reversed record
- **description** – the description for the reversed record
- **features** – keep and adjust location of features if True
- **annotations** – keep annotations if True
- **letter\_annotations** – keep letter\_annotations if True
- **dbxrefs** – keep dbxrefs if True

**Returns** a reversed copy of the record

**orfs\_to\_features** (*start=['ATG'], stop=None, filter=None, add=False*)

Determines all open reading frames in a sequence record. All the returned ORFs have a length that is a multiple of 3. Thus, for sequences without any stop codon, 3 ORFs are returned, one for each frame. Both strands are examined but it is possible to filter the ORFs by length, frame, etc. with the FeatureFilter defined by the filter argument.

#### Parameters

- **start** – a list of accepted start codons
- **stop** – a list of accepted stop codons
- **filter** – a FeatureFilter object defining a filter to select ORFs according to some criteria
- **add** – if True, the selected ORFs are added to the record's features

**Returns** a list of ORFs as SeqFeatureEM2 objects

## 2.4 The “seq” module

Extension of Bio.Seq.Seq class from Biopython to add or improve functionalities

**class** `em2lib.seq.SeqEM2` (*data*, *seqtype*)

SeqEM2 class providing extension to Bio.Seq.Seq class of BioPython package.

**classmethod** `dna` (*data*)

Creates a DNA sequence

**Parameters** *data* – the sequence string

**Returns** a SeqEM2 DNA instance

**classmethod** `protein` (*data*)

Creates a protein sequence

**Parameters** *data* – the sequence string

**Returns** a SeqEM2 protein instance

**is\_protein** ()

Tests whether sequence was created as a protein

**Returns** boolean, True if sequence was created as a protein, False otherwise.

**length\_in\_range** (*minlength=None*, *maxlength=None*)

Checks whether the sequence length is with the specified range.

**Parameters**

- **minlength** – lower length bound
- **maxlength** – upper length bound

**Returns** True if sequence length is within specified range, False otherwise

**re\_search** (*regex*)

Searches a sequence using a regular expression

**Parameters** *regex* – the regular expression

**Returns** a list of re.Match instances

**search** (*pattern*)

Searches sequence for a pattern specified with a fuzznuc or fuzzpro like syntax

**Parameters** *pattern* – the pattern to be searched for that is converted into a regular expression

**Returns** a list of re.Match objects

**get\_orfs** (*start=['ATG']*, *stop=None*)

Determines all open reading frames in a sequence. It only examines the forward strand. All the returned ORFs have a length that is a multiple of 3. Thus, for sequences without any stop codon, 3 ORFs are returned, one for each frame.

**Parameters**

- **start** – list of accepted start codon or None if ORFs do not need to start at a start codon.
- **stop** – list of accepted stop codons

**Returns** a set of tuples (start, end) of orfs where start is the starting position of the orf and end its ending position, not including the stop codon

## 2.5 The “seq\_feature” module

Extension of Bio.SeqFeature module from Biopython to add or improve functionalities

**class** `em2lib.seq_feature.SeqFeatureEM2` (*parent=None, \*\*kwargs*)  
SeqFeatureEM2 class providing extension to Bio.SeqFeature.SeqFeature class of BioPython package.

**length\_in\_range** (*minlength=None, maxlength=None*)  
Checks whether the feature length is with the specified range.

**Parameters**

- **minlength** – lower length bound
- **maxlength** – upper length bound

**Returns** True if feature length is within specified range, False otherwise

**lies\_within** (*start, end*)  
Determines whether feature lies entirely with the specified range. Fuzzy positions are turned into integers.

**Parameters**

- **start** – start of range either int or ExactPosition
- **end** – end of range either int or ExactPosition

**Returns** True if feature boundaries lie with the specified range.

**covers** (*start, end*)  
Determines whether feature covers the whole range specified by start and end

**Parameters**

- **start** – start of range either int or ExactPosition
- **end** – end of range either int or ExactPosition, if None then end=start

**Returns** True if feature covers the specified range

**overlaps** (*start, end=None*)  
Determines whether feature overlaps a position range.

**Parameters**

- **start** – start of range either int or ExactPosition
- **end** – end of range either int or ExactPosition

**Returns** True if feature overlaps range

**intersect** (*other, \*\*kwargs*)  
Creates a new feature which is the intersection of feature and another one

**Parameters** **other** – the other feature

**move** (*offset*)  
Moves a feature by a certain offset

**Parameters** **offset** – offset by which the feature must be moved

**class** `em2lib.seq_feature.FeatureFilter`  
A class for the definition and application of a filter to a list of features. May be used to retrieve features from a record or any list of features according to length, location, type, strand and frame. It is possible to combine several criteria into one single filter.

