LabBlouinTools

API Documentation

January 6, 2015

Contents

1 Package labblouin

Bioinformatic python utilities modules, and libraries for use with Lab Blouin applications.

1.1 Modules

2 Module labblouin.FASTAnet

A simple parser and writer for FASTA sequence files that uses a rich object hierarchy.

2.1 Variables

Name	Description
___package___	**Value:** `None`

2.2 Class FASTAsequence

2.2.1 Methods

___init___(*self*, *name*, *seq*)

Initialize this object. Provide a name for the sequence and the sequence itself as parameters.

___iter___(*self*)

Iterate through a sequence in a pseudo-line-by-line manner as if it was read in a FASTA file.

___hash___(*self*)

___eq___(*self*, *o*)

___ne___(*self*, *o*)

count(*self*, *item*)

Return the number of characters in the sequence equal to input string.

removeGaps(*self*)

Modify the sequence so gaps are removed and return it.

toUpper(*self*)

Modify the sequence so it is uppercase and return it.

toLower(*self*)

Return a lowercase version of the sequence, also changing it in the structure.

___len___(*self*)

___str___(*self*)

2.3 Class FASTAstructure

2.3.1 Methods

___init___(*self*, *filein*=`''`, *uniqueOnly*=`True`, *curate*=`False`)

A file to be read is optional. If uniqueOnly is set to false, multiple duplicate sequences are allowed; otherwise, duplicates are ignored and their aliases are recorded in sequence Names. If curate is triggered, will remove special characters from names.

getSequenceNames(*self*)

getSequences(*self*)

getSequenceByName(*self*, *n*)

getStrictlyUngappedPositions(*self*, *seqInds*=`None`)

Acquire the positions of all strictly ungapped sites. If parameter is set, expects a list of what sequences (by index) you are checking. Defaults to all sequences.

readFile(*self*, *fin*)

Read a file in. Return this FASTA object.

read(*self*, *fast*)

Read the contents of a FASTA file.

writeFile(*self*, *fout*)

Write the information currently contained in the FASTAstructure to a file as a FASTA-formatted file.

addSequence(*self*, *name*, *seq*)

Add a sequence to the FASTA object.

renameSequence(*self*, *oldname*, *newname*)

Renames a given sequence.

removeSequence(*self*, *name*)

Remove a sequence from the FASTA object and return it; or return None if it was not found.

reorderSequences(*self*, *iterable*)

Reorder all sequences by an iterable sequence of their names.

removeGaps(*self*)

Remove the gaps for all sequences.

allUpper(*self*)

Change all sequences to uppercase.

allLower(*self*)

Change all sequences to lowercase.

___iter___(*self*)

Iterate through the FASTA by going through its sequences.

___len___(*self*)

Return the number of sequences in the FASTA object.

___str___(*self*)

Return the FASTA object as FASTA file text content.

3 Module labblouin.IO

Different shortcut functions for dealing with file systems.

IO Python Library / 2012 / Alex Safatli

E-mail: safatli@cs.dal.ca Dependencies: -

3.1 Functions

getFileName(*path*)

Returns a filename of a given path.

getFolderName(*path*)

Returns the directory name of a given path.

makeFolder(*folder_name*)

Makes a folder (if does not already exist). Returns path.

moveFile(*origin*, *target*)

Moves a file. Returns path.

copyFile(*origin*, *target*)

Copies a file. Returns path.

removeFile(*origin*)

Removes a file.

getFilesInFolder(*folder*, *ext*)

Returns a list of all full file paths for a given extension in a given folder.

3.2 Variables

Name	Description
___package___	**Value:** `'labblouin'`

4 Module labblouin.MATT2GM

```
Process MATT output and convert them to the GM format.

MATT2GM Copyright (C) 2012 Christian Blouin
MATT2GM Version Copyright (C) 2013 Christian Blouin - Jose Sergio Hleap - Alex Safatli

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see <http://www.gnu.org/licenses/>.

E-mail: cblouin@cs.dal.ca, jshleap@squalus.org, iltafas@gmail.com

Requirements:
1) Python:
       a) PDBnet.py
       ########################################
       #PDBnet.py is a python script and have#
       #to be provided with this code        #
       ########################################
```

4.1 Functions

LoadFasta(*filename*)

GetMask(*A*)

RemoveNonHomologous(*pdb*, *aln*, *msk*, *mapfile*)

Rename_fasta(*prefix*)

Rename the chains in the fasta file

main(*prefix*, *alphaC*=`False`, *redundant*=`False`)

4.2 Variables

Name	Description
aa	**Value:** `{'A': 'ALA', 'C': 'CYS', 'D': 'ASP', 'E': 'GLU', 'F': 'PH...`
___package___	**Value:** `'labblouin'`

5 Module labblouin.Ordination

Ordination is a class designed to compute and plot ordination methods such as PCA and MDS. It is intended as a helper function to PDBnet, but have the functionality to work with gm files.

This file is based on Nelle Varoquaux <nelle.varoquaux@gmail.com> code plotmds.py, available at http://scikit-learn.org/stable/auto_examples/manifold/plot_mds.html, and recomendations in stackoverflow by Jaime Fernandez (http://numericalrecipes.wordpress.com/)

Dependencies: SKlearn, PDBnet, Scipy, matplotlib

Author: Jose Sergio Hleap email: jshleap@dal.ca

5.1 Variables

Name	Description
colors	**Value:** `['b', 'r', 'k', 'g', 'y', 'm', 'c', '#EE3A8C', '#ADFF2F',...`
hexa	**Value:** `['#EE3A8C', '#ADFF2F', '#00C957', '#EEE9E9', '#EED8AE', '...`
___package___	**Value:** `'labblouin'`

5.2 Class ORD

A class for the most popular ordination methods using PDBnet instaces or gm files.

