FLER-FollicularLymphomaEvolutionReconstructor

Generated by Doxygen 1.6.3

Thu May 27 14:27:40 2010

Contents

Chapter 1

FLER - Follicular Lymphoma Evolution Reconstructor

Author

Code: Peter Vasil, Technical University, Berlin
Method: Martin Wartenberg, Technical University, Munich

Version

1.0

Date

2010-05-10

Description:

FLER is the implementation of the correspondent mathematic/statistic algorithm to calculate the most likely evolution of monoclonal operational taxonomic units (OTU) driven by somatic hypermutation. Thus, given a group of OTUs defined by various mutation patterns of a certain DNA-sequence (compared to an unmutated wildtype configuration (germline)), FLER uses the number of equally mutated loci among subgroups of OTUs, that share different spectra of mutations, as foundation for creation of hypothetical predecessor clones (HPCs). Among the abundance of HPCs the most probable HPCs are chosen to recapitulate evolution from the wildtype configuration to each OTU. Probability is assessed by calculation of the probability measure "pm" for each HPC. Input parameters of the pm-formula are:

$$pm = \sqrt[(t-r+1)\cdot i\cdot z]{\prod_{k=1}^{l} P_k}$$

r: number of OTUs in an OTU subgroup, defining the considered HPC

i: nuber of mutations of an HPC (thus number of mutations shared by all clones of the HPC-defining subgroup)

z: number of reapeats of an HPC with the identical mutation patterns among the abundance of HPCs

t: total number of OTUs

P: relative frequency of a certain mutation among the considered population of OTUs

l: sequence length

k: counter, starting at locus 1 and stepping through the sequence until the last locus l is reached

Unlike multiple sequence alignment (MSA) algorithms, like the neighbor joining (NJ) algorithm, FLER calculates the most likely HPCs from mutation frequencies and considers the mutation number as surrogate marker for elapsed time. Thus clonal interchange among different populations of OTUs can also be assessed by doing population-specific assays first, complemented by an assay comprising all OTUs as one population. By comparing pm-values of HPCs of different assays, interchange among various populations rather than population-contained evolution can be evaluated. Then, migration directions are determined by the population affiliation of the majority of HPC-defining OTUs.

Help:

Graphical user interface (GUI):

sequence length: - user can determine the sequence length (all sequences must have the identical length).

GERMLINE - user can choose the germline sequence. This sequence will be regarded as unmutated wildtype configuration.

Each clone sequence will be compared to the germline sequence to determine the clone-specific mutation pattern.

CLONES - user can choose the clones, defining the sequence population for that the evolution reconstruction will be implemented by FLER.

RUN FLER - starts the algorithm

SAVE FILE - user can choose where to save the file

RESET - sets the default for a new run of FLER

Input format / data:

Individual DNA sequences have to be saved in a plain text file (.txt), each. The sequence itself has to be written in the first line only, with no figures (lower or capital case) except for A (adenine), C (cytidine), T (thymine) , G (guanine). The name of the txt-file will be used as clone name by FLER (example: the sequence contained in file "clone1.txt" will be addressed as "clone1"). FLER accounts for single nucleotid point mutations only. Insertions, deletions, dublications or other chromosomal aberrations are not considered, and thus not to be included.

Output data:

The user can choose whether to display only the pm-directed hierarchy of HPCs, the groups of OTUs defining a certain HPC, or individual clonal metatdata like mutated loci, mutation number etc. (see "save options" in the menu bar of FLER). All output data will be saved in a file the user can determine interactively. It is recommended to use the extension <.txt>.

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Class Documentation

4.1 AppClass Class Reference

Application class.

Public Member Functions

- AppClass ()
- ~AppClass ()
- void initialise (const String &commandLine)
- void shutdown ()
- const String getApplicationName ()
- const String getApplicationVersion ()
- bool moreThanOneInstanceAllowed ()
- void anotherInstanceStarted (const String &commandLine)

Private Attributes

- MainAppWindow * theMainWindow
- ApplicationCommandManager * m_commandManager

4.1.1 Detailed Description

Application class.

