Pylogeny

API Documentation

January 22, 2015

Contents

1 Package pylogeny

Pylogeny is a Python library and code framework for phylogenetic tree reconstruction and scoring.

Allows one to perform the following tasks: (1) Generate and manage phylogenetic tree landscapes. (2) Build and rearrange phylogenetic trees using preset operators such as NNI, SPR, and TBR. (3) Score phylogenetic trees by Log-likelihood and Parsimony.

Dependencies: Pandas, P4 Phylogenetic Library. Suggested: FastTree, RAxML, PytBEAGLEhon.

1.1 Modules

2 Module pylogeny.JSONWriter

Serialize a phylogenetic landscape into a JSON object.

2.1 Variables

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

2.2 Class JSONWriter

object
└
pylogeny.landscapeWriter.landscapeWriter
└
pylogeny.JSONWriter.JSONWriter

2.2.1 Methods

___init___(*self*, *ls*, *name*)

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

Overrides: object.___init___ extit(inherited documentation)

nodeToJSON(*self*, *node*)

getOnlyImprovements(*self*, *groups*=`None`)

getCompleteLandscape(*self*)

getJSON(*self*)

Inherited from pylogeny.landscapeWriter.landscapeWriter(Section 10.2)

writeFile()

Inherited from object

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

2.2.2 Properties

Name	Description
Inherited from object	
__class__	

3 Module pylogeny.___version___

3.1 Variables

Name	Description
VERSION	**Value:** `'0.3.6.9'`
___package___	**Value:** `None`

4 Module pylogeny.alignment

Handle input biological sequence alignment files for the purposes of phylogenetic inference. Will read all types of alignment files by utilizing the P4 python phylogenetic library.

4.1 Variables

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

4.2 Class alignment

object
└
pylogeny.alignment.alignment

Known Subclasses: pylogeny.alignment.phylipFriendlyAlignment

Wrap a biological sequence alignment to enable functionality necessary for phylogenetic inference. Makes use of temporary files; requires to be closed once no longer needed.

4.2.1 Methods

___init___(*self*, *inal*=`None`)

Instantiate an object intended to wrap an alignment for the purposes of running phylogenetic inference.

:param inal: An alignment file path (most formats are accepted).

Overrides: object.___init___

___getitem___(*self*, *i*)

___str___(*self*)

str(x)

Overrides: object.___str___ extit(inherited documentation)

___len___(*self*)

___iter___(*self*)

close(*self*)

Delete all temporary files and clear data.

toStrList(*self*)

Get all sequences as a list of strings.

getStateModel(*self*)

getSize(*self*)

Return the size of the alignment, or how many characters there are in each respective item in the alignment.

getNumSeqs(*self*)

Return the number of sequences that are present in the sequence alignment.

getDim(*self*)

Return the dimensionality of the sequence alignment (how many different types of characters).

getSequence(*self*, *i*)

Acquire the ith sequence.

getFASTA(*self*)

Get (and create if not already) a path to a temporary FASTA file. This will be deleted upon closure of the alignment instance.

getApproxMLNewick(*self*)

Get a tree in newick format via use of FastTree that serves as an approximation of the maximum likelihood tree for this data.

getApproxMLTree(*self*)

Get a tree object for an approximation of the maximum likelihood tree for this data using FastTree.

getFastTreeNewick(*self*)

Alias for the "getApproxMLNewick()" function.

getTaxa(*self*)

Return taxa names.

getAlignment(*self*)

Acquire the alignment data structure (P4 module).

bootstrap(*self*)

Perform bootstrapping on the alignment data.

Inherited from object

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

4.2.2 Properties

Name	Description
Inherited from object	
___class___	

4.3 Class phylipFriendlyAlignment

object

pylogeny.alignment.alignment

pylogeny.alignment.phylipFriendlyAlignment

An alignment object that renames all comprising taxa in order to be able to be written as a strict Phylip file.

4.3.1 Methods

___init___(*self*, *inal*=`None`)

Instantiate an object intended to wrap an alignment for the purposes of running phylogenetic inference.

:param inal: An alignment file path (most formats are accepted).

Overrides: object.___init___ extit(inherited documentation)

getPhylip(*self*)

Get a path to a temporary Phylip file. This will be deleted upon closure of the alignment instance.

writeProperNexus(*self*, *wri*)

Write a Nexus file with proper names.

reassignFromReinterpretedNewick(*self*, *tr*)

Replace all proper names with reassigned names in a Newick tree.

reinterpretNewick(*self*, *tr*)

Replaces all reassigned names to proper names in a Newick tree.

getProperName(*self*, *n*)

Return the actual name for an integer-based sequence name that was reassigned at initialization.

getTaxa(*self*)

Return current taxa names in the alignment.

Overrides: pylogeny.alignment.alignment.getTaxa

recreateObject(*self*)

Reintializes the object.

Inherited from pylogeny.alignment.alignment(Section 4.2)

___getitem___(), ___iter___(), ___len___(), ___str___(), bootstrap(), close(), getAlignment(), getApproxMLNewick(), getApproxMLTree(), getDim(), getFASTA(), getFastTreeNewick(), getNumSeqs(), getSequence(), getSize(), getStateModel(),

toStrList()

Inherited from object

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

4.3.2 Properties

Name	Description
Inherited from object	
___class___	

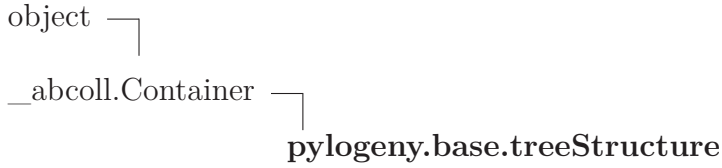
5 Module pylogeny.base

Definitions for generalized containers and objects used by other structures in this framework.

5.1 Variables

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

5.2 Class treeStructure



Known Subclasses: pylogeny.base.trie, pylogeny.rearrangement.topology

Defines a base collection of treeNodes and treeBranches in a hierarchical tree structure.

5.2.1 Methods

___init___(*self*, *root*=`None`)

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

Overrides: object.___init___ extit(inherited documentation)

___contains___(*self*, *x*)

Determines whether a node is found in the tree structure.

Overrides: _abcoll.Container.___contains___

getRoot(*self*)

Return the top-level, root, node of the tree.

leaves(*root*)

getAllLeaves(*self*)

nodes(*root*)

getAllNodes(*self*)

postOrderTraversal(*root*)

getPostOrderTraversal(*self*)

___str___(*self*)

Returns a string representation of the tree.

Overrides: object.___str___

Inherited from _abcoll.Container

___subclasshook___()

Inherited from object

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

5.2.2 Properties

Name	Description
Inherited from object	
___class___	

5.2.3 Class Variables

Name	Description
root	**Value:** `None`
___abstractmethods___	**Value:** `frozenset([])`

5.3 Class treeNode

object

pylogeny.base.treeNode

Known Subclasses: pylogeny.base.trieNode, pylogeny.newick.node

A node in a tree.

5.3.1 Methods

___init___(*self*, *lbl*=`None`, *children*=`None`, *parent*=`None`)

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

Overrides: object.___init___ extit(inherited documentation)

getLabel(*self*)

getParent(*self*)

addChild(*self*, *item*)

getChildByIndex(*self*, *i*)

getChildren(*self*)

isLeaf(*self*)

isInternalNode(*self*)

Inherited from object

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

5.3.2 Properties

Name	Description
Inherited from object	
___class___	

5.3.3 Class Variables

Name	Description
label	**Value:** `None`
parent	**Value:** `None`
children	**Value:** `None`

5.4 Class treeBranch

object
└ **pylogeny.base.treeBranch**

Known Subclasses: pylogeny.newick.branch

A branch in a tree.

5.4.1 Methods

___init___(*self*, *parent*=`None`, *child*=`None`, *label*=`''`)

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

Overrides: object.___init___ extit(inherited documentation)

getLabel(*self*)

getParent(*self*)

getChild(*self*)

Inherited from object

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

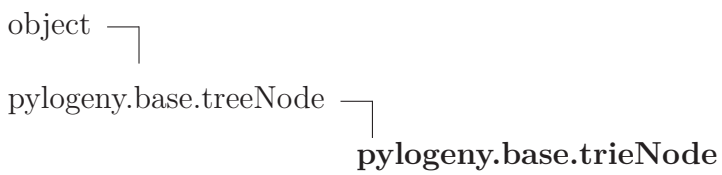
5.4.2 Properties

Name	Description
Inherited from object	
___class___	

5.4.3 Class Variables

Name	Description
label	**Value:** `''`
parent	**Value:** `None`
child	**Value:** `None`

5.5 Class trieNode



A subclass of treeNode that allows for checking non-zero members amongst children branches and other conveniences.

5.5.1 Methods

getParentNode(*self*)

setChildNode(*self*, *child*, *newchild*)

iterNonEmptyChildrenNodes(*self*)

Iterate over all children that are not empty.

getNonEmptyChildrenNodes(*self*)

Acquire a list of all non-empty children.

getNonEmptyChildrenBranches(*self*)

Acquire a list of all non-empty children.

getNonEmptyChildrenBranchLabels(*self*)

numEmptyChildrenNodes(*self*)

Acquire the number of children nodes that are marked 0 or nonexistent.

Inherited from pylogeny.base.treeNode(Section 5.3)

___init___(), addChild(), getChildByIndex(), getChildren(), getLabel(), getParent(), isInternalNode(), isLeaf()

Inherited from object

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

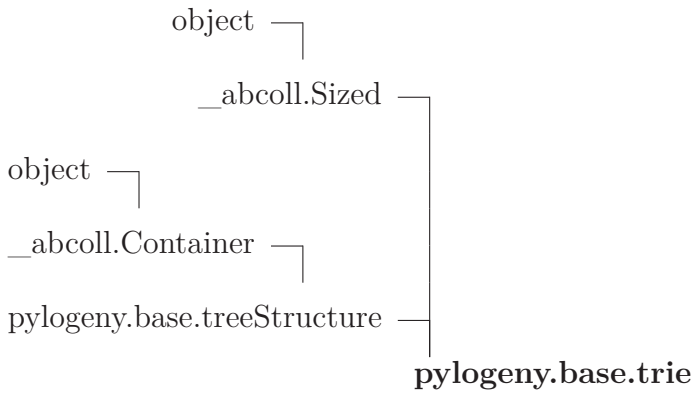
5.5.2 Properties

Name	Description
Inherited from object	
___class___	

5.5.3 Class Variables

Name	Description
Inherited from pylogeny.base.treeNode (Section 5.3)	
children, label, parent	

5.6 Class trie



Known Subclasses: pylogeny.base.patriciaTree

Defines a trie across a range of strings.