**keep** (*keep=True*)

Set the keep attribute of the filter. If True, the features corresponding to the specified criteria will be kept, otherwise, they will be discarded.

**Parameters** **keep** – boolean, True if features consistent with the criteria should be kept.

**Returns** the current filter

**type** (*feat\_type=None*)

Set the type of feature.

**Parameters** **feat\_type** – the type of feature to select

**Returns** the current filter

**length** (*minlength=None, maxlength=None*)

Set the length range of the feature.

**Parameters**

- **minlength** – the minimum length of the feature or 0 if None
- **maxlength** – the maximum length of the feature or no limit if set to None

**Returns** the current filter

**strand** (*strand=0*)

Set the strand of the feature.

**Parameters** **strand** – the strand of the feature

**Returns** the current filter

**frame** (*frame=0, strand=1*)

Set the frame of the filter. By default, this is on the forward strand unless the strand is set to -1.

**Parameters**

- **frame** – the frame 0, 1 or 2
- **strand** – the strand 1 or - 1

**Returns** the current filter

**covers** (*start=None, end=None*)

Set a region that must be covered by the feature.

**Parameters**

- **start** – start position of the region
- **end** – end position of the region

**Returns** the current filter

**overlaps** (*start=None, end=None*)

Set a region that must overlap the feature.

**Parameters**

- **start** – start position of the region
- **end** – end position of the region

**Returns** the current filter

**lies\_within** (*start=None, end=None*)

Set a region within which the feature must lie.

**Parameters**

- **start** – start position of the region
- **end** – end position of the region

**Returns** the current filter

**type\_applies** (*feature*)

Test if type criterion applies to the feature and return a boolean stating whether the feature should be kept or rejected. If the criterion has not been set, then True is returned.

**Parameters** **feature** – the feature to test

**Returns** True the feature must be kept

**length\_applies** (*feature*)

Test if length criterion applies to the feature and return a boolean stating whether the feature should be kept or rejected. If the criterion has not been set, then True is returned.

**Parameters** **feature** – the feature to test

**Returns** True the feature must be kept

**covers\_applies** (*feature*)

Test if covers criterion applies to the feature and return a boolean stating whether the feature should be kept or rejected. If the criterion has not been set, then True is returned.

**Parameters** **feature** – the feature to test

**Returns** True the feature must be kept

**overlaps\_applies** (*feature*)

Test if overlaps criterion applies to the feature and return a boolean stating whether the feature should be kept or rejected. If the criterion has not been set, then True is returned.

**Parameters** **feature** – the feature to test

**Returns** True the feature must be kept

**lies\_within\_applies** (*feature*)

Test if lies\_within criterion applies to the feature and return a boolean stating the feature should be kept or rejected. If the criterion has not been set, then True is returned.

**Parameters** **feature** – the feature to test

**Returns** True the feature must be kept

**location\_applies** (*feature*)

Test if location criterion applies to the feature and return a boolean stating whether the feature should be kept or rejected. If the criterion has not been set, then True is returned.

**Parameters** **feature** – the feature to test

**Returns** True the feature must be kept

**strand\_applies** (*feature*)

Test if strand criterion applies to the feature and return a boolean stating whether the feature should be kept or rejected. If the criterion has not been set, then True is returned.

**Parameters** **feature** – the feature to test

**Returns** True the feature must be kept

**frame\_applies** (*feature*)

Test if frame criterion applies to the feature and return a boolean stating whether the feature should be kept or rejected. If the criterion has not been set, then True is returned.

**Parameters** **feature** – the feature to test

**Returns** True the feature must be kept

**apply** (*features*)

Test if all defined criteria apply to the features in the list and return the list of features corresponding to the specified criteria.

**Parameters** **features** – the list of features to filter

**Returns** the filtered list of features

## 2.6 The “seq\_utils” module

Extension module to the Biopython Bio.SeqUtils module

`em2lib.seq_utils.ambiguous2string` (*code*, *protein=False*)

Converts an ambiguous residue into a string with all compatible unambiguous residues. If the input code is not ambiguous, it is returned without any conversion.

