phylotoy

0

Generated by Doxygen 1.8.13

Contents

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

Class Documentation

3.1 Controller Class Reference

`#include <Controller.h>`

Public Member Functions

- Controller ()
- void SetRandomSeed (int seed)
- int GetRandomSeed ()
- int CheckRandomSeed ()
- void SetAlignmentFilePath (std::string path)
- std::string GetAlignmentFilePath ()
- std::string CheckAlignmentFilePath ()
- void SetChainName (std::string name)
- std::string GetChainName ()
- std::string CheckChainName ()
- void CheckCLIOptions ()
- void Run ()

3.1.1 Detailed Description

Definition at line 14 of file Controller.h.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 Controller()

`Controller::Controller ( )`

Definition at line 14 of file Controller.cpp.

```
14                      {
15
16
17
18 }
```

3.1.3 Member Function Documentation

3.1.3.1 CheckAlignmentFilePath()

```
std::string Controller::CheckAlignmentFilePath ( )
```

Definition at line 59 of file Controller.cpp.

```
59                                              {
60
61   //if the user forgot to set the ali path this char* will be null and we need to throw an exception
62   if (!alignment_file_path.length()){
63
64     throw "Alignment file path (option -i) was not set but is required";
65
66   }else{
67
68     return alignment_file_path;
69   }
70
71 }
```

3.1.3.2 CheckChainName()

```
std::string Controller::CheckChainName ( )
```

Definition at line 86 of file Controller.cpp.

```
86                                          {
87
88   //if the user forgot to set the chain name this char* will be null and we need to throw an exception
89   if (!chain_name.length()){
90
91     throw "Chain name (option -c) was not set but is required";
92
93   }else{
94
95     return chain_name;
96
97   }
98
99 }
```

3.1.3.3 CheckCLIOptions()

```
void Controller::CheckCLIOptions ( )
```

Definition at line 101 of file Controller.cpp.

```
101                                {
102
103   /*We need to check whether the necessary options were set*/
104
105   try {
106
107     this->CheckAlignmentFilePath();
108     this->CheckChainName();
109     this->CheckRandomSeed();
110
111   } catch (const char* exception) {
112
113     std::string what_exception(exception);
114     std::string error = "Error: " + what_exception + "\n";
115     output_printer.PrintMessage2Out(error);
116
117     exit(1);
118
119   }
120
121 }
```

3.1.3.4 CheckRandomSeed()

```
int Controller::CheckRandomSeed ( )
```

Definition at line 33 of file Controller.cpp.

```
33                                {
34
35   //if the user forgot to set the random seed this int will be null and we need to throw an exception
36   if (random_seed <= 0){
37
38     throw "Random seed (option -r) was not set but is required";
39
40   }else{
41
42     return random_seed;
43   }
44
45 }
```

3.1.3.5 GetAlignmentFilePath()

```
std::string Controller::GetAlignmentFilePath ( )
```

Definition at line 53 of file Controller.cpp.

```
53                                          {
54
55     return alignment_file_path;
56
57 }
```

3.1.3.6 GetChainName()

```
std::string Controller::GetChainName ( )
```

Definition at line 80 of file Controller.cpp.

```
80                                    {
81
82     return chain_name;
83
84 }
```

3.1.3.7 GetRandomSeed()

```
int Controller::GetRandomSeed ( )
```

Definition at line 26 of file Controller.cpp.

```
26                                {
27
28   return random_seed;
29
30
31 }
```