5.6.1 Methods

___init___(*self*)

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

Overrides: object.___init___ extit(inherited documentation)

___contains___(*self*, *x*)

Implementing for interface (Container).

Overrides: _abcoll.Container.___contains___

___len___(*self*)

Implementing for interface (Sized).

Overrides: _abcoll.Sized.___len___

getAlphabet(*self*)

getRoot(*self*)

Return the top-level, root, node of the tree.

Overrides: pylogeny.base.treeStructure.getRoot extit(inherited documentation)

search(*self*, *seq*)

Search for a sequence in the trie. Returns true if it exists.

insert(*self*, *seq*)

Dynamically insert a sequence into the trie.

delete(*self*, *seq*)

Remove a sequence from the trie. Will not remove added characters to alphabet.

Inherited from _abcoll.Sized

___subclasshook___()

Inherited from pylogeny.base.treeStructure(Section 5.2)

___str___(), getAllLeaves(), getAllNodes(), getPostOrderTraversal(), leaves(), nodes(), postOrderTraversal()

Inherited from object

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

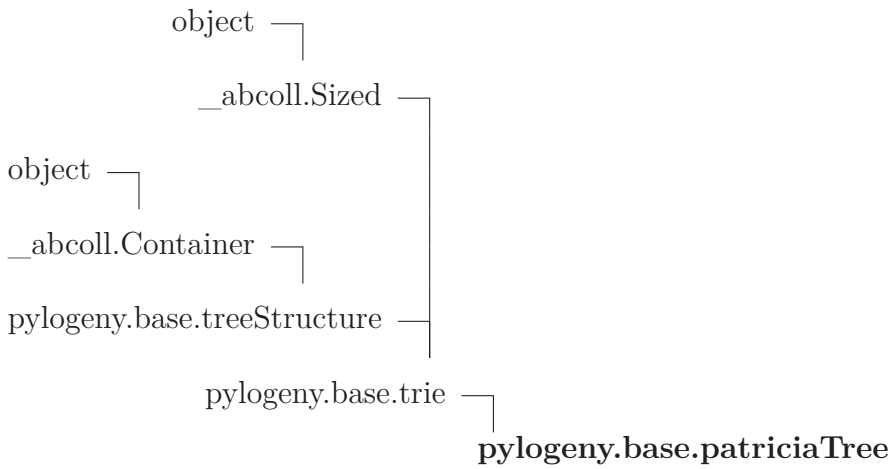
5.6.2 Properties

Name	Description
Inherited from object	
___class___	

5.6.3 Class Variables

Name	Description
alphabet	**Value:** `None`
root	**Value:** `None`
count	**Value:** `0`
nextLabel	**Value:** `1`
__abstractmethods__	**Value:** `frozenset([])`

5.7 Class patriciaTree



Defines a PATRICIA tree (condensed trie) across a range of strings.

5.7.1 Methods

__contains__(*self*, *x*)

Implementing for interface (Container).

Overrides: _abcoll.Container.__contains__

search(*self*, *seq*)

Search for a sequence in the PATRICIA tree. Returns its position in addition sequence if it exists. Else, returns 0.

Overrides: pylogeny.base.trie.search

insert(*self*, *seq*)

Dynamically insert a sequence into the PATRICIA tree. Returns the unique index in the tree for that string.

Overrides: pylogeny.base.trie.insert

delete(*self*, *seq*)

Remove a sequence from the PATRICIA tree. Will not remove added characters to alphabet.

Overrides: pylogeny.base.trie.delete

Inherited from pylogeny.base.trie(Section 5.6)

___init___(), ___len___(), getAlphabet(), getRoot()

Inherited from _abcoll.Sized

___subclasshook___()

Inherited from pylogeny.base.treeStructure(Section 5.2)

___str___(), getAllLeaves(), getAllNodes(), getPostOrderTraversal(), leaves(), nodes(), postOrderTraversal()

Inherited from object

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

5.7.2 Properties

Name	Description
Inherited from object ___class___	

5.7.3 Class Variables

Name	Description
Inherited from pylogeny.base.trie (Section 5.6) ___abstractmethods___, alphabet, count, nextLabel, root	

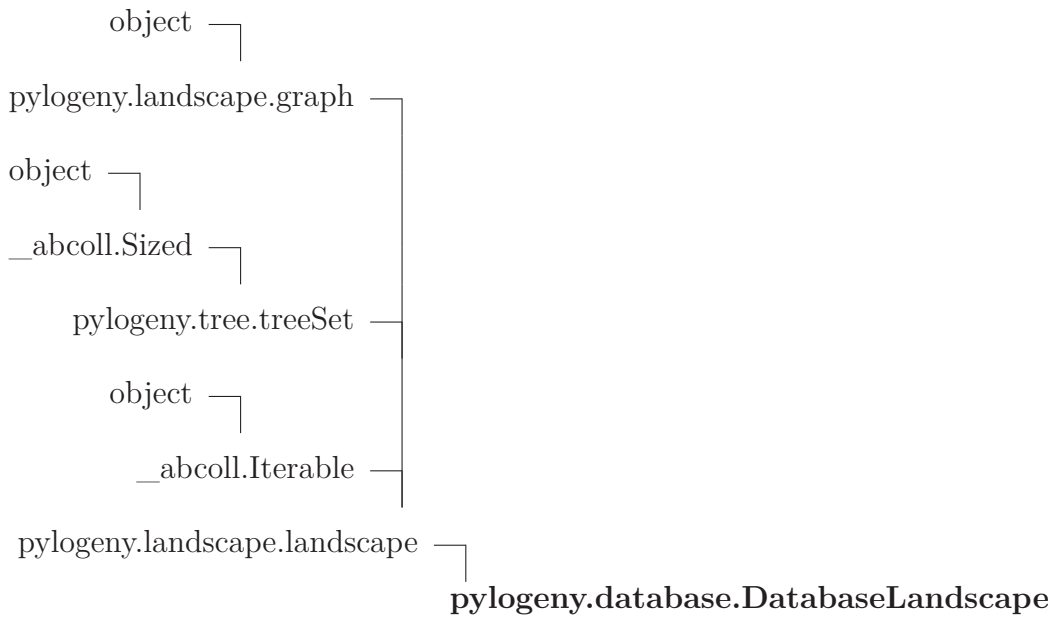
6 Module pylogeny.database

Connect, access, + manipulate external tree data from a remote SQL server or from a sqlite file.

6.1 Variables

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

6.2 Class DatabaseLandscape



Known Subclasses: pylogeny.database.SQLExhaustiveLandscape

Abstract the landscape to one comprising a landscape.

6.2.1 Methods

getNode(*self*, *i*)

Overrides: pylogeny.landscape.graph.getNode

Inherited from pylogeny.landscape.landscape(Section 9.3)

___getitem___(), ___init___(), ___iter___(), ___str___(), addTree(), exploreRandomTree(), exploreTree(), findTree(), findTreeTopology(), findTreeTopologyByStructure(), getAlignment(), getAllPathsOfBestImprovement(), getBestImprovement(), getBipartitionFoundInTreeByIndex(), getGlobalOptimum(), getLocalOptima(), getLocks(), getNumberTaxa(), getPathOfBestImprovement(), getPossibleNumberRootedTrees(), getPossibleNumberUnrootedTrees(), getRoot(), getRootTree(), getTree(), getVertex(), indexOf(), isLocalOptimum(), isViolating(), iterAllPathsOfBestImprovement(), iterTrees(), lockBranchFoundInTree(), lockBranchFoundInTreeByIndex(), removeTree(), setAlignment(), toProperNewickTreeSet(), toTreeSet(), toggleLock()

Inherited from pylogeny.landscape.graph(Section 9.2)

___len___(), clearEdgeWeights(), getCenter(), getCliqueNumber(), getCliques(), getCliquesOfNode(), getComponentOfNode(), getComponents(), getDegreeFor(), getDiameter(), getEdge(), getEdges(), getEdgesFor(), getMST(), getNeighborsFor(), getNetworkXObject(), getNodeNames(), getNodes(), getNumCliques(), getNumComponents(), getShortestPath(), getShortestPathLength(), getSize(), hasPath(), iterNodes(), setDefaultWeight()

Inherited from pylogeny.tree.treeSet(Section 17.4)

addTreeByNewick(), toTreeFile()

Inherited from _abcoll.Sized

___subclasshook___()

Inherited from object

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

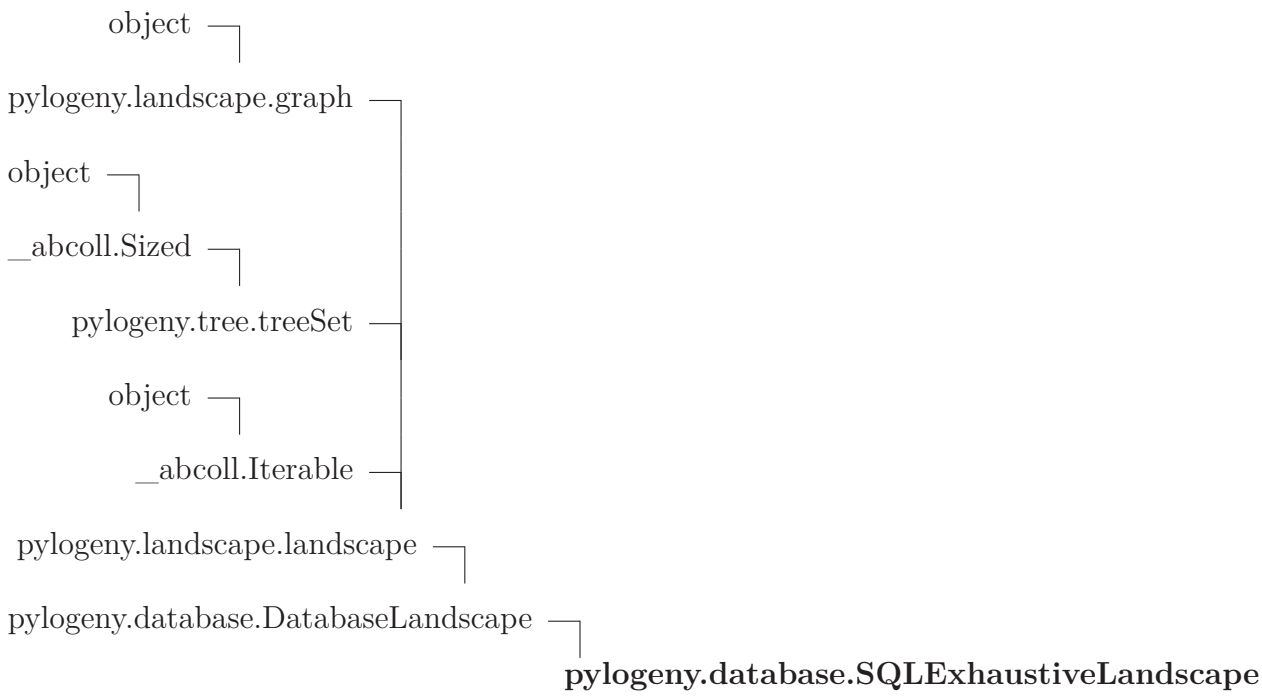
6.2.2 Properties

Name	Description
Inherited from object	
___class___	

6.2.3 Class Variables

Name	Description
___abstractmethods___	**Value:** `frozenset(['_fetchRearrangementsFromDatabase', '_fetchTre...`