Definition at line 112 of file Main.cpp.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 AppClass::AppClass () `[inline]`

Definition at line 120 of file Main.cpp.

4.1.2.2 AppClass::~AppClass () [inline]

Definition at line 126 of file Main.cpp.

4.1.3 Member Function Documentation

4.1.3.1 void AppClass::anotherInstanceStarted (const String & *commandLine*) [inline]

Definition at line 163 of file Main.cpp.

4.1.3.2 const String AppClass::getApplicationName () [inline]

Definition at line 148 of file Main.cpp.

4.1.3.3 const String AppClass::getApplicationVersion () [inline]

Definition at line 153 of file Main.cpp.

4.1.3.4 void AppClass::initialise (const String & *commandLine*) [inline]

Definition at line 131 of file Main.cpp.

4.1.3.5 bool AppClass::moreThanOneInstanceAllowed () [inline]

Definition at line 158 of file Main.cpp.

4.1.3.6 void AppClass::shutdown () [inline]

Definition at line 141 of file Main.cpp.

4.1.4 Member Data Documentation

4.1.4.1 ApplicationCommandManager* AppClass::m_commandManager [private]

Definition at line 116 of file Main.cpp.

4.1.4.2 MainAppWindow* AppClass::theMainWindow [private]

Definition at line 114 of file Main.cpp.

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

- Main.cpp

4.2 Clone Struct Reference

Struct for name and sequence.

`#include <Data.h>`

Public Attributes

- String name
- String seq

4.2.1 Detailed Description

Struct for name and sequence.

Definition at line 28 of file Data.h.

4.2.2 Member Data Documentation

4.2.2.1 String Clone::name

Definition at line 29 of file Data.h.

4.2.2.2 String Clone::seq

Definition at line 30 of file Data.h.

The documentation for this struct was generated from the following file:

- Data.h

4.3 CloneCalculations Class Reference

Class for clone calculations.

`#include <CloneCalculations.h>`

Public Member Functions

- CloneCalculations ()
- ~CloneCalculations ()
- void setSeqLength (const int seqLength)

  Set sequence length.

- void setGlFile (const String &glFilePath)

  Set germline.

- void setClones (const std::vector< Clone > &clones)

  Set clones.

- const int getSeqLength () const
- const std::vector< Clone > & getClones () const
- const String & getGlString () const
- const std::vector< CloneVsGl > & getCloneVsGl () const
- const std::vector< CloneVsGl > & getCloneVsGlSorted () const
- const std::vector< std::vector< CloneVsGl > > & getCloneRefs () const
- const std::vector< CloneRefsAndPosition > & getCloneRefsSorted () const
- const std::vector< IntermediateClones > & getIntermediateClones () const
- const std::vector< IntermediateClones > & getIntermediateClonesSorted () const
- const std::vector< IntermediateClones > & getIntermediateClonesUnique () const
- const std::vector< IntermediateClones > & getIntermediateClonesUniqueSorted () const
- const std::vector< int > & getDiffTimesA () const
- const std::vector< int > & getDiffTimesC () const
- const std::vector< int > & getDiffTimesT () const
- const std::vector< int > & getDiffTimesG () const
- const std::vector< double > & getDiffTimesARelative () const
- const std::vector< double > & getDiffTimesCRelative () const
- const std::vector< double > & getDiffTimesTRelative () const
- const std::vector< double > & getDiffTimesGRelative () const
- const String & getStatusMessage () const
- void calculate ()

  Calculate evrything.

- void clearValues ()

  Clear all calculation values.

Private Member Functions

- void checkCloneVsGl ()

  Create clone information regards germline.

- void sortClones ()

  Sort clones.

- void countMutationOnPosition ()

  Generate information for mutation on positions.