**Parameters**

- **code** – the input code to be converted into a list of residues.
- **protein** – True if residue is amino-acid

**Returns** a string corresponding to the unambiguous residues compatible with the input code

`em2lib.seq_utils.isambiguous` (*code*, *protein=False*)

Checks code is an ambiguous residue specification or not.

**Parameters**

- **code** – the input code that must be checked for ambiguity
- **protein** – True if code is amino-acid code

**Returns** boolean, True if code is ambiguous, False otherwise

`em2lib.seq_utils.pattern2regex` (*pattern*, *protein=False*)

Converts a fuzznuc or fuzzpro-like pattern into a regular expression that can be used to search a sequence string.

- [ABC] => any of ABC residues,
- {ABC} => any residue except ABC,
- <ABC... => start of sequence,
- ...ABC> => end of sequence,
- A(n)(ABC)(n) => repeat residue or subsequence n times,
- A(n,m)(ABC)(n,m) => repeat residue or subsequence from n up to m times.

**Parameters**

- **pattern** – the pattern definition (string)
- **protein** – True if pattern applies to a protein sequence, False otherwise.

**Returns** the regular expression pattern as a string

**class** `em2lib.seq_utils.SeqFilter`

A class for the creation of a sequence filter to specify filtering criteria and applying the filter to a list of sequence records.

**length** (*minlength=None, maxlength=None*)

Minimal and maximal length specification

**Parameters**

- **minlength** – minimal accepted length
- **maxlength** – maximal accepted length

**Returns** SeqFilter instance

**pattern** (*pattern=None*)

pattern specification

**Parameters** **pattern** – pattern that must be in the sequence

**Returns** SeqFilter instance

**name** (*name=None*)

sequence record name specification

**Parameters** **name** – name regular expression

**Returns** SeqFilter instance

**keep** (*keep=True*)

Boolean defining whether the matching sequences must be kept (True) or removed (False)

**Parameters** **keep** – True to keep positive sequences, False to remove them

**Returns** SeqFilter instance

**length\_applies** (*rec*)

test whether length criterion applies to the sequence record

**Parameters** **rec** – the sequence record to test

**Returns** boolean True if criterion applies or False otherwise

**pattern\_applies** (*rec*)

test whether parameter criterion applies to the sequence record

**Parameters** **rec** – the sequence record to test

**Returns** boolean True if criterion applies or False otherwise

**name\_applies** (*rec*)

test whether name criterion applies to the sequence record

**Parameters** **rec** – the sequence record to test

**Returns** boolean True if criterion applies or False otherwise

**apply** (*records*)

Filters a list of SeqRecords instances, keeping only records satisfying the specified criteria of length, match of a pattern, name specification. It is possible to invert the filtering process by setting the keep boolean to False and thus only keep records which do not satisfy the criteria.

**Parameters** **records** – list of SeqRecord instances to apply

**Returns** the filtered list of records



**class** `em2lib.seq_utils.GFF` (*feature\_list=None, input\_df=None*)  
 Manipulation of features based upon gffpandas package

**add\_feature\_list** (*feature\_list=None*)  
 Adds a list of feature to the list of an existing GFF object

**Parameters** `feature_list` – list of features to add to DataFrame

**Returns** the GFF object with feature list appended

**static df\_from\_feature** (*feature*)  
 Create a pandas DataFrame from a feature (SeqFeatureEM2 or SeqFeature)

**Parameters** `feature` – the feature to convert into a dataframe

**Returns** the resulting dataframe

**to\_feature\_list** (*parents=None*)  
 Converts features in a GFF object into a list of SeqFeatureEM2 objects

**Parameters** `parents` – list of references to parent SeqRecord objects or a single parent reference if all features are defined in the same parent. If it is a list, it should be of the same length as the dataframe, repeating references as needed to get the right number.

**Returns** a list of SeqFeatureEM2 objects

## 2.7 The “table” module

Some utilities for Table manipulation, comparison, etc. using pandas.DataFrame module.

`em2lib.table.get_common_keys` (*first, other, keys\_first=None, keys\_other=None*)  
 Return the index keys in common between first DataFrame and other DataFrame

**Parameters**

- **first** – the first DataFrame object to compare
- **other** – the other DataFrame object to compare
- **keys\_first** – a list of references to the columns defining indices in first DataFrame. None refers to the original index
- **keys\_other** – a list of references to the columns defining indices in other DataFrame. None refers to the original index

**Returns** DataFrame, the keys that are common between the two DataFrames.

`em2lib.table.get_common_rows` (*first, other, keys\_first=None, keys\_other=None, lsuffix="", rsuffix="", drop=True*)

Return rows with the same key values. Key columns from the other DataFrame object are dropped by default but can be kept if drop is set to False.