6.3 Class SQLExhaustiveLandscape



6.3.1 Methods

___init___(*self*, *dbobj*, *aliname*)

Initialize the landscape.

:param ali: An :class:‘alignment.alignment‘ object. :param starting_tree: An optional tree object to start the landscape with. :param root: Whether or not to acquire an approximate maximum likelihood tree (FastTree) or start the landscape with a given starting tree. :param operator: A string that describes what operator the landscape is mostly comprised of.

Overrides: object.___init___ extit(inherited documentation)

exploreRandomTree(*self*, *i*)

Acquire a single neighbor to a tree in the landscape by performing a random rearrangement of type SPR (by default), NNI, or TBR. Rearrangement type is provided as a rearrangement module type definition of form, for example, TYPE_SPR, TYPE_NNI, etc.

Overrides: pylogeny.landscape.landscape.exploreRandomTree extit(inherited documentation)

getDatabaseNode(*self*, *i*)

exploreTree(*self*, *i*)

Get all neighbors to a tree named i in the landscape using a respective rearrangement operator as defined in the rearrangement module. Rearrangement type is provided as a rearrangement module type definition of form, for example, TYPE_SPR, TYPE_NNI, etc. By default, this is TYPE_SPR.

Overrides: pylogeny.landscape.landscape.exploreTree extit(inherited documentation)

Inherited from pylogeny.database.DatabaseLandscape(Section 6.2)

getNode()

Inherited from pylogeny.landscape.landscape(Section 9.3)

___getitem___(), ___iter___(), ___str___(), addTree(), findTree(), findTreeTopology(), findTreeTopologyByStructure(), getAlignment(), getAllPathsOfBestImprovement(), getBestImprovement(), getBipartitionFoundInTreeByIndex(), getGlobalOptimum(), getLocalOptima(), getLocks(), getNumberTaxa(), getPathOfBestImprovement(), getPossibleNumberRootedTrees(), getPossibleNumberUnrootedTrees(), getRoot(), getRootTree(), getTree(), getVertex(), indexOf(), isLocalOptimum(), isViolating(), iterAllPathsOfBestImprovement(), iterTrees(), lockBranchFoundInTree(), lockBranchFoundInTreeByIndex(), removeTree(), setAlignment(), toProperNewickTreeSet(), toTreeSet(), toggleLock()

Inherited from pylogeny.landscape.graph(Section 9.2)

___len___(), clearEdgeWeights(), getCenter(), getCliqueNumber(), getCliques(), getCliquesOfNode(), getComponentOfNode(), getComponents(), getDegreeFor(), getDiameter(), getEdge(), getEdges(), getEdgesFor(), getMST(), getNeighborsFor(), getNetworkXObject(), getNodeNames(), getNodes(), getNumCliques(), getNumComponents(), getShortestPath(), getShortestPathLength(), getSize(), hasPath(), iterNodes(), setDefaultWeight()

Inherited from pylogeny.tree.treeSet(Section 17.4)

addTreeByNewick(), toTreeFile()

Inherited from __abcoll.Sized

___subclasshook___()

Inherited from object

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

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

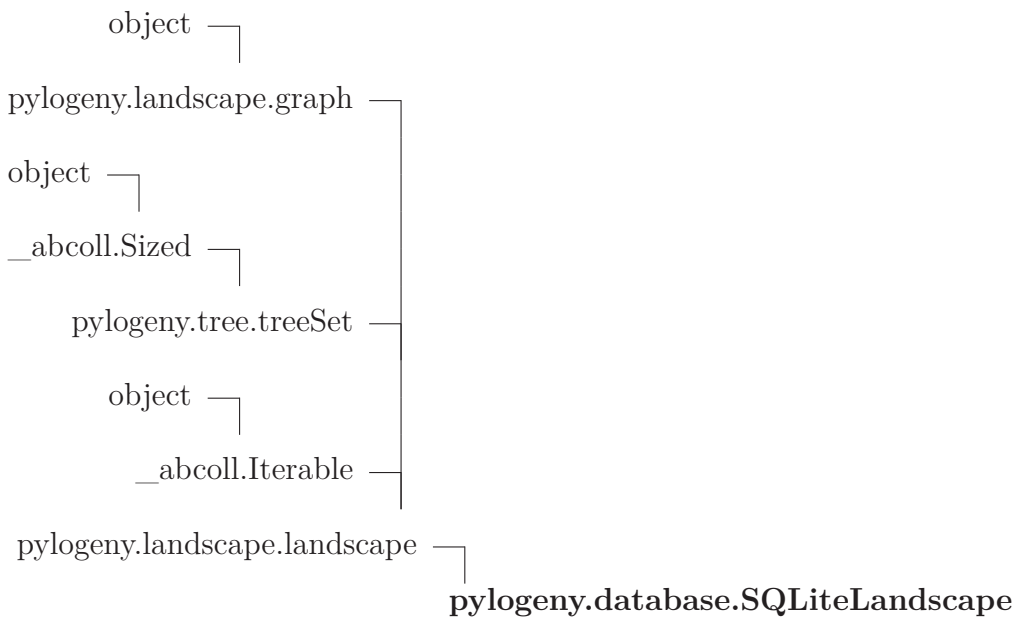
6.3.2 Properties

Name	Description
Inherited from object	
___class___	

6.3.3 Class Variables

Name	Description
___abstractmethods___	**Value:** `frozenset([])`

6.4 Class SQLiteLandscape



Allow random access of all landscape data from an sqlite file found on the hard disk.

6.4.1 Methods

___init___(*self*, *dbobj*)

Initialize the landscape.

:param ali: An :class:‘alignment.alignment‘ object. :param starting_tree: An optional tree object to start the landscape with. :param root: Whether or not to acquire an approximate maximum likelihood tree (FastTree) or start the landscape with a given starting tree. :param operator: A string that describes what operator the landscape is mostly comprised of.

Overrides: object.___init___ extit(inherited documentation)

Inherited from pylogeny.landscape.landscape(Section 9.3)

___getitem___(), ___iter___(), ___str___(), addTree(), exploreRandomTree(), exploreTree(), findTree(), findTreeTopology(), findTreeTopologyByStructure(), getAlignment(), getAllPathsOfBestImprovement(), getBestImprovement(), getBipartitionFoundInTreeByIndex(), getGlobalOptimum(), getLocalOptima(), getLocks(), getNumberTaxa(), getPathOfBestImprovement(), getPossibleNumberRootedTrees(), getPossibleNumberUnrootedTrees(), getRoot(), getRootTree(), getTree(), getVertex(), indexOf(), isLocalOptimum(), isViolating(), iterAllPathsOfBestImprovement(), iterTrees(), lockBranchFoundInTree(), lockBranchFoundInTreeByIndex(), removeTree(), setAlignment(), toProperNewickTreeSet(), toTreeSet(), toggleLock()

Inherited from pylogeny.landscape.graph(Section 9.2)

___len___(), clearEdgeWeights(), getCenter(), getCliqueNumber(), getCliques(), getCliquesOfNode(), getComponentOfNode(), getComponents(), getDegreeFor(), getDiameter(), getEdge(), getEdges(), getEdgesFor(), getMST(), getNeighborsFor(), getNetworkXObject(), getNode(), getNodeNames(), getNodes(), getNumCliques(), getNumComponents(), getShortestPath(), getShortestPathLength(), getSize(), hasPath(), iterNodes(), setDefaultWeight()

Inherited from pylogeny.tree.treeSet(Section 17.4)

addTreeByNewick(), toTreeFile()

Inherited from _abcoll.Sized

___subclasshook___()

Inherited from object

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

6.4.2 Properties

Name	Description
Inherited from object	
__class__	

6.4.3 Class Variables

Name	Description
Inherited from pylogeny.tree.treeSet (Section 17.4)	
__abstractmethods__	

6.5 Class database

object
└ **pylogeny.database.database**

Known Subclasses: pylogeny.database.SQLDatabase, pylogeny.database.SQLiteDatabase

Allow interfacing with a SQL/sqlite database.

6.5.1 Methods

getTables(*self*)

getColumns(*self*, *table*)

isEmpty(*self*)

Determine if the database is empty.

getHeaders(*self*, *table*)

Get only header names for a given table's columns.

getRecordsColumn(*self*, *table*, *col*)

Get all data for a single colmun from records for a table.

getRecords(*self*, *table*)

Get all records from a given table in the database.

iterRecords(*self*, *table*)

Get a record, one at a time, from a table in the database.

filterRecords(*self*, *table*, *condn*)

Get all records from a given table following a condition.

getRecordsAsDict(*self*, *table*)

Acquires records using getRecords() and then leverages access using a dictionary data structure.

newTable(*self*, *tablename*, **args*)

insertRecords(*self*, *tablename*, *items*)

insertRecord(*self*, *tablename*, *record*)

query(*self*, *q*)

querymany(*self*, *q*, *i*)

close(*self*)

Inherited from object

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

6.5.2 Properties

Name	Description
Inherited from object	
___class___	

6.5.3 Class Variables

Name	Description
cursor	**Value:** `None`

continued on next page

Name	Description
___abstractmethods___	**Value:** `frozenset(['close', 'getColumns', 'getTables', 'query', '...`

6.6 Class SQLDatabase

object
└
pylogeny.database.database
└
pylogeny.database.SQLDatabase

Database object to allow reading from a MySQL database.

