RCMAP

Release 1.4.0

Laura Boeglin & Frédéric Plewniak

Jun 18, 2020

CONTENTS:

Residue Conservation in Multiple Alignment of Proteins (RCMAP) is a Python package to help manual annotation of protein sequences by comparing them to a multiple alignment of reference sequences belonging to a functional family.

The RCMAP package provides the shell command `evaluate_seq` whose input is a multiple alignment file in FastA format, containing reference sequences and one or more unknown sequences to annotate. It then displays for each unknown sequence whether it is consistent at every user-specified position with the aminoacid conservation profile of the reference sequences.

CHAPTER

ONE

GETTING STARTED

1.1 Installation

1.1.1 Linux

1. Download and unzip master branch zip file:

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

2. Install the RCMAP Python package and requirements with pip:

```
pip install RCMAP-master/.
```

1.1.2 Windows

1. Download and unzip master branch zip file.
2. Open the unzipped `RCMAP-master` directory in a new window
3. Open a Windows PowerShell in the `RCMAP-master` directory. (in Windows explorer, type `Ctrl-L` to select the address bar of the `RCMAP-master` window then type `powershell` and validate)
4. In the PowerShell window, install the RCMAP Python package and its requirements with pip:

```
pip install RCMAP-master/.
```

Note: python and pip must be in your path

1.2 Usage

The RCMAP package provides the `evaluate_seq` command:

```
usage: evaluate_seq [-h] --file File --seqeval string [string ...] [--positions␣
↪POSITIONS [POSITIONS ...]]

    required arguments:
      --file File           input multiple protein sequence alignment file containing␣
↪reference
                            sequences and sequences to evaluate
```

(continues on next page)

(continued from previous page)

```
      --seqeval string [string ...]
                            names of the sequences to evaluate

    optional arguments:
      -h, --help            show help message and exit
      --positions POSITIONS [POSITIONS ...]
                            list of position ranges to examine specified as start and␣
↪end positions
                            separated by a ':'. If the start position is not␣
↪specified, then the
                            range will start at 1. If the end position is not␣
↪specified, then the
                            range will end at the last position of the alignment.
                            If no range is specified the whole alignment will be␣
↪examined.
      --gaps                toggles accounting for gaps when computing information␣
↪content
      --strict              toggles strict evaluation, an aminoacid is considered␣
↪compatible only if it has been
                            in at least one of the reference sequences
      --min_info FLOAT      the minimum information content at a given position in␣
↪the reference alignment required
                            to display the position
      --method              calculation method of the background entropy for the␣
↪information
      --window              number of positions to calculate the average of␣
↪information, must be odd
      --window_method       calculation method of the weights of positions to␣
↪calculate the information of a position,
                            using a window
```

1.2.1 Examples

Basic example

```
evaluate_seq --file ArsM_aln.faa --seqeval WP_045226361.1 Q969Z2 --positions 200:210 -
↪-gaps

WP_045226361.1 :  200 : S :   True : 4.39 :   {'S'}    {'S': 6}
WP_045226361.1 :  201 : N :   True : 4.39 :   {'N'}    {'N': 6}
WP_045226361.1 :  202 : C :   True : 4.39 :   {'C'}    {'C': 6}
WP_045226361.1 :  203 : - :   True : 4.39 :   set()    {'-': 6}
WP_045226361.1 :  204 : - :   True : 4.39 :   set()    {'-': 6}
WP_045226361.1 :  205 : - :   True : 4.39 :   set()    {'-': 6}
WP_045226361.1 :  206 : - :   True : 4.39 :   set()    {'-': 6}
WP_045226361.1 :  207 : V :   True : 4.39 :   {'V'}    {'V': 6}
WP_045226361.1 :  208 : L :   True : 3.47 : Hydrophobic {'C': 4, 'I': 2}
WP_045226361.1 :  209 : N :   True : 4.39 :   {'N'}    {'N': 6}
WP_045226361.1 :  210 : L :   True : 4.39 :   {'L'}    {'L': 6}

 WP_045226361.1
 Number of True  11  : Information True  47.4
 Number of False  0  : Information False  0
```

(continues on next page)