- void calculateIntermediateClones ()

  Calculate HPCs.

- void calculateIntermediateClonesUnique ()

  Calculate HPCs unique.

- void calculatePm ()

  Calculate pm hierarchy.

- CloneCalculations (const CloneCalculations &)

  Prevent copy constructor being generated.

- const CloneCalculations & operator= (const CloneCalculations &)

  Prevent operator= being generated.

Private Attributes

- String m_gl
- std::vector< Clone > m_cloneVec
- std::vector< CloneVsGl > m_cloneVsGl
- std::vector< CloneVsGl > m_cloneVsGlSorted
- std::vector< std::vector< CloneVsGl > > m_cloneRefs
- std::vector< CloneRefsAndPosition > m_cloneRefsSorted
- std::vector< IntermediateClones > m_intermediatClones
- std::vector< IntermediateClones > m_intermediatClonesSorted
- std::vector< IntermediateClones > m_intermediateClonesUnique
- std::vector< IntermediateClones > m_intermediateClonesUniqueSorted
- int m_seqLength
- std::vector< int > m_diffTimesA
- std::vector< int > m_diffTimesC
- std::vector< int > m_diffTimesT
- std::vector< int > m_diffTimesG
- std::vector< double > m_diffTimesARelative
- std::vector< double > m_diffTimesCRelative
- std::vector< double > m_diffTimesTRelative
- std::vector< double > m_diffTimesGRelative
- String m_statusText

4.3.1 Detailed Description

Class for clone calculations.

Author

Peter Vasil, Technical University, Berlin
Martin Wartenberg, Technical University, Munich

Definition at line 30 of file CloneCalculations.h.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 CloneCalculations::CloneCalculations ()

Definition at line 23 of file CloneCalculations.cpp.

4.3.2.2 CloneCalculations::~CloneCalculations ()

Definition at line 32 of file CloneCalculations.cpp.

4.3.2.3 CloneCalculations::CloneCalculations (const CloneCalculations &) [private]

Prevent copy constructor being generated.

4.3.3 Member Function Documentation

4.3.3.1 void CloneCalculations::calculate ()

Calculate evrything.

Definition at line 445 of file CloneCalculations.cpp.

4.3.3.2 void CloneCalculations::calculateIntermediateClones () [private]

Calculate HPCs.

Definition at line 175 of file CloneCalculations.cpp.

4.3.3.3 void CloneCalculations::calculateIntermediateClonesUnique () [private]

Calculate HPCs unique.

Definition at line 253 of file CloneCalculations.cpp.

4.3.3.4 void CloneCalculations::calculatePm () [private]

Calculate pm hierarchy.

Definition at line 378 of file CloneCalculations.cpp.

4.3.3.5 void CloneCalculations::checkCloneVsGl () [private]

Create clone information regards germline.

Definition at line 47 of file CloneCalculations.cpp.

4.3.3.6 void CloneCalculations::clearValues ()

Clear all calculation values.

Definition at line 460 of file CloneCalculations.cpp.

4.3.3.7 void CloneCalculations::countMutationOnPosition () [private]

Generate information for mutation on positions.

Definition at line 89 of file CloneCalculations.cpp.

4.3.3.8 const std::vector<std::vector<CloneVsGl> >& CloneCalculations::getCloneRefs () const [inline]

Definition at line 58 of file CloneCalculations.h.

4.3.3.9 const std::vector<CloneRefsAndPosition>& CloneCalculations::getCloneRefsSorted () const [inline]

Definition at line 59 of file CloneCalculations.h.

4.3.3.10 const std::vector<Clone>& CloneCalculations::getClones () const [inline]

Definition at line 54 of file CloneCalculations.h.

4.3.3.11 const std::vector<CloneVsGl>& CloneCalculations::getCloneVsGl () const [inline]

Definition at line 56 of file CloneCalculations.h.