5.2.1 Methods

___init___(*self*, *prefix*, *data*, *fastafile*=`None`, *n_comp*=`2`)

PrepData4PDBnet(*self*, *data*)

Load the data to the class assuming is a PDBnet instance file

LoadDataGMfile(*self*, *data*)

Load the data to the class assuming is a GM file

dict2array2matrix(*self*, *dict*)

Giving an upper-triangle distance matrix in a dictionary, returns a distance-like array

Store(*self*)

MDS(*self*, *options*, *typeof*='classic', *dist*=False, *groups*=None)

Perform Multidimensional Scaling wither classic (PCoA) or non-metric. If you have the upper triangle of a distance matrix as a dictionary, pass the dictionary as dist.

PCA(*self*, *options*, *groups*=None)

Performs a principal coordinate analysis of the data

Plot(*self*, *options*, *groups*=None)

Plot the components from an ordination method of the class ORD. If the number of components is greater than 3, it will plot the first three components. Components has to be a n x k numpy array of eigenvectors, where n is the observations/individuals and k the components. The option groups allow to pass a list (of the same lenght of the arrar, that is a lenght of n).

PlotXDA(*self*, *membership*, *options*, *group_labels*=None)

Plots a Linear Discriminant Analysis (LDA) or a Quadratic Discriminan Analysis (QDA) with confidence ellipses at std (standard deviations)

ellipse(*self*, *singlegroupx*, *singlegroupy*, *std*=2, *color*='k')

Create an ellipse given points with x coordinates in singlegroupx and singlegroupy

pointsInEllipse(*self*, *Xs*, *Ys*, *ellipse*)

Tests which set of points are within the boundaries defined by the ellipse. The set of points are defined in two arrays Xs and Ys for the x-coordinates and y-coordinates respectively

getEllipses(*self*, *stds*)

will populate the ellipses attribute

LDA(*self*, *membership*, *options*, *group_labels*=None)

Perform a Linear discriminant analysis of the data and plot it. Membership must be an array of integers of the same lenght of the number of observations in the data.

QDA(*self*, *membership*, *options*, *group_labels*=None)

6 Module labblouin.PDBS2VMDstate

PDB(s) to VMD state Giving a list of structures (can be a single one), return a VMD state, with the same color scale (from the minimum of all to the maximum of all), white background, and other visual hacks

6.1 Functions

GetFDs(*files*)

6.2 Variables

Name	Description
InvariantLine	**Value:** `'#!/usr/local/bin/vmd\n# VMD script written by save_state...`
molLine	**Value:** `'mol new PATHNPDBNAME type pdb first 0 last -1 step 1 fil...`
lastline	**Value:** `'foreach v $viewplist {\n molinfo $v set {center_matrix ...`
fout	**Value:** `open(sys.argv [1]+ '.vmd', 'w')`
files	**Value:** `sys.argv [2:]`
scale	**Value:** `GetFDs(files)`
___package___	**Value:** `'labblouin'`

7 Module labblouin.PDBnet

PDBnet is a collection of Python objects intended to model and contain PDB protein data.

E-mail: cblouin@cs.dal.ca, safatli@cs.dal.ca, jshleap@dal.ca Dependencies: Scipy, BioPython, FASTAnet (contained in LabBlouinTools)

Version: 1.1.0

Authors: Christian Blouin, Jose Sergio Hleap, Alexander Safatli

7.1 Variables

Name	Description
PDB_LARGE_FILE_SIZE	**Value:** `100000000`
aa	**Value:** `{'ALA': 'A', 'ARG': 'R', 'ASN': 'N', 'ASP': 'D', 'CYS': '...`
aa_names	**Value:** `{'A': 'ALA', 'C': 'CYS', 'D': 'ASP', 'E': 'GLU', 'F': 'PH...`
aa_lists	**Value:** `{'ALA': ['N', 'CA', 'C', 'O', 'CB'], 'ARG': ['N', 'CA', '...`
__package__	**Value:** `'labblouin'`

7.2 Class PDBatom

object
└ **labblouin.PDBnet.PDBatom**

Known Subclasses: labblouin.PDBnet.PDBterminator

ATOM in a PDB protein structure.

7.2.1 Methods

__init__(*self*, *serial*, *name*, *x*, *y*, *z*, *oc*, *b*, *symbol*, *charge*)

Construct an atom in a PDB protein structure.

Overrides: object.__init__

fixname(*self*)

Ensures the name of this atom fits within 4 characters.

___str___(*self*)

Express this atom as a single line in a PDB file (see PDB format).

Overrides: object.___str___

GetPosition(*self*)

Get the 3-dimensional coordinate of this atom in space.

DistanceTo(*self*, *atom*)

Acquire the distance from this atom to another atom.

Inherited from object

___delattr___(), ___format___(), ___getattribute___(), ___hash___(), ___new___(), ___reduce___(), ___reduce_ex___(), ___repr___(), ___setattr___(), ___sizeof___(), ___subclasshook___()

7.2.2 Properties

Name	Description
charge	
name	
occupancy	
parent	
serial	
symbol	
tempFactor	
x	
y	
z	
Inherited from object ___class___	

7.3 Class PDBterminator

object

labblouin.PDBnet.PDBatom

labblouin.PDBnet.PDBterminator

A placeholder class that represents a terminating ATOM-like line in the PDB file.

7.3.1 Methods

___init___(*self*, *chaininst*)

Construct an atom in a PDB protein structure.

Overrides: object.___init___ extit(inherited documentation)

___str___(*self*)

Express this atom as a single line in a PDB file (see PDB format).

Overrides: object.___str___ extit(inherited documentation)

Inherited from labblouin.PDBnet.PDBatom(Section 7.2)

DistanceTo(), GetPosition(), fixname()

Inherited from object

___delattr___(), ___format___(), ___getattribute___(), ___hash___(), ___new___(), ___reduce___(), ___reduce_ex___(), ___repr___(), ___setattr___(), ___sizeof___(), ___subclasshook___()

7.3.2 Properties

Name	Description
charge	
lastatom	
lastreschain	
lastresind	
lastresname	
name	
occupancy	
parent	
serial	
symbol	
tempFactor	
x	
y	
z	
Inherited from object ___class___	

7.4 Class PDBresidue

A residue (collection of ATOM fields) in a PDB protein structure.