3.1.3.8 Run()

```
void Controller::Run ( )
```

Definition at line 124 of file Controller.cpp.

```
124                       {
125
126   this->CheckCLIOptions();
127
128   //if all option are set, we tell the user how the program was invoked.
129   std::string program_call = "phylotoy was invoked with the following options:\n\n\tRandom seed = " +
      std::to_string(random_seed) + "\n\tAlingment path = " + alignment_file_path + "\n\tChain name = " + chain_name +
      "\n\n";
130
131   output_printer.PrintMessage2Out(program_call);
132
133   //tmp vector of strings
134   std::vector<std::string>* alignment;
135
136   output_printer.PrintMessage2Out("reading alignment\n");
137   //now we need to open and store the Alignment in a string vector
138   alignment = input_reader.ReadInputFile(alignment_file_path);
139
140
141   output_printer.PrintMessage2Out("initializing tree\n");
142
143   phylo_tree.CreateBifurcatingTree(alignment);
144
145   //phylo_tree.CreateStarTree(alignment);
146
147   output_printer.PrintMessage2Out("collecting node info\n");
148   std::vector<std::string>* nodes_info = phylo_tree.CollectTreeNodesInfo();
149
150   while(!nodes_info->empty()) {
151
152     output_printer.PrintMessage2Out("in while\n");
153     output_printer.PrintMessage2Out(nodes_info->back());
154     output_printer.PrintMessage2Out("\n");
155
156     nodes_info->pop_back();
157   }
158
159
160 }
```

3.1.3.9 SetAlignmentFilePath()

```
void Controller::SetAlignmentFilePath (
            std::string path )
```

Definition at line 47 of file Controller.cpp.

```
47                                                  {
48
49   alignment_file_path = path;
50
51 }
```

3.1.3.10 SetChainName()

```
void Controller::SetChainName (
            std::string name )
```

Definition at line 74 of file Controller.cpp.

```
74                                              {
75
76   chain_name = name;
77
78 }
```

3.1.3.11 SetRandomSeed()

```
void Controller::SetRandomSeed (
            int seed )
```

Definition at line 20 of file Controller.cpp.

```
20                                          {
21
22   random_seed = seed;
23
24 }
```

The documentation for this class was generated from the following files:

- /home/sergio/Repos/phylotoy/src/Controller.h
- /home/sergio/Repos/phylotoy/src/Controller.cpp

3.2 InputReader Class Reference

```
#include <InputReader.h>
```

Public Member Functions

- InputReader ()
- InputReader (std::string path)
- void SetPath (std::string path)
- std::string GetPath ()
- std::vector< std::string > * ReadInputFile (std::string path)

3.2.1 Detailed Description

Definition at line 11 of file InputReader.h.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 InputReader() [1/2]

```
InputReader::InputReader ( )
```

Definition at line 13 of file InputReader.cpp.

```
13 {}
```

3.2.2.2 InputReader() [2/2]

```
InputReader::InputReader (
            std::string path )
```

Definition at line 15 of file InputReader.cpp.

```
15                                      {
16
17   input_file_path = path;
18
19 }
```

3.2.3 Member Function Documentation

3.2.3.1 GetPath()

```
std::string InputReader::GetPath ( )
```

Definition at line 27 of file InputReader.cpp.

```
27                                {
28
29   return input_file_path;
30
31 }
```

3.2.3.2 ReadInputFile()

```
std::vector< std::string > * InputReader::ReadInputFile (
              std::string path )
```

Definition at line 33 of file InputReader.cpp.

```
33                                                                {
34
35   /* create and open the input stream. This will always be done, we need to control later if the stream
36    * is good or not...
37   */
38
39   std::ifstream input_stream(path);
40
41   //create a vector of strings to store the data
42   std::vector<std::string>* lines = new std::vector<std::string>;
43
44   if(input_stream.good()) {
45
46     //dummy string to store the first line
47     std::string first_line;
48
49     //read the first line
50     std::getline(input_stream, first_line);
51
52     //string to store the lines.
53     std::string input_line;
54
55     //now read the file line by line and push the line into the vector
56     while(std::getline(input_stream, input_line)) {
57
58       //add the lines to the string vector
59       lines->push_back(input_line);
60
61     }
62
63     //input_stream.close();
64
65   }else {
66
67     std::cerr << "Something went wrong reading the alignment file: " << path << "\n";
68     exit(1);
69
70   }
71
72
73   return lines;
74
75 }
```

3.2.3.3 SetPath()

```
void InputReader::SetPath (
            std::string path )
```

Definition at line 21 of file InputReader.cpp.

```
21                                      {
22
23   input_file_path = path;
24
25 }
```

The documentation for this class was generated from the following files:

- /home/sergio/Repos/phylotoy/src/InputReader.h
- /home/sergio/Repos/phylotoy/src/InputReader.cpp

3.3 Node Class Reference

```
#include <Node.h>
```

Public Member Functions

- Node ()
- void SetSequence (std::string species_sequence)
- std::string GetSequence ()
- void SetSpeciesName (std::string name)
- std::string GetSpeciesName ()
- void SetIsTip (bool tip)
- bool GetIsTip ()
- void SetParentNode (Node *parent)
- Node * GetParentNode ()
- void SetLengthSubtendingBranch (float branch_length)
- float GetLengthSubtendingBranch ()
- void AddNodeToChildVector (Node *child)
- void SetChildVector (std::vector< Node *> childs)
- std::vector< Node * > GetChildVector ()
- void CreateBifurcatingNode (std::vector< Node *>, int *calls)
- std::vector< std::string > * GetNodeInfo (std::vector< std::string > *collected_node_info)

3.3.1 Detailed Description

Definition at line 11 of file Node.h.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 Node()

```
Node::Node (
            void  )
```

Definition at line 14 of file Node.cpp.

```
14 {}
```

3.3.3 Member Function Documentation

3.3.3.1 AddNodeToChildVector()

```
void Node::AddNodeToChildVector (
            Node * child )
```

Definition at line 80 of file Node.cpp.

```
80                                          {
81
82   child_nodes.push_back(child);
83
84 }
```

3.3.3.2 CreateBifurcatingNode()

```
void Node::CreateBifurcatingNode (
            std::vector< Node *> tip_nodes,
            int * calls )
```

Definition at line 98 of file Node.cpp.

```
98                                                                      {
99
100   std::cerr << "creating bifurcating node\n";
101
102   *calls = *calls +1;
103
104   if(!tip_nodes.empty()){
105
106     Node* tip_node_to_insert = tip_nodes.back();
107
108     /* We create the tree as follows:
109      * 1. the current node has no children, i.e. the size of the vector child_nodes is 0.
110      * We insert the tip node as a child of this node, sending a reference of it to the child node to point
       to its parent node.
111      * Once this is done, we create a new internode, set this node as its parent and add it to the
       child_nodes vector of the current node.
112      * Then we recursively call this method on the new internode but only if we still have more than 1 tip
       nodes left.
113      *
114      */
115
116     std::cerr << "inserting tip: " << tip_node_to_insert->GetSpeciesName() << "\n";
117     tip_node_to_insert->SetParentNode(this);
118     this->AddNodeToChildVector(tip_node_to_insert);
```

```
119     tip_nodes.pop_back();//this deletes the last element of the array
120
121     if(tip_nodes.size() > 1){
122
123       std::cerr << "inserting new internode " << *calls << "\n";
124
125       Node* internode = new Node();
126       internode->SetIsTip(false);
127       internode->SetParentNode(this);
128       internode->SetSpeciesName(std::to_string(*calls));
129       this->AddNodeToChildVector(internode);
130
131       internode->CreateBifurcatingNode(tip_nodes, calls);
132
133     }else{
134       //there is only one tip left, this we can add to the current internode because we only added 1 tip to
      it and only 1 remains to be added
135
136       std::cerr << "inserting last tip: " << tip_node_to_insert->GetSpeciesName() << "\n";
137       tip_node_to_insert->SetParentNode(this);
138       this->AddNodeToChildVector(tip_node_to_insert);
139       tip_nodes.pop_back();//this deletes the last element of the array
140
141     }
142   }
143 }
```

3.3.3.3 GetChildVector()

```
std::vector< Node * > Node::GetChildVector ( )
```

Definition at line 92 of file Node.cpp.

```
92                                      {
93
94   return child_nodes;
95
96 }
```

3.3.3.4 GetIsTip()

```
bool Node::GetIsTip ( )
```

Definition at line 48 of file Node.cpp.

```
48                      {
49
50   return is_tip;
51
52 }
```

3.3.3.5 GetLengthSubtendingBranch()

```
float Node::GetLengthSubtendingBranch ( )
```

Definition at line 61 of file Node.cpp.

```
61                                      {
62
63   return length_of_subtending_branch;
64
65 }
```