6.6.1 Methods

___init___(*self*, *host*, *user*, *pw*, *db*)

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

Overrides: object.___init___ extit(inherited documentation)

connect(*self*)

getTables(*self*)

Overrides: pylogeny.database.database.getTables

getColumns(*self*, *table*)

Return column information for a given table.

Overrides: pylogeny.database.database.getColumns

query(*self*, *q*)

Overrides: pylogeny.database.database.query

querymany(*self*, *q*, *i*)

Overrides: pylogeny.database.database.querymany

close(*self*)

Overrides: pylogeny.database.database.close

Inherited from pylogeny.database.database(Section 6.5)

filterRecords(), getHeaders(), getRecords(), getRecordsAsDict(), getRecordsColumn(), insertRecord(), insertRecords(), isEmpty(), iterRecords(), newTable()

Inherited from object

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

6.6.2 Properties

Name	Description
Inherited from object	
___class___	

6.6.3 Class Variables

Name	Description
___abstractmethods___	**Value:** `frozenset([])`
Inherited from pylogeny.database.database (Section 6.5)	
cursor	

6.7 Class SQLiteDatabase

object

pylogeny.database.database

pylogeny.database.SQLiteDatabase

6.7.1 Methods

___init___(*self*, *filepath*)

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

Overrides: object.___init___ extit(inherited documentation)

getColumns(*self*, *table*)

Return column information for a given table.

Overrides: pylogeny.database.database.getColumns

getTables(*self*)

Overrides: pylogeny.database.database.getTables

query(*self*, *q*)

Overrides: pylogeny.database.database.query

querymany(*self*, *q*, *i*)

Overrides: pylogeny.database.database.querymany

close(*self*)

Overrides: pylogeny.database.database.close

Inherited from pylogeny.database.database(Section 6.5)

filterRecords(), getHeaders(), getRecords(), getRecordsAsDict(), getRecordsColumn(), insertRecord(), insertRecords(), isEmpty(), iterRecords(), newTable()

Inherited from object

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

6.7.2 Properties

Name	Description
Inherited from object	
___class___	

6.7.3 Class Variables

Name	Description
___abstractmethods___	**Value:** `frozenset([])`
Inherited from pylogeny.database.database (Section 6.5)	
cursor	

7 Module pylogeny.executable

Defines an interface to manage interfacing with the system for respective application calls and implements multiple of these for executables such as FastTree and RAxML. Requires a UNIX environment.

7.1 Functions

exeExists(*cmd*)

Determines whether a function exists in a UNIX environment.

7.2 Variables

Name	Description
E_FASTTREE	**Value:** `'fasttree'`
E_RAXML	**Value:** `'raxmlHPC'`
E_TREEPUZZ	**Value:** `'puzzle'`
__package__	**Value:** `'pylogeny'`

7.3 Class aTemporaryDirectory

object
└ **pylogeny.executable.aTemporaryDirectory**

A class intended to be used as a context manager that allows Python to run in a temporary directory for a finite period of time.

7.3.1 Methods

__init__(*self*, *dir*=`None`)

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

Overrides: object.__init__ extit(inherited documentation)

__enter__(*self*)

__exit__(*self*, **args*)

Inherited from object

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

7.3.2 Properties

Name	Description
Inherited from object	
___class___	

7.4 Class executable

object
└
pylogeny.executable.executable

Known Subclasses: pylogeny.executable.consel, pylogeny.executable.fasttree, pylogeny.executable.raxml, pylogeny.executable.treepuzzle

An interface for the instantation and running of a single instance for a given application.

7.4.1 Methods

getInstructionString(*self*)

run(*self*)

Perform a run of this application.

Inherited from object

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

7.4.2 Properties

Name	Description
Inherited from object	
___class___	

7.4.3 Class Variables

Name	Description
exeName	**Value:** `None`
___abstractmethods___	**Value:** `frozenset(['getInstructionString'])`

7.5 Class treepuzzle

object
└
pylogeny.executable.executable
└
pylogeny.executable.treepuzzle

Wrap TREE-PUZZLE in order to create an intermediate file for CONSEL to read and assign confidence to a set of trees. Requires TREE-PUZZLE to be installed.

7.5.1 Methods

___init___(*self*, *ali*, *treefile*)

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

Overrides: object.___init___ extit(inherited documentation)

getInstructionString(*self*)

Overrides: pylogeny.executable.executable.getInstructionString

getSiteLikelihoodFile(*self*)

Inherited from pylogeny.executable.executable(Section 7.4)

run()

Inherited from object

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

7.5.2 Properties

Name	Description
Inherited from object	
___class___	

7.5.3 Class Variables

Name	Description
exeName	**Value:** `'puzzle'`
___abstractmethods___	**Value:** `frozenset([])`

7.6 Class consel

object

pylogeny.executable.executable

pylogeny.executable.consel

Denotes a single run of the CONSEL workflow in order to acquire a confidence interval and perform an AU test on a set of trees. Requires CONSEL to be installed.

7.6.1 Methods

___init___(*self*, *treeset*, *alignment*, *name*)

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

Overrides: object.___init___ extit(inherited documentation)

getInstructionString(*self*)

Overrides: pylogeny.executable.executable.getInstructionString

getInterval(*self*)

Compute the AU test. Return the interval of trees.

Inherited from pylogeny.executable.executable(Section 7.4)

run()

Inherited from object

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

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

7.6.2 Properties

Name	Description
Inherited from object	
___class___	

7.6.3 Class Variables

Name	Description
___abstractmethods___	**Value:** `frozenset([])`
Inherited from pylogeny.executable.executable (Section 7.4)	
exeName	

7.7 Class fasttree

object

pylogeny.executable.executable

pylogeny.executable.fasttree

Denotes a single run of the FastTree executable in order to acquire an approximate maximum likelihood tree for the input alignment. See http://www.microbesonline.org/fasttree/ for more information on FastTree. Requires FastTree to be installed.

7.7.1 Methods

___init___(*self*, *inp_align*, *out_file*=`None`, *isProtein*=`True`)

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

Overrides: object.___init___ extit(inherited documentation)

getInstructionString(*self*)

Overrides: pylogeny.executable.executable.getInstructionString

Inherited from pylogeny.executable.executable(Section 7.4)

run()

Inherited from object

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

7.7.2 Properties

Name	Description
Inherited from object	
___class___	

7.7.3 Class Variables

Name	Description
exeName	**Value:** `'fasttree'`
___abstractmethods___	**Value:** `frozenset([])`

7.8 Class raxml

object

pylogeny.executable.executable

pylogeny.executable.raxml

Denotes a single run of the RAxML executable. See http://sco.h-its.org/exelixis/software.html for more information on RAxML. Requires RAxML to be installed.

7.8.1 Methods

___init___(*self*, *inp_align*, *out_file*, *model*=`None`, *is_Protein*=`True`, *interTrees*=`False`, *alg*=`None`, *startingTree*=`None`, *rapid*=`False`, *slow*=`False`, *optimizeBootstrap*=`False`, *numboot*=`100`, *log*=`None`, *wdir*=`None`)

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

Overrides: object.___init___ extit(inherited documentation)

getInstructionString(*self*)

Overrides: pylogeny.executable.executable.getInstructionString

runFunction(*self*, *alg*)

Inherited from pylogeny.executable.executable(Section 7.4)

run()

Inherited from object

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

7.8.2 Properties

Name	Description
Inherited from object ___class___	

7.8.3 Class Variables

Name	Description
exeName	**Value:** `'raxmlHPC'`
___abstractmethods___	**Value:** `frozenset([])`

8 Module pylogeny.heuristic

Define the interface for a heuristic in order to implement any manner of heuristic for a combinatorial problem that can be abstracted into a state graph. In this case, a phylogenetic tree space.

8.1 Variables

Name	Description
SMGR_MIN_NUM_LIKELIHOOD	**Value:** `32`
__package__	**Value:** `'pylogeny'`

8.2 Class heuristic

object
└
pylogeny.heuristic.heuristic

Known Subclasses: pylogeny.heuristic.phylogeneticLinearHeuristic

A base interface for a heuristic that explores a state graph.

8.2.1 Methods

__init__(*self*, *G*=`None`, *start*=`None`)

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

Overrides: object.__init__ extit(inherited documentation)

explore(*self*)

getStateGraph(*self*)

getStartState(*self*)

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

8.2.2 Properties

Name	Description
Inherited from object	
___class___	

8.3 Class phylogeneticLinearHeuristic

object
└
pylogeny.heuristic.heuristic
└
pylogeny.heuristic.phylogeneticLinearHeuristic

Known Subclasses: pylogeny.heuristic.RAxMLIdentify, pylogeny.heuristic.likelihoodGreedy, pylogeny.heuristic.parsimonyGreedy, pylogeny.heuristic.smoothGreedy

A base class for a heuristic that works on a phylogenetic landscape and only possesses a single path (of search).

8.3.1 Methods

___init___(*self*, *ls*, *startNode*)

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

Overrides: object.___init___ extit(inherited documentation)

getPath(*self*)

getBestTree(*self*)

Inherited from pylogeny.heuristic.heuristic(Section 8.2)

explore(), getStartState(), getStateGraph()

Inherited from object

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

8.3.2 Properties

Name	Description
Inherited from object	
___class___	

8.3.3 Class Variables

Name	Description
bestTree	**Value:** `None`
path	**Value:** `[]`

8.4 Class parsimonyGreedy

object
pylogeny.heuristic.heuristic
pylogeny.heuristic.phylogeneticLinearHeuristic
pylogeny.heuristic.parsimonyGreedy

Greedy (hill-climbing) landscape exploration by comparsion of parsimony.