4.3.3.12 const std::vector<CloneVsGl>& CloneCalculations::getCloneVsGlSorted () const [inline]

Definition at line 57 of file CloneCalculations.h.

4.3.3.13 const std::vector<int>& CloneCalculations::getDiffTimesA () const [inline]

Definition at line 64 of file CloneCalculations.h.

4.3.3.14 const std::vector<double>& CloneCalculations::getDiffTimesARelative () const [inline]

Definition at line 68 of file CloneCalculations.h.

4.3.3.15 const std::vector<int>& CloneCalculations::getDiffTimesC () const [inline]

Definition at line 65 of file CloneCalculations.h.

4.3.3.16 const std::vector<double>& CloneCalculations::getDiffTimesCRelative () const [inline]

Definition at line 69 of file CloneCalculations.h.

4.3.3.17 const std::vector<int>& CloneCalculations::getDiffTimesG () const [inline]

Definition at line 67 of file CloneCalculations.h.

4.3.3.18 const std::vector<double>& CloneCalculations::getDiffTimesGRelative () const [inline]

Definition at line 71 of file CloneCalculations.h.

4.3.3.19 const std::vector<int>& CloneCalculations::getDiffTimesT () const [inline]

Definition at line 66 of file CloneCalculations.h.

4.3.3.20 const std::vector<double>& CloneCalculations::getDiffTimesTRelative () const [inline]

Definition at line 70 of file CloneCalculations.h.

4.3.3.21 const String& CloneCalculations::getGlString () const [inline]

Definition at line 55 of file CloneCalculations.h.

4.3.3.22 const std::vector<IntermediateClones>& CloneCalculations::getIntermediateClones () const [inline]

Definition at line 60 of file CloneCalculations.h.

4.3.3.23 const std::vector<IntermediateClones>& CloneCalculations::getIntermediateClonesSorted () const [inline]

Definition at line 61 of file CloneCalculations.h.

4.3.3.24 const std::vector<IntermediateClones>& CloneCalculations::getIntermediateClonesUnique () const [inline]

Definition at line 62 of file CloneCalculations.h.

4.3.3.25 const std::vector<IntermediateClones>& CloneCalculations::getIntermediateClonesUniqueSorted () const [inline]

Definition at line 63 of file CloneCalculations.h.

4.3.3.26 const int CloneCalculations::getSeqLength () const [inline]

Definition at line 53 of file CloneCalculations.h.

4.3.3.27 const String& CloneCalculations::getStatusMessage () const [inline]

Definition at line 73 of file CloneCalculations.h.

4.3.3.28 const CloneCalculations& CloneCalculations::operator= (const CloneCalculations &) [private]

Prevent operator= being generated.

4.3.3.29 void CloneCalculations::setClones (const std::vector< Clone > & *clones*) [inline]

Set clones.

Parameters

clones a vector with clones.

Definition at line 51 of file CloneCalculations.h.

4.3.3.30 void CloneCalculations::setGlFile (const String & *glFilePath*) [inline]

Set germline.

Parameters

glFilePath path of germline file.

Definition at line 46 of file CloneCalculations.h.

4.3.3.31 void CloneCalculations::setSeqLength (const int *seqLength*) [inline]

Set sequence length.

Parameters

seqLength sequence length

Definition at line 41 of file CloneCalculations.h.

4.3.3.32 void CloneCalculations::sortClones () [private]

Sort clones.

Definition at line 76 of file CloneCalculations.cpp.

4.3.4 Member Data Documentation

4.3.4.1 std::vector<std::vector<CloneVsGl> > CloneCalculations::m_cloneRefs [private]

Definition at line 122 of file CloneCalculations.h.

4.3.4.2 std::vector<CloneRefsAndPosition> CloneCalculations::m_cloneRefsSorted [private]

Definition at line 123 of file CloneCalculations.h.

4.3.4.3 std::vector<Clone> CloneCalculations::m_cloneVec [private]

Definition at line 119 of file CloneCalculations.h.