3.3.3.6 GetNodeInfo()

```
std::vector< std::string > * Node::GetNodeInfo (
            std::vector< std::string > * collected_node_info )
```

Definition at line 145 of file Node.cpp.

```
145                                                                                 {
146
147   if(child_nodes.empty()) {
148
149     std::cerr << "empty node vector\n";
150
151     //add the species name and sequence to the vector collecting the node information
152     std::string node_info = species_name + ' ' + sequence;
153
154     std::cerr << "adding " << node_info << "to info vector\n";
155
156     collected_node_info->push_back(node_info);
157
158
159   }else {
160
161     std::cerr << "recursively calling nodes\n";
162     //for each child node, call this function.
163     for(auto child : child_nodes) {
164
165       std::cerr << "in for\n";
166       child->GetNodeInfo(collected_node_info);
167
168     }
169
170     std::cerr << "back at internode \n";
171
172     //add your own info if needed
173
174   }
175
176   return collected_node_info;
177
178 }
```

3.3.3.7 GetParentNode()

```
Node * Node::GetParentNode ( )
```

Definition at line 73 of file Node.cpp.

```
73                          {
74
75   return parent_node;
76
77 }
```

3.3.3.8 GetSequence()

```
std::string Node::GetSequence ( )
```

Definition at line 23 of file Node.cpp.

```
23                          {
24
25   return sequence;
26
27 }
```

3.3.3.9 GetSpeciesName()

```
std::string Node::GetSpeciesName ( )
```

Definition at line 36 of file Node.cpp.

```
36                            {
37
38   return species_name;
39
40 }
```

3.3.3.10 SetChildVector()

```
void Node::SetChildVector (
             std::vector< Node *> childs )
```

Definition at line 86 of file Node.cpp.

```
86                                          {
87
88   child_nodes = childs;
89
90 }
```

3.3.3.11 SetIsTip()

```
void Node::SetIsTip (
             bool tip )
```

Definition at line 42 of file Node.cpp.

```
42                          {
43
44   is_tip = tip;
45
46 }
```

3.3.3.12 SetLengthSubtendingBranch()

```
void Node::SetLengthSubtendingBranch (
            float branch_length )
```

Definition at line 55 of file Node.cpp.

```
55                                                          {
56
57   length_of_subtending_branch = branch_length;
58
59 }
```

3.3.3.13 SetParentNode()

```
void Node::SetParentNode (
            Node * parent )
```

Definition at line 67 of file Node.cpp.

```
67                                      {
68
69   parent_node = parent;
70
71 }
```

3.3.3.14 SetSequence()

```
void Node::SetSequence (
            std::string species_sequence )
```

Definition at line 16 of file Node.cpp.

```
16                                                  {
17
18   std::cerr << "setting sequence\n";
19   sequence = species_sequence;
20
21 }
```

3.3.3.15 SetSpeciesName()

```
void Node::SetSpeciesName (
            std::string name )
```

Definition at line 29 of file Node.cpp.

```
29                                      {
30
31   std::cerr << "setting species name\n";
32   species_name = name;
33
34 }
```

The documentation for this class was generated from the following files:

- /home/sergio/Repos/phylotoy/src/Node.h
- /home/sergio/Repos/phylotoy/src/Node.cpp

3.4 OutputPrinter Class Reference

```
#include <OutputPrinter.h>
```

Public Member Functions

- OutputPrinter ()
- void PrintMessage2Out (std::string text)

3.4.1 Detailed Description

Definition at line 10 of file OutputPrinter.h.