**Parameters**

- **first** – the first DataFrame object to compare
- **other** – the other DataFrame object to compare
- **keys\_first** – a list of references to the columns defining keys in first DataFrame. None refers to the original index
- **keys\_other** – a list of references to the columns defining keys in other DataFrame. None refers to the original index

- **lsuffix** – str, Suffix to add to first column names in case of redundancy.
- **rsuffix** – str, Suffix to add to right column names in case of redundancy.
- **drop** – True by default. Drop the other key columns in the resulting DataFrame.

**Returns** DataFrame with the same keys as first DataFrame, containing the rows that have the same key values between the two DataFrames.

`em2lib.table.get_keys_not_in (first, other, keys_first=None, keys_other=None)`

Return the keys in first DataFrame which are not in other DataFrame

#### Parameters

- **first** – the first DataFrame object to compare
- **other** – the other DataFrame object to compare
- **keys\_first** – a list of references to the columns defining keys in first DataFrame. None refers to the original index
- **keys\_other** – a list of references to the columns defining keys in other DataFrame. None refers to the original index

**Returns** DataFrame, the keys that are in first DataFrame but not in the other one.

`em2lib.table.get_rows_not_in (first, other, keys_first=None, keys_other=None)`

Return a DataFrame with rows with keys that are in first but not in other.

#### Parameters

- **first** – the first DataFrame object to compare
- **other** – the other DataFrame object to compare
- **keys\_first** – a list of references to the columns defining keys in first DataFrame. None refers to the original index
- **keys\_other** – a list of references to the columns defining keys in other DataFrame. None refers to the original index

**Returns** DataFrame, the rows whose keys are in first DataFrame but not in the other one.

`em2lib.table.statistics (table, groupby=None, columns=None, func=<function mean>)`

Returns a DataFrame containing the requested statistics on the specified columns in a table, optionally grouping data according to one or several other columns.

#### Parameters

- **table** – the table to compute the statistics
- **groupby** – the column or list of columns for grouping data
- **columns** – the column or list of columns to compute the statistics about
- **func** – the statistical function or list thereof to be computed

**Returns** a DataFrame with the requested statistics

`em2lib.table.implode (input_df, index=None)`

Reverts action of DataFrame.explode method. This method can also be applied to DataFrames that are not the result of an explode() call but in this case, applying back the explode() method may not yield the original DataFrame.

#### Parameters

- **input\_df** – the input DataFrame

- **index** – the index column(s)

**Returns** the resulting DataFrame

`em2lib.table.collapse(input_df, groupby=None, columns=None, name=None)`

Collapses a DataFrame into a single column containing lists of tuples representing all the values in the specified columns with the same index as defined by groupby. Note: results may be inconsistent if input DataFrame contains elements that are lists or tuples.

#### Parameters

- **input\_df** – the input DataFrame
- **groupby** – the column(s) defining the index for collapsing values
- **columns** – the columns that will be collapsed
- **name** – the name for the resulting column

**Returns** a single-column collapsed DataFrame

`em2lib.table.expand(input_df, columns=None)`

Expands a collapsed table. Exact reverse of collapse() function.

#### Parameters

- **input\_df** – the collapsed table to expand
- **columns** – names of the columns, if None, will attempt to deduce them from the collapsed column name expected to be a tuple of the original names as produced by collapse(). Note that if less column names are specified than the actual number of columns to expand then extra column names will be added automatically.

**Returns** the expanded table

`em2lib.table.table_from_edges(df_edges: pandas.core.frame.DataFrame, graph=True, directed=True, link=None, no_link=None)`

Turns a DataFrame representing edges in two or three columns into a table representing edges between nodes in rows and columns. The first two columns of the input DataFrame represent the linked nodes. The optional third column contains a value (edge weight, label, etc.).

A call to `table_from_edges(df_edges, no_link=0, link=1, graph=False)` is basically equivalent to a call to `pandas.crosstab(df_edges[0], df_edges[1])`.

A call to `table_from_edges(df_edges, graph=False)` is basically equivalent to a call to `df_edges.pivot(0,1,2)`.