8.4.1 Methods

___init___(*self*, *ls*, *startNode*)

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

Overrides: object.___init___ extit(inherited documentation)

explore(*self*)

Perform greedy search of the landscape using a method of greed via parsimonious criterion.

Overrides: pylogeny.heuristic.heuristic.explore

Inherited from pylogeny.heuristic.phylogeneticLinearHeuristic(Section 8.3)

getBestTree(), getPath()

Inherited from pylogeny.heuristic.heuristic(Section 8.2)

getStartState(), getStateGraph()

Inherited from object

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

8.4.2 Properties

Name	Description
Inherited from object ___class___	

8.4.3 Class Variables

Name	Description
Inherited from pylogeny.heuristic.phylogeneticLinearHeuristic (Section 8.3) bestTree, path	

8.5 Class likelihoodGreedy

object
└
pylogeny.heuristic.heuristic
└
pylogeny.heuristic.phylogeneticLinearHeuristic
└
pylogeny.heuristic.likelihoodGreedy

Greedy (hill-climbing) landscape exploration by comparsion of likelihood.

8.5.1 Methods

___init___(*self*, *ls*, *startNode*)

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

Overrides: object.___init___ extit(inherited documentation)

explore(*self*)

Perform greedy search of the landscape using a method of greed via likelihood.

Overrides: pylogeny.heuristic.heuristic.explore

Inherited from pylogeny.heuristic.phylogeneticLinearHeuristic(Section 8.3)

getBestTree(), getPath()

Inherited from pylogeny.heuristic.heuristic(Section 8.2)

getStartState(), getStateGraph()

Inherited from object

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

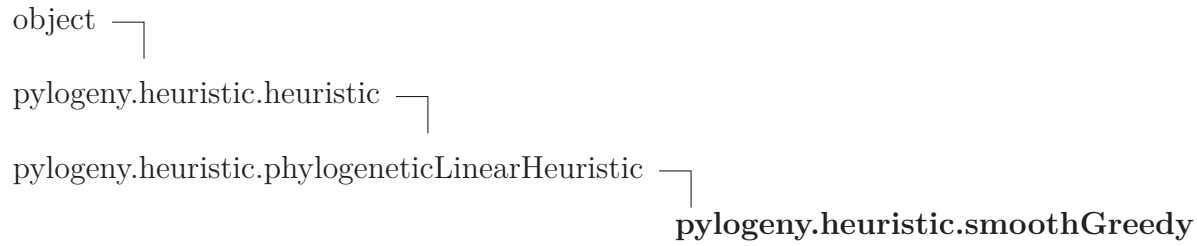
8.5.2 Properties

Name	Description
Inherited from object	
___class___	

8.5.3 Class Variables

Name	Description
Inherited from pylogeny.heuristic.phylogeneticLinearHeuristic (Section 8.3)	
bestTree, path	

8.6 Class smoothGreedy



Parsimony-driven greedy landscape exploration by comparsion of likelihoods.

8.6.1 Methods

___init___(*self*, *ls*, *startNode*)

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

Overrides: object.___init___ extit(inherited documentation)

explore(*self*)

Perform greedy search of the landscape using a method of greed via parsimonious criterion and then performing final smoothing via likelihood on top 10% of 1-SPR neighbors ranked on basis of parsimony.

Overrides: pylogeny.heuristic.heuristic.explore

Inherited from pylogeny.heuristic.phylogeneticLinearHeuristic(Section 8.3)

getBestTree(), getPath()

Inherited from pylogeny.heuristic.heuristic(Section 8.2)

getStartState(), getStateGraph()

Inherited from object

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

8.6.2 Properties

Name	Description
Inherited from object	
___class___	

8.6.3 Class Variables

Name	Description
Inherited from pylogeny.heuristic.phylogeneticLinearHeuristic (Section 8.3)	
bestTree, path	

8.7 Class RAxMLIdentify

object
pylogeny.heuristic.heuristic
pylogeny.heuristic.phylogeneticLinearHeuristic
pylogeny.heuristic.RAxMLIdentify

RAxML-driven landscape evaluation of intermediate checkpoint trees output from the RAxML executable.

8.7.1 Methods

___init___(*self*, *ls*, *startNode*, *workdir='.rxml'*)

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

Overrides: object.___init___ extit(inherited documentation)

___setupWorkDir___(*self*)

___setupExecutable___(*self*)

___readLogFile___(*self*)

___readIterTrees___(*self*, *iters*)

explore(*self*)

Overrides: pylogeny.heuristic.heuristic.explore

Inherited from pylogeny.heuristic.phylogeneticLinearHeuristic(Section 8.3)

getBestTree(), getPath()

Inherited from pylogeny.heuristic.heuristic(Section 8.2)

getStartState(), getStateGraph()

Inherited from object

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

8.7.2 Properties

Name	Description
Inherited from object ___class___	

8.7.3 Class Variables

Name	Description
Inherited from pylogeny.heuristic.phylogeneticLinearHeuristic (Section 8.3) bestTree, path	

9 Module pylogeny.landscape

Encapsulate a phylogenetic tree space. A phylogenetic landscape or tree space refers to the entire combinatorial space comprising all possible phylogenetic tree topologies for a set of n taxa. The landscape of n taxa can be defined as consisting of a finite set T of tree topologies. Tree topologies can be associated with a fitness function *f(t_i)* describing their fit. This forms a discrete solution search space and finite graph *(T, E) = G*. *E(G)* refers to the neighborhood relation on *T(G)*. Edges in this graph are bidirectional and represent transformation from one tree topology to another by a tree rearrangement operator. An edge between *t_i* and *t_j* would be notated as *e_{ij}* in *E(G)*.

9.1 Variables

Name	Description
LS_NOT_DEFINED	**Value:** `-1`
__package__	**Value:** `'pylogeny'`

9.2 Class graph

object
└
pylogeny.landscape.graph

Known Subclasses: pylogeny.landscape.landscape

Define an empty graph object.

9.2.1 Methods

__init__(*self*, *gr*=`None`)

Instantiate a graph.

:param gr: A networkx graph object, if already exists.

Overrides: object.__init__

getNetworkXObject(*self*)

Return the internal networkx graph object.

__len__(*self*)

___iter___(*self*)

getSize(*self*)

Return the number of nodes in the graph.

getNodeNames(*self*)

Return the names of nodes in the graph.

iterNodes(*self*)

Iterate over all node keys.

getNodes(*self*)

getEdges(*self*)

getEdgesFor(*self*, *i*)

getNode(*self*, *i*)

getEdge(*self*, *i*, *j*)

getNeighborsFor(*self*, *i*)

getDegreeFor(*self*, *i*)

Return in- and out-degree for node named i.

setDefaultWeight(*self*, *w*)

clearEdgeWeights(*self*)

getNumComponents(*self*)

Get the number of components of the graph.

getComponents(*self*)

Get the connected components in the graph.

getComponentOfNode(*self*, *i*)

Get the graph component of a given node.

getCliques(*self*)

Get the cliques present in the graph.

getCliqueNumber(*self*)

Get the clique number of the graph.

getNumCliques(*self*)

Get the number of cliques found in the graph.

getCliquesOfNode(*self*, *i*)

Get the clique that a node corresponds to.

getCenter(*self*)

Get the centre of the graph.

getDiameter(*self*)

Acquire the diameter of the graph.

getMST(*self*)

Acquire the minimum spanning tree for the graph.

hasPath(*self*, *nodA*, *nodB*)

See if a path exists between two nodes.

getShortestPath(*self*, *nodA*, *nodB*)

Get the shortest path between two nodes.

getShortestPathLength(*self*, *nodA*, *nodB*)

Get the shortest path length between two nodes.

Inherited from object

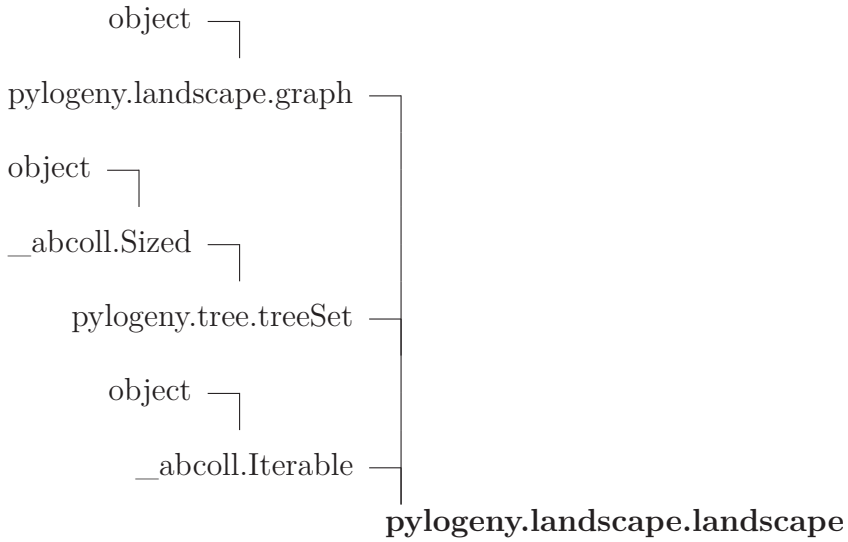
___delattr___(), ___format___(), ___getattribute___(), ___hash___(), ___new___(), ___reduce___(), ___reduce_ex___(), ___repr___(), ___setattr___(), ___sizeof___(),

___str___(), ___subclasshook___()

9.2.2 Properties

Name	Description
Inherited from object	
___class___	

9.3 Class landscape



Known Subclasses: pylogeny.database.DatabaseLandscape, pylogeny.database.SQLiteLandscape

Defines an entire phylogenetic tree space.

9.3.1 Methods

___init___(*self*, *ali*, *starting_tree*=`None`, *root*=`True`, *operator*=`'SPR'`)

Initialize the landscape.

:param ali: An :class:‘alignment.alignment‘ object. :param starting_tree: An optional tree object to start the landscape with. :param root: Whether or not to acquire an approximate maximum likelihood tree (FastTree) or start the landscape with a given starting tree. :param operator: A string that describes what operator the landscape is mostly comprised of.