(continued from previous page)

```
Q969Z2 :  200 : S :   True : 4.39 :   {'S'}    {'S': 6}
Q969Z2 :  201 : D :  False : 4.39 :   {'N'}    {'N': 6}
Q969Z2 :  202 : I :  False : 4.39 :   {'C'}    {'C': 6}
Q969Z2 :  203 : P :  False : 4.39 :   set()    {'-': 6}
Q969Z2 :  204 : F :  False : 4.39 :   set()    {'-': 6}
Q969Z2 :  205 : G :  False : 4.39 :   set()    {'-': 6}
Q969Z2 :  206 : K :  False : 4.39 :   set()    {'-': 6}
Q969Z2 :  207 : K :  False : 4.39 :   {'V'}    {'V': 6}
Q969Z2 :  208 : F :   True : 3.47 : Hydrophobic {'C': 4, 'I': 2}
Q969Z2 :  209 : K :  False : 4.39 :   {'N'}    {'N': 6}
Q969Z2 :  210 : L :   True : 4.39 :   {'L'}    {'L': 6}

 Q969Z2
 Number of True  3  : Information True  12.26
 Number of False  8  : Information False  35.14
```

All positions between 200 and 210 in the WP_045226361.1 sequence are consistent with the aminoacid observed in the reference sequences shown in the last two columns. On the other hand, a majority of positions are not compatible with the reference conservation profile in Q969Z2. Strictly conserved aminoacids at positions 201, 202 207 and 209 are not conserved in this sequence, and it has an insertion from 203 to 206.

Raw information content accounting for gaps

```
evaluate_seq --file ArsM_aln.faa --seqeval WP_045226361.1 Q969Z2 --positions 50:70␣
↪115:125 200:210 --gaps
```

Displays compatibility at positions from 50 to 70, 115 to 125 and 200 to 210 of sequences WP_045226361.1 and Q969Z2 with the reference alignment in ArsM_aln.faa. Gaps are taken into account when computing information content.

Smoothed information content without gaps

```
evaluate_seq --file ArsM_aln.faa --seqeval WP_045226361.1 Q969Z2 --positions :10 20␣
↪200: --window_method hamming --window 5
```

Displays compatibility at positions from 1 to 10, at 20 and 200 to end of sequences WP_045226361.1 and Q969Z2 with the reference alignment in ArsM_aln.faa. Gaps are not taken into account. Information content along the aligment is smoothed over a sliding window weighted using the Hamming method.

CHAPTER

TWO

THE RCMAP API REFERENCE

2.1 The "alignment" module

class RCMAP.alignment.**Alignments**(*file*, *seqs_to_evaluate*)

Opens the file and processes the alignment of sequences

count_aa_ref()

Counts the number of every amino acids at every position in the reference sequences, and lists them in a dictionary

Returns the count of amino acids at every position in all reference sequences

determine_ref_categories()

Recovers information about every position in the reference sequences

Returns the list of categories of amino acids at every position in the reference sequences, list of the names of the categories, the list of the amino acids observed at every position in reference sequences in lists, the list of the number of every amino acid present at the position for every position and the list of the amino acids observed at every position in reference sequences in sets

entropy_background(*method*, *gaps*)

Calculates the background entropy from the frequencies of the amino acids given by the method. Can take into account the gaps in the frequencies if gaps = True. There are three methods :

- database : the frequencies of the amino acids come from the bank UniprotKB,TrEMBL april 2020
- ref : the frequencies of the amino acids come from the average of the counts in the reference sequences
- equiprobable : the frequencies of the amino acids are all the same

NB : the entropy function can take in parameter the accounts of amino acids or frequencies directly

Parameters

- **method** – calculation method
- **gaps** – True if you want to consider gaps, False if not

Returns the background entropy in the reference alignment

entropy_pos_obs(*pos*)

Calculates the entropy from the frequencies of the amino acids observed at a position NB : the entropy function can take in parameter the accounts of amino acids or frequencies directly

Parameters **pos** – position in the alignment

Returns the entropy associated to the position

get_aa_at_pos(*pos*, *name_seq*)
Recovers the amino acid present at this position in the given sequence

Parameters

- **pos** – position of the amino acid in seqref or seqeval
- **name_seq** – name of the sequence in seqref or seqeval

Returns set() containing the amino acid at this position in the sequence

get_aa_observed_at_pos(*pos*)
Recovers the amino acids observed at a position in the reference sequences and the count of these amino acids. The data are in a dictionary sorted from the amino acid the most present to the least present at this position. Uses the position -1 because the user counts from 1 and python from 0

