latticeproteins Documentation

Release 0.2

Zach Sailer

Apr 12, 2017

Contents:

Latticeproteins is a python package for evaluating 2d lattice protein models. For a basic example of how to use this package, see the documentation below.

This package was originally written by Jesse Bloom, and later adapted by Zach Sailer. If you use this package, please cite Jesse's papers:

- Protein stability promotes evolvability
- Stability and the evolvability of function in a model protein

CHAPTER 1

Install

To get the latest release from Pypi:

```
pip install latticeproteins
```

Tutorials

A few things before you start

There are a few things you need to know before you get started. Due to the poor scaling of lattice protein calculations, the `latticeproteins` package takes a few precautions. First, folding a sequence is done in separate steps (and functions) rather than through a single call. This forces you to be aware of the magnitude of each call. You'll notice in these tutorials, we have to import many functions.

Second, the hardest step (most memory and time) in the calculation, by far, is enumerating conformations on the grid. `latticeproteins` tries to reduce the pain by creating a database of conformations in pickle files after the first creation. If you delete this directory, it will have to recreate it next time you run calculations.

A basic example of a lattice protein

Import the latticeproteins package.

Input:

```
import latticeproteins as lp
```

The `LatticeThermodyanmics` class creates objects that can calculate lattice protein thermodynamics for any sequences of a specified length. In the example below, we initialize this object for sequences of length 10. Note that to avoid repeating expensive conformation enumerations, the `LatticeThermodynamics` object creates a directory in your current location called `database`. Inside this directory, it stores python `pickle` files that include a database of all conformations on a 2d grid.

Input:

```
seq_length = 10
temperature = 1.0
lattice = lp.LatticeThermodynamics.from_length(seq_length, 1.0)
```

Now, we'll create a random sequence with the given length and start evaluating thermodynamic values.

Input:

```
seq = lp.random_sequence(seq_length)
print(seq)
```

Output:

```
['L', 'E', 'V', 'R', 'A', 'H', 'F', 'K', 'G', 'F']
```

Input:

```
print("Energy of native conformation: %f" % lattice.nativeE(seq))
print("stability of native conformation: %f" % lattice.stability(seq))
print("fraction folded: %f" % lattice.fracfolded(seq))
```

Output:

```
Energy of native conformation: -22.400000
stability of native conformation: 0.608617
fraction folded: 0.352375
```

The lattice protein package comes with a drawing module that creates SVG drawing of the lattice conformations.

Input:

```
conf = lattice.native_conf(seq)
lp.draw.in_notebook(seq, conf)
```

Output:

Fold lattice protein to nonnative state

The `LatticeThermodynamics` object can also do the above calculations while using a specified *target* native state.

Input:

```
# Find the 5 lowest energy conformations.
alt_conf = lattice.k_lowest_confs(seq, 5)

# Choose the 5th lowest as the target fold.
target = alt_conf[-1]
lp.draw.in_notebook(seq, target)
```

Output:

Input:

```
print("Energy of native conformation: %f" % lattice.nativeE(seq, target=target))
print("stability of native conformation: %f" % lattice.stability(seq, target=target))
print("fraction folded: %f" % lattice.fracfolded(seq, target=target))
```

Output:

```
Energy of native conformation: -20.110000
stability of native conformation: 3.296724
fraction folded: 0.035684
```

API documentation

latticeproteins package

Submodules

latticeproteins.conformations module

Module for constructing conformation database for sequences of set length.

Originally written by Jesse Bloom, 2004.

Updated by Zach Sailer, 2017.