Overrides: object.___init___

getAlignment(*self*)

Acquire the alignment object associated with this space.

getNumberTaxa(*self*)

Return the number of different taxa present in any respective tree in the landscape.

getPossibleNumberRootedTrees(*self*)

Assuming all of the trees in the space are rooted, return the maximum possible number of unrooted trees that can possibly be generated for the number of taxa of trees in the landscape.

getPossibleNumberUnrootedTrees(*self*)

Assuming all of the trees in the space are unrooted, return the maximum possible number of unrooted trees that can possibly be generated for the number of taxa of trees in the landscape.

getRoot(*self*)

Returns the index to the root (starting) tree of the space.

getRootTree(*self*)

Acquire the first tree that was placed in this space.

setAlignment(*self*, *ali*)

Set the alignment present in this landscape. WARNING; will not modify existing scores.

getTree(*self*, *i*)

Get the tree object for a tree by its ID or name i.

iterTrees(*self*)

Iterate over all trees found in this landscape.

___iter___(*self*)

Overrides: _abcoll.Iterable.___iter___

getVertex(*self*, *i*)

Acquire a vertex object from the landscape; this is a high-level representation of a tree in the landscape with additional functionality. Object created upon invocation of this function.

removeTree(*self*, *tree*)

Remove a tree from the landscape by object.

Overrides: pylogeny.tree.treeSet.removeTree

addTree(*self*, *tree*)

Add a tree to the landscape. Will return its index.

Overrides: pylogeny.tree.treeSet.addTree

exploreRandomTree(*self*, *i*, *type*=`1`)

Acquire a single neighbor to a tree in the landscape by performing a random rearrangement of type SPR (by default), NNI, or TBR. Rearrangement type is provided as a rearrangement module type definition of form, for example, TYPE_SPR, TYPE_NNI, etc.

exploreTree(*self*, *i*, *type*=`1`)

Get all neighbors to a tree named i in the landscape using a respective rearrangement operator as defined in the rearrangement module. Rearrangement type is provided as a rearrangement module type definition of form, for example, TYPE_SPR, TYPE_NNI, etc. By default, this is TYPE_SPR.

getLocks(*self*)

toggleLock(*self*, *lock*)

Add a biparition to the list of locked bipartitions if not present; otherwise, remove it. Return status of lock.

lockBranchFoundInTree(*self*, *tr*, *br*)

Given a tree node and a branch object, add a given bipartition to the bipartition lock list. Returns true if locked.

getBipartitionFoundInTreeByIndex(*self*, *tr*, *brind*, *topol*=`None`)

Given a tree node and a branch index, return the associated bipartition.

lockBranchFoundInTreeByIndex(*self*, *tr*, *brind*)

Given a tree node and a branch index, add a given bipartition to the bipartition lock list. Returns true if locked.

isViolating(*self*, *i*)

Determine if a tree is violating any locks intrinsic to the landscape.

___getitem___(*self*, *i*)

Overrides: pylogeny.tree.treeSet.___getitem___

indexOf(*self*, *tr*)

Acquire the index/name in this landscape of a tree object. Returns -1 if not found.

Overrides: pylogeny.tree.treeSet.indexOf

findTree(*self*, *newick*)

Find a tree by Newick string, taking into account branch lengths. Returns the name of this tree in the landscape.

findTreeTopology(*self*, *newick*)

Find a tree by topology, not taking into account branch lengths.

findTreeTopologyByStructure(*self*, *struct*)

Find a tree by topology, not taking into account branch lengths, given the topology.

getBestImprovement(*self*, *i*)

For a tree in the landscape, investigate neighbors to find a tree that leads to the best improvement of fitness function score on the basis of likelihood.

getPathOfBestImprovement(*self*, *i*)

For a tree in the landscape, investigate neighbors iteratively until a best path of score improvement is found on basis of likelihood.

getAllPathsOfBestImprovement(*self*)

Return all paths of best improvement as a dictionary.

iterAllPathsOfBestImprovement(*self*)

Return an iterator for all paths of best improvement.

isLocalOptimum(*self*, *i*)

Determine if a tree is, without any doubt, a local optimum.

getLocalOptima(*self*)

Get all trees in the landscape that can be labelled as a local optimum.

getGlobalOptimum(*self*)

Get the global optimum of the current space.

___str___(*self*)

str(x)

Overrides: object.___str___ extit(inherited documentation)

toProperNewickTreeSet(*self*)

Convert this landscape into an unorganized set of trees where taxa names are transformed to their original form (i.e. not transformed to a state friendly for the Phylip format).

toTreeSet(*self*)

Convert this landscape into an unorganized set of trees.

Inherited from pylogeny.landscape.graph(Section 9.2)

___len___(), clearEdgeWeights(), getCenter(), getCliqueNumber(), getCliques(), getCliquesOfNode(), getComponentOfNode(), getComponents(), getDegreeFor(), getDiameter(), getEdge(), getEdges(), getEdgesFor(), getMST(), getNeighborsFor(), getNetworkXObject(), getNode(), getNodeNames(), getNodes(), getNumCliques(), getNumComponents(), getShortestPath(), getShortestPathLength(), getSize(), hasPath(), iterNodes(), setDefaultWeight()

Inherited from pylogeny.tree.treeSet(Section 17.4)

addTreeByNewick(), toTreeFile()

Inherited from _abcoll.Sized

___subclasshook___()

Inherited from object

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

9.3.2 Properties

Name	Description
Inherited from object	
___class___	

9.3.3 Class Variables

Name	Description
Inherited from pylogeny.tree.treeSet (Section 17.4)	
___abstractmethods___	

9.4 Class vertex

object
└
pylogeny.landscape.vertex

Encapsulate a single vertex in the landscape and add convenient functionality to alias parent landscape functions.

9.4.1 Methods

___init___(*self*, *obj*, *ls*)

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

Overrides: object.___init___ extit(inherited documentation)

getIndex(*self*)

getDict(*self*)

getObject(*self*)

getTree(*self*)

getNewick(*self*)

getScore(*self*)

getOrigin(*self*)

getNeighbors(*self*)

getDegree(*self*)

isLocalOptimum(*self*)

isExplored(*self*)

isFailed(*self*)

setExplored(*self*, *exp*)

Sets the "explored" flag of this node in the landscape.

approximatePossibleNumNeighbors(*self*)

Approximate the possible number of neighbors to this vertex by considering the type of tree rearrangement operator.

scoreLikelihood(*self*)

Acquire the log-likelihood for this vertex.

getBestImprovement(*self*)

Alias function for function of same name in parent landscape.

getPathOfBestImprovement(*self*)

Alias function for function of same name in parent landscape.

isBestImprovement(*self*)

Check to see if this vertex is a best move for another node.

isViolating(*self*)

Alias function for function of same name in parent landscape.

getProperNewick(*self*)

Get the proper Newick string for a tree. :returns: A string.

iterBipartitions(*self*)

Return a generator to iterate over all bipartitions for this vertex.

getBipartitions(*self*)

Get all bipartitions for this vertex.

getBipartitionScores(*self*)

Get all corresponding bipartition vectors of SPR scores.

getNeighborsOfBipartition(*self*, *bi*)

Get corresponding neighbors of a bipartition in this vertex's tree.

getNeighborsOfBranch(*self*, *br*)

Get corresponding neighbors of a branch in this vertex's tree.

Inherited from object

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

9.4.2 Properties

Name	Description
Inherited from object	
___class___	

10 Module pylogeny.landscapeWriter

Serialize a phylogenetic landscape into an SQLlite database file made up of three components: all tree IDs and respective scores, the alignment file as a set of sequences, and a representation of the graph as an edge list.

10.1 Variables

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

10.2 Class landscapeWriter

object
└
pylogeny.landscapeWriter.landscapeWriter

Known Subclasses: pylogeny.JSONWriter.JSONWriter

Encapsulate the writing of a landscape to a file format.

10.2.1 Methods

___init___(*self*, *landscape*, *name*)

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

Overrides: object.___init___ extit(inherited documentation)

writeFile(*self*, *path='.'*)

Write the landscape serialized file to given path.

Inherited from object

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

10.2.2 Properties

Name	Description
Inherited from object	
___class___	

10.3 Class landscapeParser

object
└
pylogeny.landscapeWriter.landscapeParser

Encapsulates the construction of a landscape object from a sqlite landscape file.

10.3.1 Methods

___init___(*self*, *path*)

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

Overrides: object.___init___ extit(inherited documentation)

getName(*self*)

Acquire the name of the parsed landscape.

parse(*self*)

Parse the file.

Inherited from object

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

10.3.2 Properties

Name	Description
Inherited from object	
___class___	

11 Module pylogeny.model

Phylogenetic tree scoring models; intended to be coupled with the use of pytbeaglehon (BEAGLE) high-performance library.

11.1 Variables

Name	Description
pytbeaglehonEnabled	**Value:** `True`
___package___	**Value:** `'pylogeny'`

11.2 Class PhyloModelError

object
└ exceptions.BaseException
└ exceptions.Exception
└ **pylogeny.model.PhyloModelError**

11.2.1 Methods

___init___(*self*, *v*)

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

Overrides: object.___init___ extit(inherited documentation)

___str___(*self*)

str(x)

Overrides: object.___str___ extit(inherited documentation)

Inherited from exceptions.Exception

___new___()

Inherited from exceptions.BaseException

___delattr___(), ___getattribute___(), ___getitem___(), ___getslice___(), ___reduce___(), ___repr___(), ___setattr___(), ___setstate___(), ___unicode___()

Inherited from object

___format___(), ___hash___(), ___reduce_ex___(), ___sizeof___(), ___subclasshook___()

11.2.2 Properties

Name	Description
Inherited from exceptions.BaseException	
args, message	
Inherited from object	
___class___	

11.3 Class DiscreteStateModel

object
└ **pylogeny.model.DiscreteStateModel**

Initialize a discrete state model for phylogenetic data. State frequencies and character time are determined from the given alignment object.

11.3.1 Methods

___init___(*self*, *alignment*)

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

Overrides: object.___init___ extit(inherited documentation)

getAlignment(*self*)

getAlignmentAsStateList(*self*)

getSequenceMatrix(*self*)

getCharType(*self*)

getStateFreqs(*self*)

getRawStateFreqs(*self*)

getRawStateFreqsAsList(*self*)

getRawStateFreqsAsDict(*self*)

getFrequencyOfState(*self*, *i*)

getRawFrequencyOfState(*self*, *i*)

Inherited from object

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

11.3.2 Properties

Name	Description
Inherited from object	
___class___	

12 Module pylogeny.newick

Newick string parsing and object interaction. A Newick string can represent a phylogenetic tree.