7.4.1 Methods

___init___(*self*, *index*=`None`, *name*=`''`)

Construct a residue (collection of atoms) in a PDB protein structure.

GetAtoms(*self*)

AddAtom(*self*, *atom*)

Add a PDBatom structure to this residue.

GetCA(*self*)

Get the alpha-carbon found in this residue as a PDBatom.

Centroid(*self*)

Calculate the centroid of this residue. Return this as a PDBatom.

InContactWith(*self*, *other*, *thres*=`4.5`)

Determine if in contact with another residue.

___int___(*self*)

___str___(*self*)

7.5 Class PDBchain

object
└ **labblouin.PDBnet.PDBchain**

Known Subclasses: labblouin.PDBnet.PDBmodel

A PDB chain (collection of protein residues).

7.5.1 Methods

___init___(*self*, *name*)

x.___init___(...) initializes x; see help(type(x)) for signature

Overrides: object.___init___ extit(inherited documentation)

GetResidues(*self*)

Acquire all of the residues in this chain. Note that if this is a large file, this will force the object to load all of these residues into memory from the file.

IterResidues(*self*)

Iteratively yield all of the residues in this chain. Note that if this is a large file, this will force the object to load all of these residues into memory from the file.

GetResidueByIndex(*self*, *index*)

Alias for acquiring a residue from this instance using [] operator.

GetAtoms(*self*)

Get all comprising atoms in this chain in order of residues.

GetIndices(*self*)

Acquire all of the indices for residues present in this chain.

AddIndexOfResidue(*self*, *index*)

Add an index for a residue to this chain. Will generate own residue information if called upon (forces lazy evaluation).

AddResidue(*self*, *resid*)

Add a residue to this chain.

AddResidueByIndex(*self*, *index*)

Add a residue to this chain by its index in the PDB. The residue object will automatically be constructed from the file.

RemoveResidue(*self*, *resid*)

Remove a residue from this chain.

ContactMap(*self*, *thres*=4.5)

Compute the contact map of this chain.

GetPrimaryPropertiesFromBioPython(*self*)

Use BioPython to populate this class with attributes.

AsFASTA(*self*)

Return the string representing this chain's FASTA sequence.

WriteAsPDB(*self*, *filename*)

Write this single chain to a file as a PDB.

SortByNumericalIndices(*self*)

Sort all internal items by a numerical index.

pop(*self*, *s*)

Pop a residue.

update(*self*, *o*)

Add residues from another chain.

___str___(*self*)

str(x)

Overrides: object.___str___ extit(inherited documentation)

___getitem___(*self*, *i*)

Force the object to load the residue if not present in this structure.

___len___(*self*)

___iter___(*self*)

Inherited from object

___delattr___(), ___format___(), ___getattribute___(), ___hash___(), ___new___(), ___reduce___(), ___reduce_ex___(), ___repr___(), ___setattr___(), ___sizeof___(), ___subclasshook___()

7.5.2 Properties

Name	Description
indices	
name	
parent	
residues	
structure	
Inherited from object	
___class___	

7.6 Class PDBmodel

object
└
labblouin.PDBnet.PDBchain
└
labblouin.PDBnet.PDBmodel

A PDB model (a special kind of chain).

7.6.1 Methods

___init___(*self*, *name*)

x.___init___(...) initializes x; see help(type(x)) for signature

Overrides: object.___init___ extit(inherited documentation)

GetResidues(*self*)

Acquire all of the residues in this model. Note that if this is a large file, this will force the object to load all of these residues into memory from the file.

Overrides: labblouin.PDBnet.PDBchain.GetResidues

IterResidues(*self*)

Iteratively yield all of the residues in this model. Note that if this is a large file, this will force the object to load all of these residues into memory from the file.

Overrides: labblouin.PDBnet.PDBchain.IterResidues

GetChains(*self*)

GetChain(*self*, *name*)

GetChainByName(*self*, *name*)

GetChainNames(*self*)

NewChain(*self*, *ch*)

Create a new PDBchain, add it to this model, and return it.

AddChain(*self*, *chain*)

Add a PDBchain (chain) to this model.

AddResidue(*self*, *resid*)

Add a residue to this model.

Overrides: labblouin.PDBnet.PDBchain.AddResidue

___getitem___(*self*, *i*)

Force the object to load the residue if not present in this structure.

Overrides: labblouin.PDBnet.PDBchain.___getitem___ extit(inherited documentation)

___len___(*self*)

Overrides: labblouin.PDBnet.PDBchain.___len___

___str___(*self*)

str(x)

Overrides: object.___str___ extit(inherited documentation)

Inherited from labblouin.PDBnet.PDBchain(Section 7.5)

AddIndexOfResidue(), AddResidueByIndex(), AsFASTA(), ContactMap(), GetAtoms(), GetIndices(), GetPrimaryPropertiesFromBioPython(), GetResidueByIndex(), RemoveResidue(), SortByNumericalIndices(), WriteAsPDB(), ___iter___(), pop(), update()

Inherited from object

___delattr___(), ___format___(), ___getattribute___(), ___hash___(), ___new___(), ___reduce___(), ___reduce_ex___(), ___repr___(), ___setattr___(), ___sizeof___(),

___subclasshook___()

7.6.2 Properties

Name	Description
chainNames	
chains	
indices	
name	
residues	
structure	
Inherited from labblouin.PDBnet.PDBchain (Section 7.5) parent	
Inherited from object ___class___	

7.7 Class PDBstructure

object
└
labblouin.PDBnet.PDBstructure

A PDB protein structure (a collection of chains/models).

7.7.1 Methods

___init___(*self*, *filein=''*)

x.___init___(...) initializes x; see help(type(x)) for signature

Overrides: object.___init___ extit(inherited documentation)

GetChainNames(*self*)

GetModelNames(*self*)

GetChain(*self*, *ch*)

Get a chain by name.

GetModel(*self*, *mod*)

Get a model by name.

NewChain(*self*, *name*)

Construct and add a new chain by name to the PDB. Returns the chain.

NewModel(*self*, *name*)

Construct and add a new model by name to the PDB. Returns the model.