class latticeproteins.conformations.**ConformationList**(*length*, *conflist*, *interaction_energies={'QI': -3.22, 'IA': -4.41, 'DY': -2.25, 'DA': -1.57, 'WV': -5.05, 'DV': -2.25, 'PD': -1.19, 'EI': -3.23, 'PA': -1.81, 'IK': -2.7, 'SP': -1.35, 'HM': -3.31, 'VI': -5.58, 'CS': -2.86, 'VC': -4.46, 'IF': -6.39, 'LR': -3.15, 'QY': -2.53, 'GK': -0.84, 'LS': -3.16, 'EE': -1.18, 'EC': -2.08, 'AK': -1.1, 'MM': -6.06, 'YT': -2.48, 'MK': -3.11, 'CI': -5.03, 'GF': -3.72, 'CR': -2.7, 'MV': -5.52, 'TP': -1.66, 'YC': -3.89, 'SH': -1.94, 'VF': -5.75, 'GY': -2.5, 'QH': -1.85, 'AR': -1.5, 'MG': -3.75, 'MT': -3.73, 'YM': -4.92, 'WP': -3.66, 'LQ': -3.09, 'NY': -2.47, 'TH': -2.31, 'CK': -1.54, 'KK': 0.13, 'KR': -0.06, 'HS': -1.94, 'AI': -4.41, 'MC': -5.05, 'LY': -4.26, 'HW': -4.02, 'TE': -1.45, 'MP': -4.11, 'WN': -3.11, 'PY': -2.8, 'CL': -5.03, 'QA': -1.7, 'YE': -2.42, 'KN': -0.91, 'NE': -1.43, 'FQ': -3.3, 'GP': -1.72, 'KD': -1.32, 'TM': -3.73, 'VQ': -2.67, 'LA': -3.96, 'SQ': -1.37, 'HR': -2.12, 'WR': -3.56, 'KH': -1.09, 'ML': -6.01, 'WT': -3.31, 'VE': -2.56, 'QW': -3.16, 'IQ': -3.22, 'MI': -6.33, 'RH': -2.12, 'IT': -3.74, 'EG': -1.22, 'QG': -1.54, 'SE': -1.48, 'LH': -3.84, 'TV': -2.95, 'GI': -3.65, 'NC': -2.59, 'KL': -2.63, 'CW': -4.76, 'CV': -4.46, 'MQ': -3.17, 'PE': -1.4, 'NF': -3.55, 'WK': -2.49, 'YN': -2.47, 'FG': -3.72, 'VL': -5.38, 'LK': -2.63, 'RF': -3.54, 'EK': -1.6, 'YW': -4.44, 'QC': -2.73, 'QL': -3.09, 'SV': -2.79, 'FF': -6.85, 'CE': -2.08, 'PI': -3.47, 'WY': -4.44, 'CP': -2.92, 'MS': -3.55, 'HQ': -1.85, 'GE': -1.22, 'HH': -2.78, 'FT': -3.76, 'SG': -1.7, 'MY': -4.92, 'IR': -3.33, 'NV': -2.36, 'YV': -4.05, 'DM': -2.9, 'HG': -1.94, 'WH': -4.02, 'NS': -1.31, 'GG': -2.17, 'VW': -5.05, 'RT': -1.97, 'PL': -3.06, 'IG': -3.65, 'LF': -6.26, 'WQ': -3.16, 'QV': -2.67, 'WS': -2.95, 'RR': -1.39, 'YS': -2.3, 'TQ': -1.59, 'NQ': -1.36, 'CM': -5.05, 'RN': -1.41, 'RL': -3.15, 'PQ': -1.85, 'LR': -2.07, 'HN': -2.01, 'QR': -1.85, 'TN': -1.51, 'LD': -2.59, 'ST': -1.59, 'EN': -1.43, 'GN': -1.56,*

Bases: `object`

Build an Conformations like object without the database. Uses a list of conformations provided by user to construct a Conformations object.

Note This will likely be much slower at calculating large lists of conformations.

Parameters

- **length** – is an integer specifying the length of the protein for which we are computing the contacts. It must be >= 2.
- **conflist** – a list of conformations.
- **interaction_energies** – specifies the interaction energies between residues. By default, this is interactions.miyazawa_jernigan.

fold_sequence(*seq*, *temp*)

Folds a protein sequence, calculate native energy and partition sum.

Parameters

- **seq** (*string*) – is the sequence of the protein to be folded as one-letter amino acid codes. It should be a string or list of length 'c.Length()'.
- **temp** – is the temperature at which the protein is to be folded. It must be a number > 0. It represents a reduced temperature, scaled so that a value of 1 represents 273 K.

Returns

- **minE** (*float*) – The energy of the lowest energy conformation.
- **conf** (*str*) – Lowest energy conformation.
- **partitionsum** (*float*) – Total partition function sum.
- **numcontacts** (*int*) – Number of contacts in the native conformation.
- **folds** (*bool*) – True if lattice protein has a single lowest energy.

length()

Returns the length of the protein these conformations are for.

max_contacts()

Gets the most contacts of any conformation.

Returns **n** – is returned as the number of contacts for the conformation with the most contacts.

Return type int

num_conformations(*contacts=None*)

Returns the number of conformations.

If 'contacts' has its default value of 'None', returns the total number of conformations (self-avoiding walks).

If 'contacts' has an integer value, returns the number of conformations with 'contacts' contacts. If there are no walks with this number of contacts, returns 0.

num_contact_sets(*contacts=None*)

Returns the number of unique contact sets.

If 'contacts' has its default value of 'None', returns the total number of unique contact sets (defined as the list of all contacts of non-adjacent residues).

If 'contacts' has an integer value, returns the number of unique contact sets with 'contacts' contacts. If there are no contact sets with this number of contacts, returns 0.

unique_conformations(*numcontacts*)

Gets all unique conformations with specified number of contacts.

Parameters **numcontacts** (*int*) – Number of contacts to include in unique conformations list.

Returns **clist** – is of all unique conformations with exactly 'numcontacts' contacts. A conformation is "unique" if it is the only conformation that gives rise to its particular contact set. If there are no unique conformations with 'numcontacts' contacts, 'clist' is an empty list. Conformations are specified as strings of 'U', 'R', 'L', and 'D' as described in 'FoldSequence'.