12.1 Functions

assignParents(*top*)

Should be a one-time use function. Goes through and assigns parents to the parsed newick tree structure nodes and branches to allow for up-traversal.

removeBranchLengths(*top*)

Goes through and removes any stored branch lengths.

removeUnaryInternalNodes(*top*)

Goes through and ensures any degree-2 internal nodes are smoothed into a single degree-3 internal node.

invertAlongPathToNode(*target*, *top*)

DANGEROUS: Reverses all directionality to a given node from a top-level node. Intended as a low-level function for rerooting a tree.

shuffleLeaves(*top*)

DANGEROUS: Given a top-level node, shuffle all leaves in this tree.

getAllBranches(*br*)

Given a branch, traverse subtree and return comprising branches as a list.

isSibling(*br*, *other*)

Given a branch, determine if that branch is adjacent to another branch.

getBalancingBracket(*newick*, *i*)

Given a position of an opening bracket in a newick string, i, output the closing bracket's position that corresponds to this opening bracket.

getBranchLength(*newick*, *i*)

Given a position of a colon symbol (indicating a branch length), return the branch length.

getLeafName(*newick*, *i*)

Given the position of a leaf, find its complete name.

parseNewick(*newick*, *i*, *j*, *top*)

Parse a newick string into a topological newick structure given a top-level node.

12.2 Variables

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

12.3 Class ParsingError

object
└ exceptions.BaseException
└ exceptions.Exception
└ **pylogeny.newick.ParsingError**

12.3.1 Methods

__init__(*self*, *val*)

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

Overrides: object.__init__ extit(inherited documentation)

__str__(*self*)

str(x)

Overrides: object.__str__ extit(inherited documentation)

Inherited from exceptions.Exception

___new___()

Inherited from exceptions.BaseException

___delattr___(), ___getattribute___(), ___getitem___(), ___getslice___(), ___reduce___(), ___repr___(), ___setattr___(), ___setstate___(), ___unicode___()

Inherited from object

___format___(), ___hash___(), ___reduce_ex___(), ___sizeof___(), ___subclasshook___()

12.3.2 Properties

Name	Description
Inherited from exceptions.BaseException args, message	
Inherited from object ___class___	

12.4 Class node

object
pylogeny.base.treeNode
pylogeny.newick.node

Newick node.

12.4.1 Methods

___init___(*self*, *lbl*='', *children*=None, *parent*=None)

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

Overrides: object.___init___ extit(inherited documentation)

___str___(*self*)

str(x)

Overrides: object.___str___ extit(inherited documentation)

Inherited from pylogeny.base.treeNode(Section 5.3)

addChild(), getChildByIndex(), getChildren(), getLabel(), getParent(), isInternalNode(), isLeaf()

Inherited from object

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

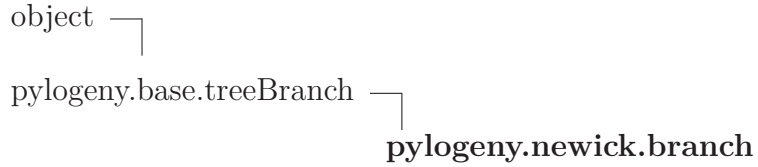
12.4.2 Properties

Name	Description
Inherited from object	
___class___	

12.4.3 Class Variables

Name	Description
Inherited from pylogeny.base.treeNode (Section 5.3)	
children, label, parent	

12.5 Class branch



Newick branch.

12.5.1 Methods

___init___(*self*, *chi*, *l*, *parent*=`None`, *s*=`None`)

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

Overrides: object.___init___ extit(inherited documentation)

___str___(*self*)

str(x)

Overrides: object.___str___ extit(inherited documentation)

Inherited from pylogeny.base.treeBranch(Section 5.4)

getChild(), getLabel(), getParent()

Inherited from object

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

12.5.2 Properties

Name	Description
Inherited from object	
___class___	

12.5.3 Class Variables

Name	Description
Inherited from pylogeny.base.treeBranch (Section 5.4)	
child, label, parent	

12.6 Class newickParser

Parsing object for Newick strings.

12.6.1 Methods

___init___(*self*, *newick*)

parse(*self*)

Parse the stored newick string into a topological structure.

___str___(*self*)

13 Module pylogeny.parsimony

Toolkit for performance of Parsimonious Criterion (Parsimony) methods of optimization of a phylogenetic topology with a particular set of data.

13.1 Functions

fitch_cost(*topology*, *profiles*)

Calculate the cost using Fitch algorithm on profile set and alignment. Deprecated: Python implementation of the Fitch algorithm; see fitch C++ module for a C++ implementation that is roughly four times faster.

fitch(*topology*, *alignment*)

Perform the Fitch algorithm on a given tree topology and associated alignment. Deprecated: Python implementation of the Fitch algorithm; see fitch C++ module for a C++ implementation that is roughly four times faster.

13.2 Variables

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

13.3 Class profile_set

Hold a set of site_profile profiles for an entire alignment.

13.3.1 Methods

___init___(*self*, *alignment*)

___len___(*self*)

weight(*self*, *val*)

get(*self*, *val*)

getForTaxa(*self*, *val*, *tax*)

13.4 Class site_profile

Consolidate the single-column alignment at a region into a set of components on the basis of similarity alone.

13.4.1 Methods

__init__(*self*, *alignment*, *site*)

__eq__(*self*, *o*)

__ne__(*self*, *o*)

__str__(*self*)

14 Module pylogeny.pll

Wrap C extension for libpll library for use in natural Python.

14.1 Variables

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

14.2 Class dataModel

Encapsulating a phylogenetic tree (as topology) + corresponding alignment into a libpll-associated data structure. Allows for log-likelihood scoring of this model. **MUST BE CLOSED AFTER USE.**

14.2.1 Methods

___init___(*self*, *topo*, *alignm*, *model*=`None`)

Initialize the data model and respective structures.

:param topo: A topology object. :type topo: :class: ‘rearrangement.topology‘ :param alignm: A phylipFriendlyAlignment object. :type alignm: :class: ‘alignment.phylipFriendlyAlignment‘

getLogLikelihood(*self*)

Calculates log-likelihood using libpll.

close(*self*)

If done with this particular problem. Frees associated memory.

14.3 Class partitionModel

A partition model intended for libpll.

14.3.1 Methods

___init___(*self*, *ali*)

getFileName(*self*)

Get the file name of the model file.

createSimpleModel(*self*, *protein*)

Establish a simple model (e.g., one type).

createModel(*self*, *models*, *partnames*, *ranges*)

Establish a more complex model.

close(*self*)

Delete file.

15 Module pylogeny.rearrangement

Phylogenetic tree structure encapsulation; allow rearrangement of said structure. Tree rearrangements inducing other topologies include Nearest Neighbor Interchange (NNI), Subtree Pruning and Regrafting (SPR), and Tree Bisection and Reconstruction (TBR). Each of these describe a transfer of one node in phylogenetic trees from one parent of a tree to a new parent. Respectively, these operators describe transformations that are subsets of those possible by the successive operator. For example, an NNI operator can perform transformations that are a subset of the transformations possible by the SPR operator.

15.1 Functions

dup(*topo*, *where*=`None`)

15.2 Variables

Name	Description
TYPE_NNI	**Value:** `2`
TYPE_SPR	**Value:** `1`
TYPE_TBR	**Value:** `3`
__package__	**Value:** `'pylogeny'`

15.3 Class RearrangementError

object
exceptions.BaseException
exceptions.Exception
pylogeny.rearrangement.RearrangementError

15.3.1 Methods

__init__(*self*, *val*)

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

Overrides: object.__init__ extit(inherited documentation)

___str___(*self*)

str(x)

Overrides: object.___str___ extit(inherited documentation)

Inherited from exceptions.Exception

___new___()

Inherited from exceptions.BaseException

___delattr___(), ___getattribute___(), ___getitem___(), ___getslice___(), ___reduce___(), ___repr___(), ___setattr___(), ___setstate___(), ___unicode___()

Inherited from object

___format___(), ___hash___(), ___reduce_ex___(), ___sizeof___(), ___subclasshook___()

15.3.2 Properties

Name	Description
Inherited from exceptions.BaseException args, message	
Inherited from object ___class___	

15.4 Class rearrangement

Encapsulates a single rearrangement move of type SPR, NNI, ...

15.4.1 Methods

___init___(*self*, *struct*, *type*, *targ*, *dest*)

Initialize by providing a pointer to a base topology, a target branch to be moved, and its destination.

getType(*self*)

Get the type of movement.

isNNI(*self*)

isSPR(*self*)

isTBR(*self*)

toTopology(*self*)

Commit the actual move and return the topology.

toNewick(*self*)

Commit the move but do not create a new structure. Only retrieve resultant Newick string; will be more efficient.

toTree(*self*)

Commit the move and transform to tree object.

doMove(*self*)

___str___(*self*)

15.5 Class topology

object
└
_abcoll.Container
└
pylogeny.base.treeStructure
└
pylogeny.rearrangement.topology

Encapsulate a tree topology, wrapping the newick tree structure. Is immutable.

15.5.1 Methods

___init___(*self*, *t*=`None`, *rerootToLeaf*=`True`, *toLeaf*=`None`)

Initialize structure with a top-level internal node OR nothing.

Overrides: object.___init___

rerootToLeaf(*self*, *toleaf*=`None`)

PRIVATE: Reroots the given tree structure such that it is rooted nearest the lowest-order leaf.

getBranches(*self*)

getLeaves(*self*)

getBipartitions(*self*)

Get all bipartitions.

getStrBipartitionFromBranch(*self*, *br*)

Given a branch, return corresponding bipartition.

getBranchFromStrBipartition(*self*, *bip*)

Given a bipartition of taxa, return a branch that creates that partition of tree taxa.

getBranchFromBipartition(*self*, *bip*)

Given a bipartition object, return a branch that creates that partition of taxa.

lockBranch(*self*, *branch*)

Given a branch, lock it such that no transitions can ever occur across it.

move(*self*, *branch*, *destination*, *returnStruct*=`True`)

Move a branch and attach to a destination branch. Return new structure, or return merely the resultant Newick string.