RemoveChain(*self*, *name*)

Remove a chain from the structure (by name). Returns the chain.

RemoveModel(*self*, *name*)

Remove a model from the structure (by name). Returns the chain.

AddChain(*self*, *chainname*, *chain*)

Add a chain as a list of residues to the PDB.

AddModel(*self*, *modelname*, *model*)

Add a model as a list of residues to the PDB.

AddResidueToChain(*self*, *chain*, *res*)

Add a residue to a chain. Deprecated; use chain class function.

AddResidueToModel(*self*, *model*, *res*)

Add a residue to a model. Deprecated; use model class function.

AddRemark(*self*, *remark*)

Add a remark (note/comment) to the structure/PDB file.

GetRemarks(*self*)

Return all remarks from the PDB as a list of strings.

CheckComplete(*self*)

For every chain, check to see if every residue is complete (see aa_list dictionary).

view(*self*, *istrajectory*=`False`)

View the structure in a Pymol window. Requires an installation of Pymol.

ViewStructure(*self*)

WriteFile(*self*, *filename*)

Write this PDB structure as a single PDB file.

read(*self*, *filename*)

Alias for ReadFile().

ReadFile(*self*, *filename*)

Read a PDB file. Populate this PDBstructure.

ChainAsFASTA(*self*, *chain*)

Return the chain as a FASTA. Deprecated; use chain class function.

ModelAsFASTA(*self*, *model*)

Return the model as a FASTA. Deprecated; use chain class function.

tmscore(*self*, *fasta*, *chains*=`None`, *native*=`None`, *CA*=`True`)

Get the TMscore between two chains. Requires a FASTA alignment and a value for the length of the native structure (e.g., for a pairwise alignment, the length of the structure used as a reference before alignment was done). The latter is computed by assuming the first of both provided chains is the native structure; otherwise, uses a provided chain name (native input).

gdt(*self*, *fasta*, *chains*=`None`, *distcutoffs*=`[1, 2, 4, 8]`, *CA*=`True`)

Get the GDT score between two chains. Requires a FASTA alignment.

rmsd(*self*, *fasta*, *chains*=`None`, *CA*=`True`)

Get the RMSD between chains. Requires a FASTA alignment.

rrmsd(*self*, *fasta*, *chains*=`None`, *CA*=`True`)

Get the RRMSD between chains. Requires a FASTA alignment. See Betancourt & Skolnick, "Universal Similarity Measure for Comparison Protein Structures".

GetAverage(*self*, *chains*=`None`, *newname*=`None`)

Acquire a new chain or model corresponding to an average of all present chains or models specified.

RadiusOfGyration(*self*, *chains*=`None`)

Acquire the radius of the gyration of the entire, or a portion of, the PDB protein molecule.

GetAllCentroid(*self*, *chain*)

Populates the centroids of all residues.

IndexSeq(*self*, *chain*, *fst*)

Store in residues the correct index to the fasta. Requires a 1-to-1 correspondance at least a portion of the way through. Deprecated; use GetFASTAIndices().

GetFASTAIndices(*self*, *thing*, *fst*)

Given a PDBchain, find 1-to-1 correspondances between it and a FASTA sequence.

IterResiduesFor(*self*, *chains*=`None`)

Produce an iterator to allow one to iterate over all residues for a subset of the structure.

IterAllResidues(*self*)

Produce an iterator to allow one to iterate over all possible residues.

gm(*self*, *fasta*, *chains*=`None`, *CA*=`False`, *typeof*=`'str'`)

Acquire Geometric Morphometric data corresponding to the (x,y,z) coordinates between all homologous residue positions. Requires a FASTA alignment. Options include using alpha-carbon positions. By default, uses centroids of residues. Returns a list of labels and a list of coordinates as raw GM data. The typeof option provides an option for coordinate output; they are returned as a semicolon-delimited string (str) or as a numpy 2d array (matrix).

WriteGM(*self*, *fasta*, *gm*, *chains*=`None`, *CA*=`False`)

Write the information present in this PDB between multiple chains as a Geometric Morphometric text file. This file will be formatted such that individual lines correspond to chains and semi-colons separate the (x,y,z) coordinates between all homologous residue positions. Requires a FASTA alignment. Options include using alpha-carbon positions. By default, uses centroids of residues.

WriteLandmarks(*self*, *fasta*, *lm*, *chains*=`None`)

Write the information present in this PDB between multiple chains as a landmark text file. This file will be formatted such that the file is partitioned in sections starting with chain names and individual lines in these correspond to homologous residue positions denoted by homologous position, residue number, and residue name tab-delimited. Requires a FASTA file.

FDmatrix(*self*, *fasta*, *chains*=`None`, *scaled*=`True`)

Compute the form difference matrix (FDM) as explained in Claude 2008. It relates to the identification of the most influential residue, with respect to the overall shape/structure. If the scaled option is True, will return an scaled list (from -1 to 1) of the of lenght equal to the number of residues. Otherwise will return the raw FDM, rounded so it can be included in a PDB. The scaled version is better for vizualization. By default the FDM is computed far all chains, but a subset can be passed to the chains option.

Map2Protein(*self*, *outname*, *lis*, *chain*, *fasta*)

Map a list of values (lis), that must have a lenght equal to that of the number of residues in the PDB to be mapped (chain). If a list of list is provided, the first list will be mapped as the beta factor and the second as occupancy

Contacts(*self*, *chain*=`None`, *thres*=`4.5`)

Compute the contact map of all chains or a chain.

:param chain: A list of chain or model names or a single string or integer. By default, entire structure. :param thres: A threshold for distinguishing contact in Angstroms. :returns: A list of tuples of indices (integers) which correspond to chains or models and their residues.

WriteContacts(*self*, *filename*)

Write contact map.

___iter___(*self*)

Returns all PDBchains as an iterator.

___len___(*self*)

Returns the length of the protein.

___str___(*self*)

As a string, outputs structure in the PDB format.