Return type list

class latticeproteins.conformations.**Conformations**(*length*, *database_dir='database/'*, *interaction_energies={'QI': -3.22, 'IA': -4.41, 'DY': -2.25, 'DA': -1.57, 'WV': -5.05, 'DV': -2.25, 'PD': -1.19, 'EI': -3.23, 'PA': -1.81, 'IK': -2.7, 'SP': -1.35, 'HM': -3.31, 'VI': -5.58, 'CS': -2.86, 'VC': -4.46, 'IF': -6.39, 'LR': -3.15, 'QY': -2.53, 'GK': -0.84, 'LS': -3.16, 'EE': -1.18, 'EC': -2.08, 'AK': -1.1, 'MM': -6.06, 'YT': -2.48, 'MK': -3.11, 'CI': -5.03, 'GF': -3.72, 'CR': -2.7, 'MV': -5.52, 'TP': -1.66, 'YC': -3.89, 'SH': -1.94, 'VF': -5.75, 'GY': -2.5, 'QH': -1.85, 'AR': -1.5, 'MG': -3.75, 'MT': -3.73, 'YM': -4.92, 'WP': -3.66, 'LQ': -3.09, 'NY': -2.47, 'TH': -2.31, 'CK': -1.54, 'KK': 0.13, 'KR': -0.06, 'HS': -1.94, 'AI': -4.41, 'MC': -5.05, 'LY': -4.26, 'HW': -4.02, 'TE': -1.45, 'MP': -4.11, 'WN': -3.11, 'PY': -2.8, 'CL': -5.03, 'QA': -1.7, 'YE': -2.42, 'KN': -0.91, 'NE': -1.43, 'FQ': -3.3, 'GP': -1.72, 'KD': -1.32, 'TM': -3.73, 'VQ': -2.67, 'LA': -3.96, 'SQ': -1.37, 'HR': -2.12, 'WR': -3.56, 'KH': -1.09, 'ML': -6.01, 'WT': -3.31, 'VE': -2.56, 'QW': -3.16, 'IQ': -3.22, 'MI': -6.33, 'RH': -2.12, 'IT': -3.74, 'EG': -1.22, 'QG': -1.54, 'SE': -1.48, 'LH': -3.84, 'TV': -2.95, 'GI': -3.65, 'NC': -2.59, 'KL': -2.63, 'CW': -4.76, 'CV': -4.46, 'MQ': -3.17, 'PE': -1.4, 'NF': -3.55, 'WK': -2.49, 'YN': -2.47, 'FG': -3.72, 'VL': -5.38, 'LK': -2.63, 'RF': -3.54, 'EK': -1.6, 'YW': -4.44, 'QC': -2.73, 'QL': -3.09, 'SV': -2.79, 'FF': -6.85, 'CE': -2.08, 'PI': -3.47, 'WY': -4.44, 'CP': -2.92, 'MS': -3.55, 'HQ': -1.85, 'GE': -1.22, 'HH': -2.78, 'FT': -3.76, 'SG': -1.7, 'MY': -4.92, 'IR': -3.33, 'NV': -2.36, 'YV': -4.05, 'DM': -2.9, 'HG': -1.94, 'WH': -4.02, 'NS': -1.31, 'GG': -2.17, 'VW': -5.05, 'RT': -1.97, 'PL': -3.06, 'IG': -3.65, 'LF': -6.26, 'WQ': -3.16, 'QV': -2.67, 'WS': -2.95, 'RR': -1.39, 'YS': -2.3, 'TQ': -1.59, 'NQ': -1.36, 'CM': -5.05, 'RN': -1.41, 'RL': -3.15, 'PQ': -1.73, 'ER': -2.07, 'HN': -2.01, 'QR': -1.85, 'TN': -1.51, 'LD': -2.59, 'ST': -1.59, 'EN': -1.43, 'GN': -1.56, 'NK': -0.91, 'CH': -3.63, 'GS': -1.7, 'RI': -3.33, 'RP': -1.85, 'SS': -1.48, 'VG': -3.06, 'AD': -1.57, 'MN': -3.5, 'YG': -2.5, 'AF': -4.36, 'KF': -2.83, 'TY': -2.48, 'PR': -1.85, 'DW': -2.91, 'AE': -1.51, 'WW': -5.42, 'KM': -3.11, 'LM': -6.01, 'NN': -1.59, 'NH': -2.01, 'ND': -1.33, 'GL': -3.43, 'FH': -4.61, 'IM': -6.33, 'ID': -2.91, 'WG': -3.37, 'AP': -1.81, 'DC':*

Bases: `object`

Creates a database of conformations for a protein of specified length.

The created 'Conformations' object 'c' stores the contact lists and the number of conformations with these contact sets for all self-avoiding walks of length 'length'. It can then be used to compute the free energy of a protein folding to the lowest energy conformation.

Parameters

- **`length`** – is an integer specifying the length of the protein for which we are computing the contacts. It must be >= 2.
- **`database_dir`** – specifies the name of the database directory storing existing conformations. If the conformation instance already exists in this database we return the existing data, and if it doesn't we store it in the database.
- **`interaction_energies`** – specifies the interaction energies between residues. By default, this is interactions.miyazawa_jernigan.

`_numconformations`
dict – A dictionary mapping the number of contact sets to the number of conformations with that contact set.