Parameters **pos** – position of the amino acids in seqrefs

Returns all the amino acids observed in seqrefs at this position and the count of this amino acids in a sorted dictionary

get_cat_at_pos(*pos*, *strict*)
Recovers the category (set of amino acids) of amino acids observed at a position in the reference sequences. Uses the position -1 because the user counts from 1 and python from 0

Parameters

- **strict** – if True, the category is only the amino acids observed
- **pos** – position of the amino acid in seqrefs

Returns the category (set) of amino acids observed in seqrefs

get_cat_name_at_pos(*pos*)
Recovers the name of the category of amino acids observed at a position in the reference sequences. Uses the position -1 because the user counts from 1 and python from 0

Parameters **pos** – position of the amino acid in seqrefs

Returns the name of the category of amino acids observed in seqrefs

information_pos(*pos*, *method*, *gaps*, *window*, *window_method*)
Calculates the information carried by a position. The running window can be used to consider the environment of a position and smooth its information.

Parameters

- **window_method** – calculation method of the weights at every position in the window
- **window** – number of positions to calculate the average of information, should be odd
- **pos** – position in the alignment
- **method** – calculation method of the frequencies in the background entropy
- **gaps** – True if you want to consider gaps, False if not

Returns the information of the position

2.2 The "classification_aa" module

class RCMAP.classification_aa.**AAcategories**

Defines the categories and determines in which category a set belongs to.

find_category(*ens*)

Finds the smallest category in which a set of amino acids (it could also be a unique amino acid) is included. Removes the gaps in the set and treats the amino acids 'B' and 'Z' which are ambiguous: 'B' represents the amino acids 'D' or 'N', 'Z' represents the amino acids 'Q' or 'E'

Parameters **ens** – set

Returns the smallest category in which the set is included, and its name

2.3 The "utilities" module

RCMAP.utilities.**compatibility**(*amino_acid*, *category*, *gaps=False*)

Tests if a set of amino acids (or a single amino acid) is included in a category. Treats the amino acids 'B' and 'Z' which are ambiguous:'B' represents the amino acids 'D' or 'N', 'Z' represents the amino acids 'Q' or 'E'. Returns gaps (True or False) if there is a gap in the set to test. If the parameter gaps = False, gaps are not taken into account. When the parameter gaps is defined True by the user, gaps are taken into account. In this case, if there is a gap in the set of amino acids observed in the reference sequences, compatibility returns True.

Parameters

- **gaps** – consider gaps if gaps is True
- **amino_acid** – set of amino acids
- **category** – a category (set)

Returns True if the set of amino acids is included in the category, False if not

RCMAP.utilities.**get_positions_list**(*positions*, *pos_max*)

Transforms the positions given by the user in a usable form for the other functions. Replaces non given positions by the beginning or the end of the sequence.

Parameters

- **pos_max** – maximal authorized value for a position (length of the sequence)
- **positions** – positions like ['3:10', '8:25' ,'32', '45:', ':5', ':']

Returns positions like [[3,10], [8,25], [32,32], [45,pos_max], [1,5], [1,pos_max]]

RCMAP.utilities.**get_weight**(*window*, *window_method*)

Defines the weights for a sliding weighted window. is evaluated. These weights can be calculated with different method:

- The Bartlett window is defined as : w(n) = 2/(M-1) * ((M-1)/2 - abs(n-(M-1)/2))
- The Hamming window is defined as : w(n) = 0.54 - 0.46 * cos((2*pi*n)/(M-1)) 0 <= n <= M-1
- The Hanning window is defined as : w(n) = 0.5 - 0.5 * cos((2*pi*n)/(M-1)) 0 <= n <= M-1
- The flat window is defined as : w(n) = 1

Parameters

- **window** – length of the window
- **window_method** – Calculation method of the weights at every position in the window

Returns the list of weights

RCMAP.utilities.**sort_categories**()

Used in classification_aa.py in find_category() which needs the categories sorted by size.

Returns the list of sorted categories, from that which contains the least of amino acids,

to that which contains the most

RCMAP.utilities.**summary_info**(*list_compatibility*, *list_info*)

Summarises the global information of a sequence: the total number of 'True' positions (which means the amino acid of the evaluated sequence is compatible with the category observed in the reference sequences) and the total of information associated. In the same way for 'False' positions.

Parameters

- **list_compatibility** – list of compatibility True or False at every position
- **list_info** – list of information at every position

Returns number of True, sum of information for True positions, number of False, sum of information for False positions

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

r

INDEX