#### Parameters

- **df\_edges** – the edge DataFrame,
- **graph** – if True, the edge list represents a graph and the resulting table will be a square: all nodes in the first two columns of the edge DataFrame are placed in both the index and the columns of the graph table. If False, the edge list represents a relation between two sets. The first column of the edge DataFrame will define rows (first set) and the second column will define the columns (second set).
- **directed** – specifies if the graph is directed or not. If True, the resulting table will be symmetrical. This parameter has no effect if graph is False
- **link** – the value to associate the specified edges with. This value will replace the original value in df\_edges DataFrame or specify one for two-column edge DataFrames. This value is needed for two-column edge definition.
- **no\_link** – the value to associate the unspecified edges with. NaN will be used if this parameter is not specified.

**Returns** a table DataFrame

`em2lib.table.edges_from_table(df_table: pandas.core.frame.DataFrame, no_link=None, directed=True)`

Converts a table into a DataFrame with list of edges. This method is the reverse of `table_from_edges()`.

**Parameters**

- **df\_table** – the table to be converted
- **no\_link** – the value indicating that there is no link between nodes
- **directed** – specifies whether the graph denoted by `df_table` is directed or not.

**Returns** the edge DataFrame

**class** `em2lib.table.TableTransform(df)`

Class of objects to transform a DataFrame according to different criteria. Different independent transformations can be chained since all transformation methods return self object.

**result()**

Getter method to retrieve the DataFrame resulting from the transformation

**Returns** the transformed DataFrame

**update()**

Update original DataFrame with the current working copy. This enables the application of conditions to previously modified values. This action cannot be undone, so it is recommended to use it after all modifications based on original data have been performed.

**cond\_transform**(*cond=<function TableTransform.<lambda>>, iftrue=<function TableTransform.<lambda>>, columns=None, original=True*)

Conditional transform of a DataFrame. If condition function returns True, then the iftrue function is applied to transform the corresponding element.

**Parameters**

- **cond** – string, function, DataFrame or Series A string considered as a test function on the columns of the DataFrame working copy. This string is passed to `self.wrkg_df.eval()` for evaluation to produce a mask DataFrame. A function is applied to elements of the original DataFrame in order to define which elements in the working DataFrame should be transformed. A DataFrame is a mask of bool values with the same shape as the original DataFrame. Elements in the working A Series is a mask of bool values defining in which rows the transformation should be applied.
- **iftrue** – string, function or a DataFrame with the same shape as the original table. The function to apply if condition is True, returns a single value from a single value. This function should test whether it is applicable to its input and return the original value if not. If a DataFrame, it defines the values that will replace those where the condition is True. If a string, it is passed to `self.wrkg_df.eval()` for evaluation and is used as a DataFrame.
- **columns** – str or list thereof specifying the column(s) which the transformation should be applied to
- **original** – if True, the original DataFrame is used to assess the condition function, otherwise, the working copy is used. This has an effect only if `cond` is a function and therefore only works for elementwise transformation. For rowwise selection, `cond` should be a DataFrame that can be computed by a function taking as input the current result DataFrame `self.result()`

**Returns** this TableTransform instance

**combine** (*other, func, columns=None, \*\*kwargs*)

A method wrapping the DataFrame.combine() method and adding the columns parameter to apply the method only to the specified columns.

#### Parameters

- **other** – the other DataFrame to combine to the current working DataFrame
- **func** – function that takes two series as inputs and returns a Series or a scalar to merge the two dataframes column by column.
- **columns** – str or list thereof specifying the column(s) which should be kept in the final result.
- **kwargs** – named arguments to pass to DataFrame.replace() method.

**Returns** this TableTransform instance

**normalize** (*columns=None, norm=<function norm\_sum\_to\_one>, by=None*)

Normalize values in DataFrame, either by row, column or over all the values.

#### Parameters

- **columns** – the columns whose values should be normalized
- **norm** – the normalization function, should apply to a 1-D array, default is “norm\_sum\_to\_one”
- **by** – defines whether the normalization should be done by column or by row. If None, then all values are used.

**Returns** this TableTransform instance

**replace** (*columns=None, \*\*kwargs*)

A method wrapping the DataFrame.replace() method and adding the columns parameter to apply the replacement only in the specified columns.