`_contactsets`
list of lists – 'self._contactsets' is a list of contact sets. 'self._contactsets[i]' is the contact set for contact i. It is a list of numbers. 'x = self._contactsets[i]' describes the residues in contact in contact 'i'. If this contact is between residues 'ires' and 'jres', then 'x = self._length * ires + jres' where 0 <= ires, jres < 'self._length', and ires < jres + 1 contact sets

`_contactsetdegeneracy`
list – 'self._contactsetdegeneracy' is a list of integers giving the degeneracy of the contact sets (the number of different conformations with this contact set). 'self_contactsetdegeneracy[i]' is the degeneracy of the contact set 'self._contactsets[i]'

`_contactsetconformation`
list – 'self._contactsetconformation' is a list of the conformations associated with each contact set. If contact set 'self._contactsets[i]' is degenerate ('self._contactsetdegeneracy[i]' > 1), the value 'self._contactsetconformation[i]' is 'None'. Otherwise, it is the string representing the conformation that gives rise to contact set 'self._contactsets[i]'. The conformations are given such that 'self._contactsetconformation[i][j]' gives the conformation of bond 'j' (0 <= j < 'self._length' - 1) as 'U' (Up), 'R' (Right), 'D' (Down), or 'L' (Left). We require the first bond to be Up, and the first non-Up bond to be Right.

`_numcontactsets`
dict – 'self._numcontactsets[i]' holds the number of different contact sets with 'i' contacts.

`fold_sequence`(*seq*, *temp*)
Folds a protein sequence; calculates native energy and partition sum.

Parameters

- **`seq`** (*`string`*) – is the sequence of the protein to be folded as one-letter amino acid codes. It should be a string or list of length 'c.Length()'.
- **`temp`** – is the temperature at which the protein is to be folded. It must be a number > 0. It represents a reduced temperature, scaled so that a value of 1 represents 273 K.

Returns

- **minE** (*float*) – The energy of the lowest energy conformation.

- **conf** (*str*) – Lowest energy conformation.
- **partitionsum** (*float*) – Total partition function sum.
- **numcontacts** (*int*) – Number of contacts in the native conformation.
- **folds** (*bool*) – True if lattice protein has a single lowest energy.

k_lowest_confs(*seq*, *temp*, *k*)
Get the *k* lowest conformations in the sequence's conformational ensemble.

length()
Returns the length of the protein these conformations are for.

max_contacts()
Gets the most contacts of any conformation.

Returns **n** – is returned as the number of contacts for the conformation with the most contacts.

Return type int

num_conformations(*contacts=None*)
Returns the number of conformations.

If 'contacts' has its default value of 'None', returns the total number of conformations (self-avoiding walks).

If 'contacts' has an integer value, returns the number of conformations with 'contacts' contacts. If there are no walks with this number of contacts, returns 0.

num_contact_sets(*contacts=None*)
Returns the number of unique contact sets.

If 'contacts' has its default value of 'None', returns the total number of unique contact sets (defined as the list of all contacts of non-adjacent residues).

If 'contacts' has an integer value, returns the number of unique contact sets with 'contacts' contacts. If there are no contact sets with this number of contacts, returns 0.

unique_conformations(*numcontacts*)
Gets all unique conformations with specified number of contacts.

Parameters **numcontacts** (*int*) – Number of contacts to include in unique conformations list.

Returns **clist** – is of all unique conformations with exactly 'numcontacts' contacts. A conformation is "unique" if it is the only conformation that gives rise to its particular contact set. If there are no unique conformations with 'numcontacts' contacts, 'clist' is an empty list. Conformations are specified as strings of 'U', 'R', 'L', and 'D' as described in 'FoldSequence'.

Return type list

exception latticeproteins.conformations.**ConformationsError**
Bases: Exception

Error finding or storing a conformation.

exception latticeproteins.conformations.**PickleProtocolError**
Bases: Exception

Error is pickle version is too old.