Overrides: object.___str___

Inherited from object

___delattr___(), ___format___(), ___getattribute___(), ___hash___(), ___new___(), ___reduce___(), ___reduce_ex___(), ___repr___(), ___setattr___(), ___sizeof___(), ___subclasshook___()

7.7.2 Properties

Name	Description
chains	
contactmap	
filepath	
handle	
ismodel	
models	
mutation	
orderofchains	
orderofmodels	
organism	
remarks	
taxid	
Inherited from object ___class___	

7.8 Class PDBfile

object

labblouin.PDBnet.PDBfile

7.8.1 Methods

___init___(*self*, *fi*)

x.___init___(...) initializes x; see help(type(x)) for signature

Overrides: object.___init___ extit(inherited documentation)

isLargeFile(*self*)

Return whether or on the PDB file this object represents is incredibly large.

close(*self*)

Close the file.

read(*self*)

Acquire all remarks, and the indices of all models and residues. Returns remarks, and biological source information as a tuple (remarks, organism, taxid, mutant).

hasResidue(*self*, *chain*, *res*, *model*=`-1`)

Determine if a residue is present in the PDBfile.

readResidue(*self*, *chain*, *res*, *model*=`-1`)

Parse a residue from the PDB file and return a PDBresidue.

hasModels(*self*)

getModelNames(*self*)

getChainNames(*self*)

iterResidueData(*self*)

Yield the model number, chain name, and residue number for all residues present in this PDB file, not necessarily in order.

getResidueNamesInChain(*self*, *ch*)

Inherited from object

___delattr___(), ___format___(), ___getattribute___(), ___hash___(), ___new___(),

___reduce___(), ___reduce_ex___(), ___repr___(), ___setattr___(), ___sizeof___(), ___str___(), ___subclasshook___()

7.8.2 Properties

Name	Description
chains	
fileHandle	
filePath	
memHandle	
modelIndices	
residueIndices	
size	
Inherited from object ___class___	

8 Module labblouin.RegExpress

Prosite pattern generator. Will turn an alignment into a regular expression

8.1 Functions

check_if_is_align(*dictionary_fasta*)

will check the read fasta to see if is an alignment

read_fasta(*prefix*)

Wll read the fasta, place sequences in a dictionary. Will break if not an alignment

strip_gaps(*matrix*)

given an alignment matrix, strip out the gap columns

reg_express_col(*col*)

given a column of an alignment, return the string corresponding to the regular expression of such column

reg_express(*fastas*, *l*)

will create a matrix with the alignment, discard columns with gaps, and create a regular expression similar to prosite

8.2 Variables

Name	Description
aa	**Value:** `{'A': 'Ala', 'C': 'Cys', 'D': 'Asp', 'E': 'Glu', 'F': 'Ph...`
allaa	**Value:** `set(['A', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'K', 'L', 'M...`
___package___	**Value:** `'labblouin'`

9 Module labblouin.SeqMask

Get the amino acid sequence from a landmark file, align this profile to more sequences if required using muscle, and mask the membership vector into this alignment.

9.1 Functions

InferSingleLettterCode(*threlettercode*)

Convert the amino acid three letter code, into a single letter code

InferSeqs_landmarks(*prefix*)

Given a landmark file, return a dictionary with the name as key and the sequence as value

dict2fasta(*outfilename*, *dic*)

Convert the dictionary return by InferSeqs_landmarks fuction and writes a fasta file. The dictionary should have the seq name as key and a string as value containing the sequence

fasta2dict(*fastafilename*)

convert a fasta file into a python dictionary, being the keys the name of the sequence and the values the sequence itself, in a list

Mem2List2tuple(*prefix*)

Read the membership vector, and returns a list and a dictionary with tupples for each module, indicating the indexes

str2list(*string*)

returns a list of a string that have no spaces between elements

MaskAln(*fastafilename*, *modulename*, *mem*, *tuples*)

Strip the module in an alignment (fasta) file

modseq2Fasta(*prefix*, *modseq*, *names*)

9.2 Variables

Name	Description
___package___	**Value:** `'labblouin'`

10 Module labblouin.expresso

An (incomplete) parsing library for output from the T-Coffee Expresso executable. expresso Python Library / Summer 2013 / Alex Safatli

E-mail: safatli@cs.dal.ca Dependencies: -

10.1 Variables

Name	Description
___package___	**Value:** `None`

10.2 Class expressoParser

Parses Expresso output.

10.2.1 Methods

___init___(*self*, *fin*)

___read___(*self*)

Read an Expresso output file; store the data inside this object.

getPDBcodes(*self*)

Return PDB code and chain as a tuple for each line.

writeNonEmpty(*self*, *fout*)

Writes non-empty PDB IDs to file.

getNonEmptySequenceNames(*self*)

Gets non-empty PDB names.

11 Module labblouin.homology

11.1 Functions

cleanFastaFile(*fname*, *targetfolder*='')

Removes gaps from target FASTA file. Allows for BLAST input using this file. Creates new files for every sequence entry. Returns the filename(s) of the new file(s).

writeModellerPIR(*pdb_files*, *seq*, *dest_file*)

writeAlignmentFromFasta(*fasta_file*, *dest_file*)

Reads a FASTA file and outputs it to a Clustal-like alignment.

writeSOAPSSAlignment(*fasta_file*, *pdb1*, *pdb2*, *dest_file*)

Reads a FASTA file and outputs to a SOAPSS alignment file.

writeFirstFromFasta(*fasta_file*, *dest_file*)

Extracts first sequence from a FASTA and writes to a new one.

extractRandomFasta(*fasta_file*, *numRandom*, *dest_file*)

Extracts a random number of sequences from a FASTA file. Writes to new FASTA.

completePDB(*pdbin*, *pdbout*)

Given a PDB, clean/complete the PDB using Modeller's complete_pdb function.

getRandomPDBFragment(*pdb*, *k*=100)

Given a PDB, get a random fragment of length k.

getFirstResidueNumber(*pdb*)

Gets the FASTA residue number of the first ATOM in a PDB file.

checkForFileCollision(*path*, *ext*)

Ensures a file is not being overwritten. Will recursively keep adding a '0' to end of base filename until a unique filename is found. Returns the path.

system(*instruction*)