SPR(*self*, *branch*, *destination*)

Perform an SPR move of a branch to a destination branch, creating a new node there. Returns a rearrangement structure (not the actual new structure) that can then be polled for the actual move; this is in order to save memory.

NNI(*self*, *branch*, *destination*)

Perform an NNI move of a branch to a destination, only if that destination branch is a parent's parent or a parent's sibling. Returns a rearrangement structure (not the actual new structure) that can then be polled for the actual move; this is in order to save memory.

iterSPRForBranch(*self*, *br*, *flip*=`True`)

Consider all valid SPR moves for a given branch in the topology and yield all possible rearrangements as a generator.

allSPRForBranch(*self*, *br*, *flip*=`True`)

Consider all valid SPR moves for a given branch in the topology and return all possible rearrangements.

allSPR(*self*)

Consider all valid SPR moves for a given topology and return all possible rearrangements.

iterNNIForBranch(*self*, *br*, *flip*=`True`)

Consider all valid NNI moves for a given branch in the topology and and yield all possible rearrangements as a generator.

allNNIForBranch(*self*, *br*, *flip*=`True`)

Consider all valid NNI moves for a given branch in the topology and return all possible rearrangements.

allNNI(*self*)

Consider all valid NNI moves for a given topology and return all possible rearrangements.

allType(*self*, *type*=`1`)

Consider all valid moves of a given rearrangement operator for a given topology. Uses a given rearrangement operator type defined in this module. For example, calling this function by providing TYPE_NNI as the type will iterate over all NNI operations. By default, the type is TYPE_SPR.

iterTypeForBranch(*self*, *br*, *type*=`1`, *flip*=`True`)

Iterate over all possible rearrangements for a branch using a given rearrangement operator type defined in this module. For example, calling this function by providing TYPE_NNI as the type will iterate over all NNI operations. By default, the type is TYPE_SPR.

fromNewick(*self*, *newickstr*)

Alias for parse().

parse(*self*, *newickstr*)

Parse a newick string and assign the tree to this object. Cannot already be initialized with a tree.

toNewick(*self*)

Return the newick string of the tree.

toUnrootedNewick(*self*)

Return the newick string of the tree as an unrooted topology with a multifurcating top-level node.

toTree(*self*)

Return the tree object for this topology.

toUnrootedTree(*self*)

Return the tree object of the unrooted version of this topology.

___str___(*self*)

Return the newick string of the tree.

Overrides: object.___str___

Inherited from pylogeny.base.treeStructure(Section 5.2)

___contains___(), getAllLeaves(), getAllNodes(), getPostOrderTraversal(), getRoot(), leaves(), nodes(), postOrderTraversal()

Inherited from _abcoll.Container

___subclasshook___()

Inherited from object

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

15.5.2 Properties

Name	Description
Inherited from object	
___class___	

15.5.3 Class Variables

Name	Description
Inherited from pylogeny.base.treeStructure (Section 5.2)	
___abstractmethods___, root	

16 Module pylogeny.scoring

Functions for phylogenetic tree goodness-of-fit scoring.

16.1 Functions

beaglegetLogLikelihood(*tree*, *alignment*)

Acquire log-likelihood via C++ library BEAGLE via use of pybeaglethon wrapper library. Currently uses HKY85 model.

:param tree: A tree object. :type tree: :class: ‘tree.tree‘ :param alignment: An alignment object. :type alignment: :class: ‘alignment.alignment‘ :returns: A floating point value.

getLogLikelihoodForTopology(*topo*, *alignment*)

Acquire log-likelihood via C library libpll.

:param topo: A topology object. :type topo: :class: ‘rearrangement.topology‘ :param alignment: An alignment object. :type alignment: :class: ‘alignment.phylipFriendlyAlignment‘ :returns: A floating point value.

getLogLikelihood(*tree*, *alignment*)

Acquire log-likelihood via C library libpll.

:param tree: A tree object. :type tree: :class: ‘tree.tree‘ :param alignment: An alignment object. :type alignment: :class: ‘alignment.phylipFriendlyAlignment‘ :returns: A floating point value.

getParsimony(*newick*, *alignment*)

Acquire parsimony via a C++ implementation.

:param newick: A New Hampshire (Newick) tree string. :param alignment: An alignment object. :type alignment: :class: ‘alignment.alignment‘ :returns: An integer value.

getParsimonyForTopology(*topo*, *alignment*)

Acquire parsimony via a C++ implementation.

:param topo: A topology object. :type topo: :class: ‘rearrangement.topology‘ :param alignment: An alignment object. :type alignment: :class: ‘alignment.alignment‘ :returns: An integer value.

getParsimonyFromProfiles(*newick*, *profiles*)

Acquire parsimony via a C++ implementation.

:param newick: A New Hampshire (Newick) tree string. :param profiles: A set of profiles corresponding to an alignment. :type profiles: :class: ‘parsimony.profile_set‘ :returns: An integer value.

getParsimonyFromProfilesForTopology(*topology*, *profiles*)

Acquire parsimony via a C++ implementation.

:param topo: A topology object. :type topo: :class: ‘rearrangement.topology‘ :param profiles: A set of profiles corresponding to an alignment. :type profiles: :class: ‘parsimony.profile_set‘ :returns: An integer value.

16.2 Variables

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

17 Module pylogeny.tree

Container definition for (phylogenetic) bifurcating or multifurcating trees defined using Newick strings, collections of them, and for splits of these trees.

17.1 Functions

numberRootedTrees(*t*)

numberUnrootedTrees(*t*)

17.2 Variables

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

17.3 Class tree

object
└ **pylogeny.tree.tree**

Defines a single (phylogenetic) tree by newick string; can possess other metadata.

17.3.1 Methods

___init___(*self*, *newi*=`''`, *check*=`False`)

If enabled, "check" will force the structure to reroot the given Newick string tree to a lowest-order leaf in order to ensure a consistent Newick string among any duplicate topologies.

Overrides: object.___init___

getName(*self*)

setName(*self*, *n*)

getScore(*self*)

setScore(*self*, *s*)

getOrigin(*self*)

setOrigin(*self*, *o*)

Set the "origin" or specification of where this tree was acquired or constructed from; a string.

getNewick(*self*)

toNewick(*self*)

setNewick(*self*, *n*)

Set Newick string to n; also reacquires corresponding "structure" or Newick string without branch lengths.

getStructure(*self*)

Returns "structure", a Newick string without branch lengths.

getRerootedNoBranchLengthNewick(*self*)

See getStructure().

getSimpleNewick(*self*)

Return a Newick string with all taxa name replaced with successive integers.

toTopology(*self*)

Return a rearrangement.topology instance for this tree to allow for rearrangement of the actual structure of the tree.

___eq___(*self*, *o*)

___ne___(*self*, *o*)

___str___(*self*)

str(x)

Overrides: object.___str___ extit(inherited documentation)

Inherited from object

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

17.3.2 Properties

Name	Description
Inherited from object	
___class___	

17.4 Class treeSet

object
_abcoll.Sized
pylogeny.tree.treeSet

Known Subclasses: pylogeny.landscape.landscape

Represents an ordered, disorganized collection of trees that do not necessarily comprise a combinatorial space.

17.4.1 Methods

___init___(*self*)

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

Overrides: object.___init___ extit(inherited documentation)

addTree(*self*, *tr*)

Add a tree object to the collection.

addTreeByNewick(*self*, *newick*)

Add a tree to the structure by Newick string.

removeTree(*self*, *tr*)

Remove a tree object from the collection if present.

indexOf(*self*, *tr*)

Acquire the index in this collection of a tree object. Returns -1 if not found.

___getitem___(*self*, *i*)

___len___(*self*)

Overrides: _abcoll.Sized.___len___

toTreeFile(*self*, *fout*)

Output this landscape as a series of trees, separated by newlines, as a text file saved at the given path.

___str___(*self*)

str(x)

Overrides: object.___str___ extit(inherited documentation)

Inherited from _abcoll.Sized

___subclasshook___()

Inherited from object

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

17.4.2 Properties

Name	Description
Inherited from object ___class___	

17.4.3 Class Variables

Name	Description
___abstractmethods___	**Value:** `frozenset([])`

17.5 Class bipartition

object
└
pylogeny.tree.bipartition

A tree bipartition. Requires a tree topology. Using the term borrowed from nomenclature of a bipartite graph, a bipartition for a phylogenetic tree coincides with the definition of two disjoint sets U and V . A branch in a phylogenetic tree defines a single bipartition that divides the tree into two disjoint sets U and V . The set U comprises all of the children leaf of the subtree associated with that branch. The set V contains the rest of the leaves or taxa in the tree.

17.5.1 Methods

___init___(*self*, *topol*, *bra*=`None`)

Construct a bipartition from a branch in a topology.

:param topol: A topology. :type topol: :class: ‘rearrangement.topology‘
:param bra: An optional argument; can still acquire a bipartition from a string. :type bra: :class: ‘newick.branch‘

Overrides: object.___init___

___hash___(*self*)

hash(x)

Overrides: object.___hash___ extit(inherited documentation)

___eq___(*self*, *o*)

___ne___(*self*, *o*)

fromStringRepresentation(*self*, *st*)

Acquire all component elements from a string representation of a bipartition.

:param st: A string representation from a :class:‘.bipartition‘ object.

getBranch(*self*)

Get branch corresponding to this bipartition.

:returns: :class:‘newick.branch‘

getBranchIndex(*self*)

Return an index of the branch with respect to a post order traversal of the topology.

getStringRepresentation(*self*)

Get the string representation corresponding to this bipartition.

getShortStringRepresentation(*self*)

Get the shorter string representation corresponding to this bipartition.

getShortStringMappings(*self*)

Get the mapping of symbols from taxa names for the shorter string representation.

getBranchListRepresentation(*self*)

Get the tuple of lists of branches that represent this bipartition.

getSPRRearrangements(*self*)

Return the set of all scores related to this bipartition.

getSPRScores(*self*, *ls*, *node*=`None`)

Given a landscape, return all possible scores, not actively performing scoring if not done.

getMedianSPRScore(*self*, *ls*, *node*=`None`)

Given a landscape, return the median SPR score.

getBestSPRScore(*self*, *ls*, *node*=`None`)

Given a landscape, return the best SPR score.

Inherited from object

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

17.5.2 Properties

Name	Description
Inherited from object	
__class__	

Index