latticeproteins.conformations.**fold_energy**(*sequence*, *conformation*, *interactions={'QI': -3.22*, *'IA': -4.41*, *'DY': -2.25*, *'DA': -1.57*, *'WV': -5.05*, *'DV': -2.25*, *'PD': -1.19*, *'EI': -3.23*, *'PA': -1.81*, *'IK': -2.7*, *'SP': -1.35*, *'HM': -3.31*, *'VI': -5.58*, *'CS': -2.86*, *'VC': -4.46*, *'IF': -6.39*, *'LR': -3.15*, *'QY': -2.53*, *'GK': -0.84*, *'LS': -3.16*, *'EE': -1.18*, *'EC': -2.08*, *'AK': -1.1*, *'MM': -6.06*, *'YT': -2.48*, *'MK': -3.11*, *'CI': -5.03*, *'GF': -3.72*, *'CR': -2.7*, *'MV': -5.52*, *'TP': -1.66*, *'YC': -3.89*, *'SH': -1.94*, *'VF': -5.75*, *'GY': -2.5*, *'QH': -1.85*, *'AR': -1.5*, *'MG': -3.75*, *'MT': -3.73*, *'YM': -4.92*, *'WP': -3.66*, *'LQ': -3.09*, *'NY': -2.47*, *'TH': -2.31*, *'CK': -1.54*, *'KK': 0.13*, *'KR': -0.06*, *'HS': -1.94*, *'AI': -4.41*, *'MC': -5.05*, *'LY': -4.26*, *'HW': -4.02*, *'TE': -1.45*, *'MP': -4.11*, *'WN': -3.11*, *'PY': -2.8*, *'CL': -5.03*, *'QA': -1.7*, *'YE': -2.42*, *'KN': -0.91*, *'NE': -1.43*, *'FQ': -3.3*, *'GP': -1.72*, *'KD': -1.32*, *'TM': -3.73*, *'VQ': -2.67*, *'LA': -3.96*, *'SQ': -1.37*, *'HR': -2.12*, *'WR': -3.56*, *'KH': -1.09*, *'ML': -6.01*, *'WT': -3.31*, *'VE': -2.56*, *'QW': -3.16*, *'IQ': -3.22*, *'MI': -6.33*, *'RH': -2.12*, *'IT': -3.74*, *'EG': -1.22*, *'QG': -1.54*, *'SE': -1.48*, *'LH': -3.84*, *'TV': -2.95*, *'GI': -3.65*, *'NC': -2.59*, *'KL': -2.63*, *'CW': -4.76*, *'CV': -4.46*, *'MQ': -3.17*, *'PE': -1.4*, *'NF': -3.55*, *'WK': -2.49*, *'YN': -2.47*, *'FG': -3.72*, *'VL': -5.38*, *'LK': -2.63*, *'RF': -3.54*, *'EK': -1.6*, *'YW': -4.44*, *'QC': -2.73*, *'QL': -3.09*, *'SV': -2.79*, *'FF': -6.85*, *'CE': -2.08*, *'PI': -3.47*, *'WY': -4.44*, *'CP': -2.92*, *'MS': -3.55*, *'HQ': -1.85*, *'GE': -1.22*, *'HH': -2.78*, *'FT': -3.76*, *'SG': -1.7*, *'MY': -4.92*, *'IR': -3.33*, *'NV': -2.36*, *'YV': -4.05*, *'DM': -2.9*, *'HG': -1.94*, *'WH': -4.02*, *'NS': -1.31*, *'GG': -2.17*, *'VW': -5.05*, *'RT': -1.97*, *'PL': -3.06*, *'IG': -3.65*, *'LF': -6.26*, *'WQ': -3.16*, *'QV': -2.67*, *'WS': -2.95*, *'RR': -1.39*, *'YS': -2.3*, *'TQ': -1.59*, *'NQ': -1.36*, *'CM': -5.05*, *'RN': -1.41*, *'RL': -3.15*, *'PQ': -1.73*, *'ER': -2.07*, *'HN': -2.01*, *'QR': -1.85*, *'TN': -1.51*, *'LD': -2.59*, *'ST': -1.59*, *'EN': -1.43*, *'GN': -1.56*, *'NK': -0.91*, *'CH': -3.63*, *'GS': -1.7*, *'RI': -3.33*, *'RP': -1.85*, *'SS': -1.48*, *'VG': -3.06*, *'AD': -1.57*, *'MN': -3.5*, *'YG': -2.5*, *'AF': -4.36*, *'KF': -2.83*, *'TY': -2.48*, *'PR': -1.85*, *'DW': -2.91*, *'AE': -1.51*, *'WW': -5.42*, *'KM': -3.11*, *'LM': -6.01*, *'NN': -1.59*, *'NH': -2.01*, *'ND': -1.33*, *'GL': -3.43*, *'FH': -4.61*, *'IM': -6.33*, *'ID': -2.91*, *'WG': -3.37*, *'AP': -1.81*, *'DC': -2.66*, *'YF': -4.95*, *'RA': -1.5*, *'HK': -1.09*, *'DD': -0.96*, *'VT': -2.95*, *'AW': -3.93*, *'CQ': -2.73*, *'FY': -4.95*, *'CY': -3.89*, *'SR': -1.22*, *'YI': -4.63*, *'DF': -3.31*, *'QN': -1.36*, *'QS': -1.37*, *'QE': -1.33*, *'EY': -2.42*, *'TK': -1.02*, *'TI': -3.74*, *'DH': -2.14*, *'AC': -3.38*, *'ET': -1.45*, *'AA': -2.51*, *'LV': -5.38*, *'IP': -3.47*, *'LW': -5.5*, *'YD': -2.25*, *'FL': -6.26*, *'NL': -2.59*, *'KV': -1.95*, *'GH': -1.94*, *'IY': -4.63*, *'TD': -1.66*, *'KC': -1.54*, *'LE': -2.91*, *'GD': -1.62*, *'PT': -1.66*, *'WA': -3.93*, *'VH': -3.38*, *'ME': -3.19*,

Calculate the energy of the sequence with the given conformation.

Parameters

- **sequence** (*str*) – Amino acid sequence to fold.
- **conformation** (*str*) – Conformation according to latticemodel's conformations format (e.g. 'UDLLDRU')

Returns **energy** – energy of the conformation (sum of all contact energies)

Return type float

latticeproteins.conformations.**lattice_contacts**(*sequence*, *conformation*)

Find all contacts in conformation.

Parameters

- **sequence** (*str*) – Amino acid sequence to fold.
- **conformation** (*str*) – Conformation according to latticemodel's conformations format (e.g. 'UDLLDRU')

Returns **contacts** – list of contact pairs

Return type list

latticeproteins.draw module

Module for creating SVG's of protein lattice configurations.