#### Parameters

- **columns** – str or list thereof specifying the column(s) to apply replacement to
- **kwargs** – named arguments to pass to DataFrame.replace() method. Note that inplace argument is deactivated as it is not needed here and might actually interfere with the columns argument. As a matter of fact, the working DataFrame is always modified.

**Returns** this TableTransform instance

**randomize** (*by=None, replacement=False, columns=None, seed=None*)

Randomize a DataFrame by row, column or all elements across the whole table with or without replacement. If by=row (or column), then elements of each row (or each column) are randomized within the row (or the column) Otherwise, all elements of the DataFrame are resampled across the whole table independently of rows or columns.

#### Parameters

- **by** – specifies whether randomization should be performed by row, by column or by element (None)
- **replacement** – if True, then randomization will occur with replacement.
- **columns** – str or list thereof specifying the column(s) which the columns affected by the randomization
- **seed** – seed for random numbers generation passed to sample() method as the random\_state argument

**Returns** the current TableTransform object

## INDICES AND TABLES

- `genindex`
- `modindex`
- `search`





## PYTHON MODULE INDEX

### e

- `em2lib`, 3
- `em2lib.argparse_em2`, 3
- `em2lib.seq`, 7
- `em2lib.seq_feature`, 8
- `em2lib.seq_record`, 4
- `em2lib.seq_utils`, 11
- `em2lib.table`, 13
- `em2lib.utils`, 3



## A

add\_feature() (*em2lib.seq\_record.SeqRecordEM2 method*), 4  
 add\_feature\_list() (*em2lib.seq\_utils.GFF method*), 13  
 ambiguous2string() (*in module em2lib.seq\_utils*), 11  
 apply() (*em2lib.seq\_feature.FeatureFilter method*), 11  
 apply() (*em2lib.seq\_utils.SeqFilter method*), 12  
 arg2list() (*em2lib.argparse\_em2.GetList static method*), 3  
 arg2set() (*em2lib.argparse\_em2.GetSet static method*), 4

## C

collapse() (*in module em2lib.table*), 15  
 combine() (*em2lib.table.TableTransform method*), 16  
 cond\_transform() (*em2lib.table.TableTransform method*), 16  
 covers() (*em2lib.seq\_feature.FeatureFilter method*), 9  
 covers() (*em2lib.seq\_feature.SeqFeatureEM2 method*), 8  
 covers\_applies() (*em2lib.seq\_feature.FeatureFilter method*), 10

## D

df\_from\_feature() (*em2lib.seq\_utils.GFF static method*), 13  
 dna() (*em2lib.seq.SeqEM2 class method*), 7

## E

edges\_from\_table() (*in module em2lib.table*), 16  
 em2lib  
   module, 3  
 em2lib.argparse\_em2  
   module, 3  
 em2lib.seq  
   module, 7  
 em2lib.seq\_feature  
   module, 8  
 em2lib.seq\_record  
   module, 4

em2lib.seq\_utils  
   module, 11  
 em2lib.table  
   module, 13  
 em2lib.utils  
   module, 3  
 expand() (*in module em2lib.table*), 15

## F

feature\_after() (*em2lib.seq\_record.SeqRecordEM2 method*), 4  
 feature\_before() (*em2lib.seq\_record.SeqRecordEM2 method*), 5  
 FeatureFilter (*class in em2lib.seq\_feature*), 8  
 frame() (*em2lib.seq\_feature.FeatureFilter method*), 9  
 frame\_applies() (*em2lib.seq\_feature.FeatureFilter method*), 10

## G

get\_common\_keys() (*in module em2lib.table*), 13  
 get\_common\_rows() (*in module em2lib.table*), 13  
 get\_keys\_not\_in() (*in module em2lib.table*), 14  
 get\_orfs() (*em2lib.seq.SeqEM2 method*), 7  
 get\_rows\_not\_in() (*in module em2lib.table*), 14  
 GetList (*class in em2lib.argparse\_em2*), 3  
 GetSet (*class in em2lib.argparse\_em2*), 4  
 GFF (*class in em2lib.seq\_utils*), 12

## I

implode() (*in module em2lib.table*), 14  
 intersect() (*em2lib.seq\_feature.SeqFeatureEM2 method*), 8  
 is\_protein() (*em2lib.seq.SeqEM2 method*), 7  
 isambiguous() (*in module em2lib.seq\_utils*), 11