3.4.2 Constructor & Destructor Documentation

3.4.2.1 OutputPrinter()

```
OutputPrinter::OutputPrinter ( )
```

Definition at line 11 of file OutputPrinter.cpp.

```
11 {}
```

3.4.3 Member Function Documentation

3.4.3.1 PrintMessage2Out()

```
void OutputPrinter::PrintMessage2Out (
             std::string text )
```

Definition at line 13 of file OutputPrinter.cpp.

```
13                                                  {
14
15   std::cout << text;
16
17 }
```

The documentation for this class was generated from the following files:

- /home/sergio/Repos/phylotoy/src/OutputPrinter.h
- /home/sergio/Repos/phylotoy/src/OutputPrinter.cpp

3.5 Tree Class Reference

```
#include <Tree.h>
```

Public Member Functions

- Tree ()
- void SetLength (float length)
- float GetLength ()
- void SetRoot (Node *root_node)
- Node * GetRoot ()
- void CreateStarTree (std::vector< std::string > *alignment)
- void CreateBifurcatingTree (std::vector< std::string > *alignment)
- std::vector< std::string > * CollectTreeNodesInfo ()

3.5.1 Detailed Description

Objects of the class Tree represent phylogenetic trees. These trees are (1) unrooted (the root node used in the code is just an internode of the tree, (2) can be initialized as start trees, (3) can contain polytomies, or (4) can be bifurcating.

This class provide a number of methods to manipulate the tree. For instance, re-rooting the tree. It also provides methods to alter the tree topology using Nearest-neighbor interchange (NNI) or subtree pruning and regraphting (SPR).

A number of attributes of the tree provide short-cuts to make modifying the tree topology or the length of the tree's branches easy. For instance, Tree objects store pointers to all of their nodes and to the current root. This makes easy to rearrange the tree during MCMC.

The Tree Class is also in charge of proposing all modifications to tree topology and branch length.

Definition at line 19 of file Tree.h.

3.5.2 Constructor & Destructor Documentation

3.5.2.1 Tree()

```
Tree::Tree ( )
```

Definition at line 12 of file Tree.cpp.

```
12          {
13
14   current_root =  new Node;
15   current_root->SetIsTip(false);
16
17 }
```

3.5.3 Member Function Documentation

3.5.3.1 CollectTreeNodesInfo()

```
std::vector< std::string > * Tree::CollectTreeNodesInfo ( )
```

This method returns information on the nodes currently included in the tree. It recursively traverses the tree starting from the root node and asks each node for its info as a string that is stored on a vector of strings.

Returns

a vector of strings

Definition at line 89 of file Tree.cpp.

```
89                                                {
90
91   std::vector<std::string>* tree_nodes_info = new std::vector<std::string>;
92
93   std::cerr << "at tree root\n";
94   tree_nodes_info = current_root->GetNodeInfo(tree_nodes_info);
95
96   return tree_nodes_info;
97 }
```

3.5.3.2 CreateBifurcatingTree()

```
void Tree::CreateBifurcatingTree (
             std::vector< std::string > * alignment )
```

This method creates a bifurcating tree. This method creates as many internodes as required to yield a bifurcating tree.

Parameters

alignment	is a sequence alignment stored on a vector of strings containing species names and sequencesseparated by an empty space.

Definition at line 71 of file Tree.cpp.

```
71                                                                  {
72
73   std::cerr << "Creating bifurcating tree\n";
74
75   int recursive_calls = 0;
76
77   //we initialize the tip nodes using the alignment provided by the user
78   current_root->CreateBifurcatingNode(this->InitializeTipNodes(alignment), &
      recursive_calls);
79
80 }
```

3.5.3.3 CreateStarTree()

```
void Tree::CreateStarTree (
              std::vector< std::string > * alignment )
```

This method creates a star tree. This tree adds all the tip nodes in one alignment to the root node.

Parameters

alignment	is a sequence alignment with species names and sequences separated by an empty space.

Definition at line 49 of file Tree.cpp.

```
49                                                          {
50
51   std::cerr << "Creating star tree\n";
52
53   //first we send the alignment to our private tip node initializer
54   current_root->SetChildVector(this->InitializeTipNodes(alignment));
55
56   //once all tips have been added as childs to the root, the root sends a pointer to him self to each of
       the child nodes.
57
58   for(auto child : current_root->GetChildVector()){
59
60     child->SetParentNode(current_root);
61     //we also need to set the length of the subtending branch leading to the parent
62   }
63
64 }
```