Originally written by Jesse Bloom, 2004.

Updated by Zach Sailer, 2017.

Example call:

```
>>> # Create an instance
>>> drawing = latticegpm.svg.Configuration(sequence, configuration, filename=
↪"drawing1.svg")
>>> # Save to file
>>> # drawing.save()
>>> # Print in Jupyter (IPython) notebook
>>> drawing.notebook
```

class latticeproteins.draw.**Configuration**(*sequence*, *configuration*, *color_sequence=None*, *rotation=0*, *font_size=20*, *dot_scale=1.0*, *font_weight='normal'*)

Bases: IPython.core.display.SVG

Main class for drawing an SVG of a lattice protein's fold.

Parameters

- **sequence** (*str*) – Amino acid sequence
- **configuration** (*str*) – sequence of direction letters describing the 2d configuration.
- **colors** (*list of strings*) – list of colors for each amino acid in sequence
- **rotation** (*int*) – rotate the configuration by 0, 90, 180, or 270 degrees
- **font_size** (*int*) – Font size, in pixels, of sequence in configuration. The svg will scale with the font size of the letters.

Examples

```
>>> # Create an instance
>>> drawing = Configuration(sequence, configuration)
>>> # Save to file
>>> # drawing.save()
>>> # Print in Jupyter (IPython) notebook
>>> drawing.notebook
```

data
Return svg as a string.

notebook
Display SVG in Jupyther notebook.

rotate (*rotation*)
Rotate the drawing by 90, 180, or 270.

save (*filename*)
save svg

latticeproteins.draw.**configuration_to_array** (*sequence*, *configuration*)
Create a square numpy array with the configuration laid out.

latticeproteins.draw.**in_notebook** (*sequence*, *conf*, ***kwargs*)
Creates a Python SVG configuration object. Automagically displays in notebook.

latticeproteins.draw.**to_file** (*sequence*, *conf*, *filename*, ***kwargs*)

latticeproteins.interactions module

Originally written by Jesse Bloom, 2004.

Updated by Zach Sailer, 2017.

latticeproteins.sequences module

Originally written by Jesse Bloom, 2004.

Updated by Zach Sailer, 2017.

exception latticeproteins.sequences.**SequenceError**
Bases: Exception

Error with a lattice protein sequence.

latticeproteins.sequences.**hamming_distance** (*seq1*, *seq2*)
Returns the Hamming distance between two sequences.

latticeproteins.sequences.**mutate_sequence** (*seq*, *mutrate*)
Mutates a protein sequence.

Parameters

- **seq** – is a protein sequence, specified as either a string or a list.
- **mutrate** – Mutates each residue in 'seq' to some different residue with probability 'mutrate'. So 'mutrate' is the per residue mutation rate.

Returns the new sequence as a list.

Return type newseq

latticeproteins.sequences.**n_mutants**(*seq*, *nmutations*, *nsequences*)
Returns sequences with a specified number of mutations.

Parameters

- **seq** – is a string or list specifying the protein we wish to mutate.
- **nmutations** – is the number of mutations each mutant of 'seq' should have. It must be <= 'len(seq)' and > 0.
- **nsequences** – is the number of mutant sequences to make. It can be 'ALL', in which case we make all possible mutants with 'nmutations', or it can be some positive integer in which case we make this many randomly chosen mutants with 'nmutations' mutations. 'ALL' is only a valid option only when 'nmutations' is 1 or 2.

Returns **seqlist** – List of mutant sequences n mutations away.

Return type list

latticeproteins.sequences.**random_sequence**(*length*)
Returns a random sequence of the specified length.

latticeproteins.thermodynamics module

Module for calculating thermodynamics of lattice protein sequences.

Originally written by Jesse Bloom, 2004.

Updated by Zach Sailer, 2017.

class latticeproteins.thermodynamics.**GroupThermodynamics**(*seqlist*, *temp*, *confs*, *target=None*)
Bases: object

Efficiently calculates thermodynamic properties for a list of lattice proteins.

Parameters

- **seqlist** (*list*) – List of lattice proteins.
- **temp** (*float*) – temperature of the system.
- **confs** (*Conformations or ConformationList object*) – Conformation database for lattice with set length
- **target** (*str (optional, default=None)*) – target conformation to fold protein list

seqlist
list – list of sequences.

temp
float – temperature of the system.

nativeEs
array – native (or target) energy for sequences in seqlist

stabilities
array – array of stabilities for sequences in seqlist

fracfolded
array – array of fraction folded for sequences in seqlist

fracfolded
Fracfolded folded for all sequences in seqlist.

stabilities
Folding stability for all sequences in seqlist.

class latticeproteins.thermodynamics.**LatticeThermodynamics**(*temp*, *confs*)
Bases: object

Attaches thermodynamic evaluators to a lattice protein conformation database.

Parameters

- **temp** (*float*) – the temperature at which the fitness is computed.
- **confs** (*conformations.Conformations object*) – is the 'conformations.Conformations' object used to fold the protein sequences. 'conformations.Length()' specifies the length of the protein sequences that can be folded.

all_metrics(*seq*)
Compute lattice NativeE, Stability, and Fitness of a given sequence.