Executes a system command line instruction. Returns list of stdout, stderr.

cleanModellerFolder(*path*)

Moves intermediate PDBs to another subfolder in the Modeller folder output from Biskit.

getModellerPDB(*workflowfolder*, *targetfile*)

Copies the model_00.pdb file from Modeller if present in given workflow directory from homologyWorkflow.

compareStructures(*foldername1*, *foldername2*, *outpath*, *verbose*=`False`)

Given 2 folders with PDB files, calculates the RMSDs with all possible pairwise combinations of PDBs; writes to file located at path determined by outpath.

startPymol()

Ensures Pymol has been launched.

radiusOfGyration(*pdbFile*, *chain*, *pdb*=`None`)

Returns the approximate radius of gyration of a given PDB molecule. Deprecated; now in PDBnet.

tmscore(*alignPDB*, *alignFASTA*)

Returns the TMscore and associated P-value. See Xu & Zhang, "How significant is a protein structure similarity with TM-score = 0.5?".

rrmsd(*alignPDB*, *alignFASTA*, *rmsd*=`False`, *chains*=`None`)

Returns the rRMSD of an aligned pairwise PDB and FASTA file. See Betancourt & Skolnick, "Universal Similarity Measure for Comparison Protein Structures". Deprecated.

rmsd(*pdbFile1*, *pdbFile2*)

Returns the RMSD of two PDB files. Requires: Pymol.

11.2 Variables

continued on next page

Name	Description

Name	Description
___package___	**Value:** `'labblouin'`

11.3 Class manualModeller

Set up and run Modeller using a given folder of PDB files and a target FASTA sequence.

11.3.1 Methods

___init___(*self*, *outfolder*, *targetfile*, *templatefldr*)

___buildList___(*self*)

Build a string list of all PDB files.

___prepare___(*self*)

Prepare input for Modeller.

run(*self*)

Run, call Modeller.

postProcess(*self*)

Make a copy of all found models and rename.

11.4 Class fasta

Open, read, and organize a FASTA file as an object.

11.4.1 Methods

___init___(*self*, *fname*, *removeGaps*=`True`, *allUpper*=`True`)

read(*self*, *rgaps*, *allup*)

12 Module labblouin.homstrad

A library to manage interfacing with the Homstrad database raw files.

homstrad Python Library / May 22, 2013 / Alex Safatli

E-mail: safatli@cs.dal.ca Dependencies: IO (LabBlouinTools)

12.1 Functions

hasNoFolders(*path*)

12.2 Variables

Name	Description
___package___	**Value:** `'labblouin'`

12.3 Class homstradFolder

Model a single Homstrad folder or set.

12.3.1 Methods

___init___(*self*, *foldrname*)

___manifest___(*self*)

getNames(*self*)

getNumSequences(*self*)

getSequences(*self*)

getFiles(*self*)

getPath(*self*)

getFASTA(*self*)

writeFASTA(*self*, *fi*, *names*=`None`)

getFASTAfor(*self*, *names*)

getAlignedPDB(*self*)

getPDBs(*self*)

getPDBfor(*self*, *name*)

getSequenceLength(*self*)

getAlignmentLength(*self*)

12.4 Class homstradDatabase

12.4.1 Methods

___init___(*self*, *dbpath*, *traverse*=`True`)

___iter___(*self*)

traverse(*self*)

getPath(*self*)

getFolders(*self*)

getFailedCount(*self*)

getSucceededCount(*self*)

13 Module labblouin.loadingBar

13.1 Variables

Name	Description
___package___	**Value:** `'labblouin'`

13.2 Class loadingBar

13.2.1 Methods

___init___(*self*, *title*, *width*=300, *height*=25)

open(*self*)

close(*self*)

update(*self*, *ratio*)

settitle(*self*, *title*)

14 Module labblouin.logfile

14.1 Functions

parseTime(*time*)

calculateTime(*timevector*)

14.2 Variables

Name	Description
___package___	**Value:** `'labblouin'`

14.3 Class XMLfile

14.3.1 Methods

___init___(*self*, *xmlf*, *root*=`'root'`)

compile(*self*)

read(*self*)

write(*self*)

add(*self*, *ele*, *it*, **args*)

14.4 Class logfile

14.4.1 Methods

___init___(*self*, *logf*, *enable*=`True`, *silent*=`False`)

setTotalNum(*self*, *tn*)

incrementTimer(*self*)

updateTimer(*self*, *numat*)

___fs___(*self*, *tot*)

writeTemporary(*self*, *msg*, *silent*=`None`)

write(*self*, *msg*, *silent*=`None`)

writeElapsedTime(*self*)

15 Module labblouin.mdsa

A library to manage interfacing with the MDSA database raw files (http://dna.cs.byu.edu/mdsas/index.shtr

mdsa Python Library / Oct 8, 2014 / Alex Safatli

E-mail: safatli@cs.dal.ca Dependencies: IO, FASTAnet (LabBlouinTools)

15.1 Variables

Name	Description
mdsaTypes	**Value:** `['smart', 'balibase', 'oxbench', 'prefab']`
___package___	**Value:** `'labblouin'`

15.2 Class mdsaAlignment

Model a single MDSA alignment.

15.2.1 Methods

___init___(*self*, *finame*)

getObject(*self*)

getNames(*self*)

getNumSequences(*self*)

getSequences(*self*)

getPath(*self*)

writeFASTA(*self*, *fi*, *names*=`None`)

getFASTAfor(*self*, *names*)

getSequenceLength(*self*)

getAlignmentLength(*self*)

15.3 Class mdsaDatabase

15.3.1 Methods

___init___(*self*, *dbpath*, *traverse*=`True`)

___iter___(*self*)

traverse(*self*)

getPath(*self*)