3.5.3.4 GetLength()

```
float Tree::GetLength ( )
```

Definition at line 25 of file Tree.cpp.

```
25                      {
26
27   return length;
28
29 }
```

3.5.3.5 GetRoot()

```
Node * Tree::GetRoot ( )
```

Definition at line 38 of file Tree.cpp.

```
38                     {
39
40   return current_root;
41
42 }
```

3.5.3.6 SetLength()

```
void Tree::SetLength (
              float length )
```

Definition at line 19 of file Tree.cpp.

```
19                                {
20
21     length = length;
22
23 }
```

3.5.3.7 SetRoot()

```
void Tree::SetRoot (
              Node * root_node )
```

Definition at line 31 of file Tree.cpp.

```
31                                 {
32
33   current_root = root_node;
34
35 }
```

The documentation for this class was generated from the following files:

- /home/sergio/Repos/phylotoy/src/Tree.h
- /home/sergio/Repos/phylotoy/src/Tree.cpp

Chapter 4

File Documentation

4.1 /home/sergio/Repos/phylotoy/src/Controller.cpp File Reference

```
#include "Controller.h"
#include <string>
#include <exception>
#include <assert.h>
#include <vector>
```

4.2 /home/sergio/Repos/phylotoy/src/Controller.h File Reference

```
#include "InputReader.h"
#include "OutputPrinter.h"
#include "Tree.h"
#include <string>
#include <vector>
```

Classes

- class Controller

4.3 /home/sergio/Repos/phylotoy/src/InputReader.cpp File Reference

```
#include "InputReader.h"
#include <fstream>
#include <vector>
#include <iostream>
```

4.4 /home/sergio/Repos/phylotoy/src/InputReader.h File Reference

```
#include <vector>
#include <string>
```

Classes

- class InputReader

4.5 /home/sergio/Repos/phylotoy/src/Node.cpp File Reference

```
#include <vector>
#include <string>
#include "Node.h"
#include <iostream>
```

4.6 /home/sergio/Repos/phylotoy/src/Node.h File Reference

```
#include <vector>
#include <string>
```

Classes

- class Node

4.7 /home/sergio/Repos/phylotoy/src/OutputPrinter.cpp File Reference

```
#include <iostream>
#include "OutputPrinter.h"
```

4.8 /home/sergio/Repos/phylotoy/src/OutputPrinter.h File Reference

```
#include <string>
```

Classes

- class OutputPrinter

4.9 /home/sergio/Repos/phylotoy/src/Phylotoy.cpp File Reference

```
#include <unistd.h>
#include <stdlib.h>
#include "Controller.h"
#include <string>
```

Functions

- int main (int argc, char ∗argv[ ])

4.9.1 Function Documentation

4.9.1.1 main()

```
int main (
            int argc,
            char * argv[ ] )
```

Definition at line 13 of file Phylotoy.cpp.

```
13                                {
14
15   //Initialize a controller object
16
17   Controller phylotoy_controller;
18
19   /* Read the CLI options set the appropriate variables in the controller
20    * Options are passsed as follows:
21    * -r random seed
22    * -i alignment path
23    * -c chain name
24    */
25
26   int option;
27
28   while ((option = getopt(argc, argv, "r:i:c:")) != -1) {
29
30     switch (option){
31
32       case 'r':
33       {
34         phylotoy_controller.SetRandomSeed(atoi(optarg));
35         break;
36       }
37       case 'i':
38       {
39         std::string input (optarg);
40         phylotoy_controller.SetAlignmentFilePath(input);
41         break;
42       }
43       case 'c':
44       {
45         std::string name (optarg);
46         phylotoy_controller.SetChainName(name);
47         break;
48       }
49       default:
50         abort();
51     }
52   }
53
54   phylotoy_controller.Run();
55
56   return 0;
57 }
```

4.10 /home/sergio/Repos/phylotoy/src/Tree.cpp File Reference

```
#include <vector>
#include <string>
#include "Tree.h"
#include <sstream>
#include <iostream>
```

4.11 /home/sergio/Repos/phylotoy/src/Tree.h File Reference

```
#include <vector>
#include <string>
#include "Node.h"
```

Classes

- class Tree

Index