Parameters **seq** (*str*) – protein sequence string.

Returns

- **nativeE** (*float*) – energy of the native state.
- **dG** (*float*) – stability of the native state.
- **fitness** (*float*) – fitness of the native state.

fracfolded(*seq*, *target=None*)
Compute the fraction folded of the sequence.

Parameters **seq** (*str or list*) – sequence to fold.

Returns **fracfolded** – fractioned folded.

Return type float

classmethod `from_length`(*length*, *temp*, *database_dir='database/'*, *interactions={'QI': -3.22, 'IA': -4.41, 'DY': -2.25, 'DA': -1.57, 'WV': -5.05, 'DV': -2.25, 'PD': -1.19, 'EI': -3.23, 'PA': -1.81, 'IK': -2.7, 'SP': -1.35, 'HM': -3.31, 'VI': -5.58, 'CS': -2.86, 'VC': -4.46, 'IF': -6.39, 'LR': -3.15, 'QY': -2.53, 'GK': -0.84, 'LS': -3.16, 'EE': -1.18, 'EC': -2.08, 'AK': -1.1, 'MM': -6.06, 'YT': -2.48, 'MK': -3.11, 'CI': -5.03, 'GF': -3.72, 'CR': -2.7, 'MV': -5.52, 'TP': -1.66, 'YC': -3.89, 'SH': -1.94, 'VF': -5.75, 'GY': -2.5, 'QH': -1.85, 'AR': -1.5, 'MG': -3.75, 'MT': -3.73, 'YM': -4.92, 'WP': -3.66, 'LQ': -3.09, 'NY': -2.47, 'TH': -2.31, 'CK': -1.54, 'KK': 0.13, 'KR': -0.06, 'HS': -1.94, 'AI': -4.41, 'MC': -5.05, 'LY': -4.26, 'HW': -4.02, 'TE': -1.45, 'MP': -4.11, 'WN': -3.11, 'PY': -2.8, 'CL': -5.03, 'QA': -1.7, 'YE': -2.42, 'KN': -0.91, 'NE': -1.43, 'FQ': -3.3, 'GP': -1.72, 'KD': -1.32, 'TM': -3.73, 'VQ': -2.67, 'LA': -3.96, 'SQ': -1.37, 'HR': -2.12, 'WR': -3.56, 'KH': -1.09, 'ML': -6.01, 'WT': -3.31, 'VE': -2.56, 'QW': -3.16, 'IQ': -3.22, 'MI': -6.33, 'RH': -2.12, 'IT': -3.74, 'EG': -1.22, 'QG': -1.54, 'SE': -1.48, 'LH': -3.84, 'TV': -2.95, 'GI': -3.65, 'NC': -2.59, 'KL': -2.63, 'CW': -4.76, 'CV': -4.46, 'MQ': -3.17, 'PE': -1.4, 'NF': -3.55, 'WK': -2.49, 'YN': -2.47, 'FG': -3.72, 'VL': -5.38, 'LK': -2.63, 'RF': -3.54, 'EK': -1.6, 'YW': -4.44, 'QC': -2.73, 'QL': -3.09, 'SV': -2.79, 'FF': -6.85, 'CE': -2.08, 'PI': -3.47, 'WY': -4.44, 'CP': -2.92, 'MS': -3.55, 'HQ': -1.85, 'GE': -1.22, 'HH': -2.78, 'FT': -3.76, 'SG': -1.7, 'MY': -4.92, 'IR': -3.33, 'NV': -2.36, 'YV': -4.05, 'DM': -2.9, 'HG': -1.94, 'WH': -4.02, 'NS': -1.31, 'GG': -2.17, 'VW': -5.05, 'RT': -1.97, 'PL': -3.06, 'IG': -3.65, 'LF': -6.26, 'WQ': -3.16, 'QV': -2.67, 'WS': -2.95, 'RR': -1.39, 'YS': -2.3, 'TQ': -1.59, 'NQ': -1.36, 'CM': -5.05, 'RN': -1.41, 'RL': -3.15, 'PQ': -1.73, 'ER': -2.07, 'HN': -2.01, 'QR': -1.85, 'TN': -1.51, 'LD': -2.59, 'ST': -1.59, 'EN': -1.43, 'GN': -1.56, 'NK': -0.91, 'CH': -3.63, 'GS': -1.7, 'RI': -3.33, 'RP': -1.85, 'SS': -1.48, 'VG': -3.06, 'AD': -1.57, 'MN': -3.5, 'YG': -2.5, 'AF': -4.36, 'KF': -2.83, 'TY': -2.48, 'PR': -1.85, 'DW': -2.91, 'AE': -1.51, 'WW': -5.42, 'KM': -3.11, 'LM': -6.01, 'NN': -1.59, 'NH': -2.01, 'ND': -1.33, 'GL': -3.43, 'FH': -4.61, 'IM': -6.33, 'ID': -2.91, 'WG': -3.37, 'AP': -1.81, 'DC': -2.66, 'YF': -4.95, 'RA': -1.5, 'HK': -1.09, 'DD': -0.96, 'VT': -2.95, 'AW': -3.93, 'CQ': -2.73, 'FY': -4.95, 'CY': -3.89, 'SR': -1.22, 'YI': -4.63, 'DF': -3.31, 'QN': -1.36, 'QS': -1.37, 'QE': -1.33, 'EY': -2.42, 'TK': -1.02, 'TI': -3.74, 