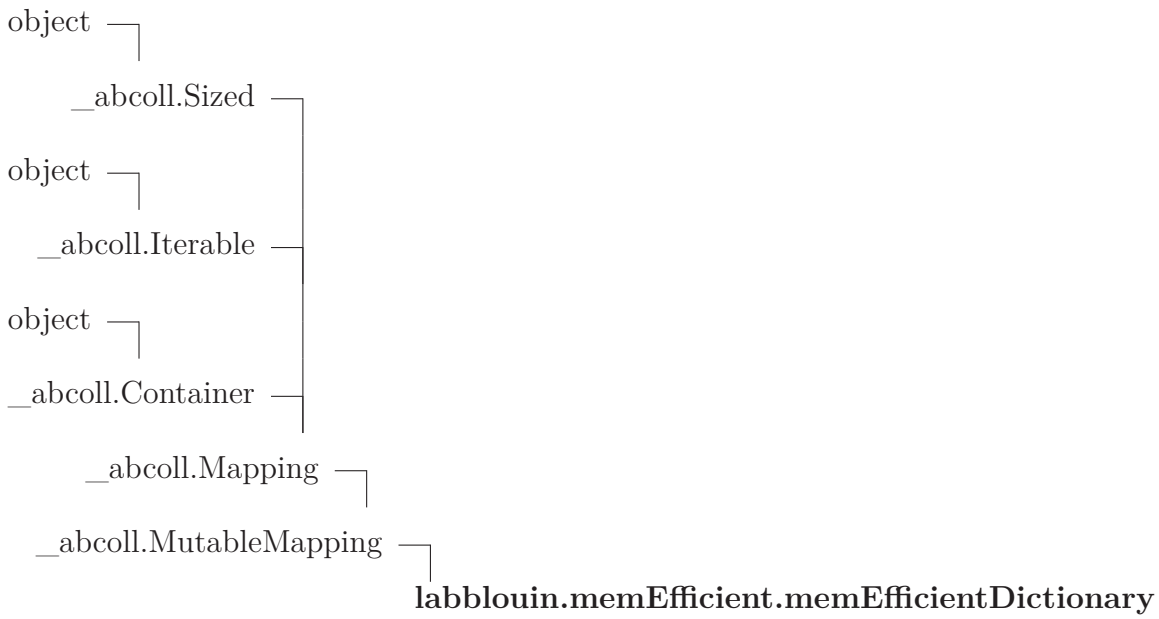
getFiles(*self*)

16 Module labblouin.memEfficient

16.1 Variables

Name	Description
DICT_START_NUM	**Value:** `8`
DICT_FREE_SLOT	**Value:** `-1`
DICT_DUMMY_SLOT	**Value:** `-2`
DICT_SHIFT	**Value:** `5`
__package__	**Value:** `'labblouin'`

16.2 Class memEfficientDictionary



A memory-efficient alternative to the built-in Python dictionary with a trade-off for (insertion) performance. Based on recipe found by Raymond Hettinger (http://code.activestate.com/recipes/578 proof-of-concept-for-a-more-space-efficient -faster/).

16.2.1 Methods

__init__(*self*, **args*, ***kwds*)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

___len___(*self*)

Overrides: _abcoll.Sized.___len___

___iter___(*self*)

Overrides: _abcoll.Iterable.___iter___

iterkeys(*self*)

Return Value

an iterator over the keys of D

Overrides: _abcoll.Mapping.iterkeys extit(inherited documentation)

keys(*self*)

Return Value

list of D's keys

Overrides: _abcoll.Mapping.keys extit(inherited documentation)

itervalues(*self*)

Return Value

an iterator over the values of D

Overrides: _abcoll.Mapping.itervalues extit(inherited documentation)

values(*self*)

Return Value

list of D's values

Overrides: _abcoll.Mapping.values extit(inherited documentation)

iteritems(*self*)

Return Value

an iterator over the (key, value) items of D

Overrides: _abcoll.Mapping.iteritems extit(inherited documentation)

items(*self*)

Return Value

list of D's (key, value) pairs, as 2-tuples

Overrides: _abcoll.Mapping.items extit(inherited documentation)

___contains___(*self*, *key*)

Overrides: _abcoll.Container.___contains___

get(*self*, *key*, *default*=`None`)

d defaults to None.

Return Value

D[k] if k in D, else d

Overrides: _abcoll.Mapping.get extit(inherited documentation)

popitem(*self*)

as a 2-tuple; but raise KeyError if D is empty.

Return Value

(k, v), remove and return some (key, value) pair

Overrides: _abcoll.MutableMapping.popitem extit(inherited documentation)

clear(*self*)

Clear the data structure.

Return Value

None

Overrides: _abcoll.MutableMapping.clear

___getitem___(*self*, *key*)

Access an item through the [] notation.

Overrides: _abcoll.Mapping.___getitem___

___setitem___(*self*, *key*, *value*)

Overrides: _abcoll.MutableMapping.___setitem___

___delitem___(*self*, *key*)

Overrides: _abcoll.MutableMapping.___delitem___

Inherited from _abcoll.MutableMapping

pop(), setdefault(), update()

Inherited from _abcoll.Mapping

___eq___(), ___ne___()

Inherited from _abcoll.Sized

___subclasshook___()

Inherited from object

___delattr___(), ___format___(), ___getattribute___(), ___new___(), ___reduce___(), ___reduce_ex___(), ___repr___(), ___setattr___(), ___sizeof___(), ___str___()

16.2.2 Properties

Name	Description
Inherited from object	
___class___	

16.2.3 Class Variables

Name	Description
___abstractmethods___	**Value:** `frozenset([])`
Inherited from _abcoll.Mapping	
___hash___	

17 Module labblouin.passToqsub

17.1 Functions

returnScript(*command*, *jobname*=`'default_job'`, *np*=`1`, *npmin*=`1`, *nomail*=`True`, *sync*=`False`)

17.2 Variables

Name	Description
___package___	**Value:** `'labblouin'`

18 Module labblouin.pdbCompare

18.1 Functions

getChecksum(*pdb_file*)

Given a PDB file, retrieve a unique checksum.

pdbCrosscheck(*folder1*, *folder2*)

Given two folder paths, determine what PDBs are unique amongst the two

outputResults(*pdbcrosscheck_results*, *fpath*=`None`)

18.2 Variables

Name	Description
___package___	**Value:** `'labblouin'`
___warningregistry___	**Value:** `{('the md5 module is deprecated; use hashlib instead', <t...`