## J

join() (*em2lib.seq\_record.SeqRecordEM2 method*), 5

## K

keep() (*em2lib.seq\_feature.FeatureFilter method*), 8  
 keep() (*em2lib.seq\_utils.SeqFilter method*), 12

**L**

`length()` (*em2lib.seq\_feature.FeatureFilter* method), 9  
`length()` (*em2lib.seq\_utils.SeqFilter* method), 12  
`length_applies()` (*em2lib.seq\_feature.FeatureFilter* method), 10  
`length_applies()` (*em2lib.seq\_utils.SeqFilter* method), 12  
`length_in_range()` (*em2lib.seq.SeqEM2* method), 7  
`length_in_range()` (*em2lib.seq\_feature.SeqFeatureEM2* method), 8  
`lies_within()` (*em2lib.seq\_feature.FeatureFilter* method), 9  
`lies_within()` (*em2lib.seq\_feature.SeqFeatureEM2* method), 8  
`lies_within_applies()` (*em2lib.seq\_feature.FeatureFilter* method), 10  
`location_applies()` (*em2lib.seq\_feature.FeatureFilter* method), 10

**M**

`module`  
    *em2lib*, 3  
    *em2lib.argparse\_em2*, 3  
    *em2lib.seq*, 7  
    *em2lib.seq\_feature*, 8  
    *em2lib.seq\_record*, 4  
    *em2lib.seq\_utils*, 11  
    *em2lib.table*, 13  
    *em2lib.utils*, 3  
`move()` (*em2lib.seq\_feature.SeqFeatureEM2* method), 8

**N**

`name()` (*em2lib.seq\_utils.SeqFilter* method), 12  
`name_applies()` (*em2lib.seq\_utils.SeqFilter* method), 12  
`norm_max()` (in module *em2lib.utils*), 3  
`norm_min_max()` (in module *em2lib.utils*), 3  
`norm_sum_to_one()` (in module *em2lib.utils*), 3  
`normalize()` (*em2lib.table.TableTransform* method), 17

**O**

`orfs_to_features()` (*em2lib.seq\_record.SeqRecordEM2* method), 6  
`overlap()` (*em2lib.seq\_record.SeqRecordEM2* method), 4  
`overlaps()` (*em2lib.seq\_feature.FeatureFilter* method), 9  
`overlaps()` (*em2lib.seq\_feature.SeqFeatureEM2* method), 8

`overlaps_applies()` (*em2lib.seq\_feature.FeatureFilter* method), 10

**P**

`pattern()` (*em2lib.seq\_utils.SeqFilter* method), 12  
`pattern2regex()` (in module *em2lib.seq\_utils*), 11  
`pattern_applies()` (*em2lib.seq\_utils.SeqFilter* method), 12  
`protein()` (*em2lib.seq.SeqEM2* class method), 7

**R**

`randomize()` (*em2lib.table.TableTransform* method), 17  
`re_search()` (*em2lib.seq.SeqEM2* method), 7  
`replace()` (*em2lib.table.TableTransform* method), 17  
`result()` (*em2lib.table.TableTransform* method), 16  
`reverse_complement()` (*em2lib.seq\_record.SeqRecordEM2* method), 6

**S**

`search()` (*em2lib.seq.SeqEM2* method), 7  
*SeqEM2* (class in *em2lib.seq*), 7  
*SeqFeatureEM2* (class in *em2lib.seq\_feature*), 8  
*SeqFilter* (class in *em2lib.seq\_utils*), 12  
*SeqRecordEM2* (class in *em2lib.seq\_record*), 4  
`statistics()` (in module *em2lib.table*), 14  
`stitch()` (*em2lib.seq\_record.SeqRecordEM2* method), 5  
`strand()` (*em2lib.seq\_feature.FeatureFilter* method), 9  
`strand_applies()` (*em2lib.seq\_feature.FeatureFilter* method), 10  
`surrounding_features()` (*em2lib.seq\_record.SeqRecordEM2* method), 5

**T**

`table_from_edges()` (in module *em2lib.table*), 15  
*TableTransform* (class in *em2lib.table*), 16  
`to_feature_list()` (*em2lib.seq\_utils.GFF* method), 13  
`type()` (*em2lib.seq\_feature.FeatureFilter* method), 9  
`type_applies()` (*em2lib.seq\_feature.FeatureFilter* method), 10

**U**

`update()` (*em2lib.table.TableTransform* method), 16

**Z**

`zscore()` (in module *em2lib.utils*), 3