'DH': -2.14, 'AC': -3.38, 'ET': -1.45, 'AA': -2.51, 'LV': -5.38, 'IP': -3.47, 'LW': -5.5, 'YD': -2.25, 'FL': -6.26, 'NL': -2.99, 'KV': -1.95, 'GH': -1.94, 'IY': -4.63, 'TD': -1.66, 'KC': -1.54, 'LE': -2.91, 'GD': -1.62, 'PT': -1.66, 'WA': -3.93, 'VH': -3.38, 'ME': -3.19, 'TT': -1.72, 'SM': -3.55, 'RM': -3.49, 'FM': -6.68, 'PF': -3.73, 'GV': -3.06, 'WD': -2.91, 'PM': -4.11, 'FK': -2.83, 'TS': -1.59, 'AT': -2.15, 'PV': -2.96, 'FD': -3.31, 'EQ': -1.33, 'VY': -4.05, 'YR': -2.75, 'FP': -3.73, 'YA': -2.85, 'WC': -4.76, 'GM': -3.75, 'AY': -2.85, 'EM': -3.19, 'PG': -1.72, 'RY': -2.75, 'SI': -3.43, 'EL': -2.91, 'AM': -3.99, 'NT': -1.51, 'PC': -2.92, 'LG': -3.43, 'TR': -1.97, 'NP': -1.43, 'EH': -2.27, 'FE': -3.51, 'ES': -1.48, 'CT': -2.88, 'SC': -2.86, 'WF': -6.02, 'DP': -1.19, 'FV': -5.75, 'CN': -2.59, 'QT': -1.59, 'IS': -3.43, 'VP': -2.96, 'DK': -1.32, 'LN': -2.99, 'MA': -3.99, 'TG': -2.03, 'FS': -3.56, 'QF': -3.3, 'PW': -3.66, 'HE': -2.27, 'HL': -3.84, 'KY': -2.01, 'AS': -1.89, 'FA': -4.36, 'QQ': -0.89, 'AV': -3.62, 'YL': -4.26, 'NG': -1.56, 'IE': -3.23, 'KE': -1.6, 'YQ': -2.53, 'HV': -3.38, 'PP': -1.18, 'KQ': -1.02, 'NI': -2.99, 'KA': -1.1, 'VS': -2.79, 'AQ': -1.7, 'EV': -2.56, 'HT': -2.31, 'TF': -3.76, 'FW': -6.02, 'PK': -0.67, 'YH': -3.33, 'HF': -4.61, 'GA': -2.15, 'LP': -3.06, 'LC': -5.03, 'CF': -5.63, 'VR': -2.78, 'RV': -2.78, 'IH': -3.76, 'RQ': -1.85, 'TC': -2.88, 'KW': -2.49, 'LT': -3.43, 'IV': -5.58, 'IC': -5.03, 'HD': -2.14, 'YY': -3.55, 'AL': -3.96, 'FR': -3.54, 'SK': -0.83, 'PH': -2.17, 'KP': -0.67, 'RW': -3.56, 'TL': -3.43, 'GQ': -1.54, 'LI': -6.17, 'NA': -1.44, 'DR': -1.98, 'EP': -1.4, 'GT': -2.03, 'QP': -1.73, 'VM': -5.52, 'IN': -2.99, 'RK': -0.06, 'WE': -2.94, 'TA': -2.15, 'PS': -1.35, 'CD': -2.66, 'SA': -1.89, 'SW': -2.95, 'EA': -1.51, 'AN': -1.44, 'NW': -3.11, 'CA': -3.38, 'QK': -1.02, 'MH': -3.31, 'VK': -1.95, 'KI': -2.7, 'LL': -5.79, 'EF': -3.51, 'SD': -1.46, 'NM': -3.5, 'RD': -1.98, 'KG': -0.84, 'DL':*

Create a thermodynamic object for sequences of a given length.

length()
Returns the sequence length for which fitnesses are computed.

nativeE(*seq*, *target=None*)
Compute the native energy and return it.

Parameters **seq** (*str or list*) – sequence to fold.

Returns **minE** – Energy of the native state.

Return type float

native_conf(*seq*)
Return the native conformation.

stability(*seq*, *target=None*)
Computes the stability of a sequence if it is below cutoff.

Parameters **seq** (*str or list*) – sequence to fold.

Returns **stability** – Folding stability of the native state.

Return type float

exception latticeproteins.thermodynamics.**ThermodynamicsError**
Bases: Exception

Error computing lattice protein thermodynamics.

Module contents

This package contains utilities for lattice protein conformations.

Uses 2-dimensional non-compact models.

Originally written by Jesse Bloom.

Extended by Zach Sailer.

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

l

Index

Symbols

A

C

D

F

G

H

I

K

L

M

N

P

R

S

T

U