19 Module labblouin.pfam

19.1 Functions

extractPDBSequences(*pdb_file*)

Extracts the Amino Acid sequences for a protein structure defined by a PDB file and returns as a dictionary.

extractPDBChain(*pdb_file*, *ch*, *dest_file*)

Extracts a PDB file from another that has only information for a single chain.

writeSequencesToFile(*seq_list*, *targetfolder*, *fasta*=`True`)

Takes in a dictionary of sequences and writes them to seperate .seq or .fasta files.

writeSequenceToFasta(*seq*, *seq_id*, *targetpath*)

Creates FASTA file from given sequence string.

writeSequencesToFasta(*seq_list*, *dest_file*)

Creates single FASTA file from given sequence tuple list where first item is sequence ID, second item is the sequence.

doSequenceSearch(*seq_file*)

Performs the PFAM search for a sequence file.

processSequenceSearch(*search_results*)

Takes in a XML result returned from a sequence search and parses it in order to determine what the most significant results are for related families. Returns a list of the family PFAM IDs.

downloadFamilySequences(*pfam_family_id*, *dest_folder*, *atype*=`'seed'`)

Takes in PFAM family ID and acquires the gzipped flat file filled with seed or full sequence allignments.

decompressGzipFile(*gzip_file_name*, *destfolder*, *file_ext*=`'ann'`)

Takes in GZIP file from PFAM or PDB and decompresses it to an .ann or specified file extension.

readPfamFile(*file_name*)

Takes in PFAM Stockholm file and reads it, returning a pfamFile object.

grabNCBIAccessionMetadata(*accession_id*)

Takes in an accession ID and returns NCBI metadata from GenBank.

grabPDBFile(*pdb_code*, *dest_folder*)

Takes in a PDB code and acquires the PDB file from the PDB database, placing it in the specified destination folder.

printListToFile(*list_in*, *dest*)

Prints a list to a file.

list2txttable(*list_in*, *title*)

Converts a list to a string table.

removeListDuplicates(*list_in*)

Removes all duplicates in a list.

19.2 Variables

Name	Description
__package__	**Value:** `'labblouin'`

19.3 Class pfamFile

19.3.1 Methods

__init__(*self*, *filepath*)

parse(*self*)

cluster(*self*)

20 Module labblouin.plotGM

this will plot a gm of a 2D shape

20.1 Variables

Name	Description
___package___	**Value:** `'labblouin'`

21 Module labblouin.psub

Run as many of a input set of jobs/commands as there are cores at any given time (i.e. emulate a queue such as qsub present on GRID-powered hardware).

21.1 Functions

call(*instr*)

21.2 Variables

Name	Description
___package___	**Value:** `'labblouin'`

22 Module labblouin.sabmark

22.1 Variables

Name	Description
___package___	**Value:** `'labblouin'`

22.2 Class sabmarkFile

22.2.1 Methods

___init___(*self*, *path*, *l*, *fp*)

22.3 Class sabmarkSummary

22.3.1 Methods

___init___(*self*, *path*)

___read___(*self*)

22.4 Class sabmarkFolder

22.4.1 Methods

___init___(*self*, *foldrname*)

___manifest___(*self*)

22.5 Class sabmarkDatabase

22.5.1 Methods

___init___(*self*, *dbpath*)

traverse(*self*)

groups(*self*, *fkey*)

23 Module labblouin.scop

23.1 Variables

Name	Description
SCOP_VERSION	**Value:** `'1.75'`
SCOP_PARSE_FILE	**Value:** `'http://scop.mrc-lmb.cam.ac.uk/scop/parse/dir.des.scop.tx.`
SCOP_PDBDB_FILE	**Value:** `'http://scop.mrc-lmb.cam.ac.uk/scop/parse/dir.cla.scop.tx.`
LOCAL_PARSE_FILE	**Value:** `'directory.scop'`
LOCAL_PDBDB_FILE	**Value:** `'pdbdb.scop'`
LOCAL_FLAT_FILE	**Value:** `'scop.pickl'`
__package__	**Value:** `'labblouin'`

23.2 Class scopHierarchy

23.2.1 Methods

__init__(*self*, *cacheloc*)

__downloadparse__(*self*)

__downloadpdbdb__(*self*)

__parsefile__(*self*, *fi*)

__pdbdbfile__(*self*, *fi*)

__readpickle__(*self*, *fi*)

__writepickle__(*self*, *fi*)

populateHierarchy(*self*)

getDissimilar(*self*, *suprfquery*, *num*=`100`)

query(*self*, *dicti*, *q*)

getSimilar(*self*, *suprfquery*, *num*=100)

getFamilies(*self*, *suprfquery*)

23.2.2 Class Variables

Name	Description
classes	**Value:** {}
folds	**Value:** {}
superfamilies	**Value:** {}
families	**Value:** {}
domains	**Value:** {}
species	**Value:** {}
entries	**Value:** {}

23.3 Class scopItem

23.3.1 Methods

___init___(*self*, *sunid*, *sccs*, *shortname*, *desc*)

addPDB(*self*, *pdb*)

getPDBs(*self*)

24 Module labblouin.svm

24.1 Variables

Name	Description
___package___	**Value:** `'labblouin'`

24.2 Class svm_model

24.2.1 Methods

___init___(*self*, *data*=`[]`, *labels*=`[]`, *kernel*=`2`, *c*=`10`)

optimize(*self*, *low*=`1`, *up*=`50`, *steps*=`5`, *by*=`'proportions'`)

Optimize c-value on the basis of by (default:proportions of correct classifications).

predict(*self*, *data*)

load(*self*, *fin*)

save(*self*, *fin*)

24.3 Class cross_validate

24.3.1 Methods

___init___(*self*, *model*, *test*=`[]`)

perform(*self*)

Perform the cross-validation.

25 Module labblouin.timer

25.1 Functions

getTime()

estimateRemainingTime(*starttime*, *numAt*, *totalNum*)

25.2 Variables

Name	Description
___package___	**Value:** `'labblouin'`

Index