4.3.4.4 std::vector<CloneVsGl> CloneCalculations::m_cloneVsGl [private]

Definition at line 120 of file CloneCalculations.h.

4.3.4.5 std::vector<CloneVsGl> CloneCalculations::m_cloneVsGlSorted [private]

Definition at line 121 of file CloneCalculations.h.

4.3.4.6 std::vector<int> CloneCalculations::m_diffTimesA [private]

Definition at line 130 of file CloneCalculations.h.

4.3.4.7 std::vector<double> CloneCalculations::m_diffTimesARelative [private]

Definition at line 134 of file CloneCalculations.h.

4.3.4.8 std::vector<int> CloneCalculations::m_diffTimesC [private]

Definition at line 131 of file CloneCalculations.h.

4.3.4.9 std::vector<double> CloneCalculations::m_diffTimesCRelative [private]

Definition at line 135 of file CloneCalculations.h.

4.3.4.10 std::vector<int> CloneCalculations::m_diffTimesG [private]

Definition at line 133 of file CloneCalculations.h.

4.3.4.11 std::vector<double> CloneCalculations::m_diffTimesGRelative [private]

Definition at line 137 of file CloneCalculations.h.

4.3.4.12 std::vector<int> CloneCalculations::m_diffTimesT [private]

Definition at line 132 of file CloneCalculations.h.

4.3.4.13 std::vector<double> CloneCalculations::m_diffTimesTRelative [private]

Definition at line 136 of file CloneCalculations.h.

4.3.4.14 String CloneCalculations::m_gl [private]

Definition at line 118 of file CloneCalculations.h.

4.3.4.15 std::vector<IntermediateClones> CloneCalculations::m_intermediatClones [private]

Definition at line 124 of file CloneCalculations.h.

4.3.4.16 std::vector<IntermediateClones> CloneCalculations::m_intermediatClonesSorted [private]

Definition at line 125 of file CloneCalculations.h.

4.3.4.17 std::vector<IntermediateClones> CloneCalculations::m_intermediateClonesUnique [private]

Definition at line 126 of file CloneCalculations.h.

4.3.4.18 std::vector<IntermediateClones> CloneCalculations::m_-intermediateClonesUniqueSorted [private]

Definition at line 127 of file CloneCalculations.h.

4.3.4.19 int CloneCalculations::m_seqLength [private]

Definition at line 129 of file CloneCalculations.h.

4.3.4.20 String CloneCalculations::m_statusText [private]

Definition at line 138 of file CloneCalculations.h.

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

- CloneCalculations.h
- CloneCalculations.cpp

4.4 CloneRefsAndPosition Struct Reference

Clones with locus.

`#include <Data.h>`

Public Member Functions

- bool operator() (CloneRefsAndPosition lhs, CloneRefsAndPosition rhs)

Public Attributes

- int pos
- std::vector< CloneVsGl > clones

4.4.1 Detailed Description

Clones with locus.

Definition at line 53 of file Data.h.

4.4.2 Member Function Documentation

4.4.2.1 bool CloneRefsAndPosition::operator() (CloneRefsAndPosition *lhs*, CloneRefsAndPosition *rhs*) [inline]

Definition at line 56 of file Data.h.

4.4.3 Member Data Documentation

4.4.3.1 std::vector<CloneVsGl> CloneRefsAndPosition::clones

Definition at line 55 of file Data.h.

4.4.3.2 int CloneRefsAndPosition::pos

Definition at line 54 of file Data.h.

The documentation for this struct was generated from the following file:

- Data.h

4.5 CloneVsGl Struct Reference

Clones with metadata.

`#include <Data.h>`

Public Member Functions

- bool operator() (CloneVsGl lhs, CloneVsGl rhs)

Public Attributes

- String name
- String seq
- String diff
- std::vector< int > mutPos
- int mutNum

4.5.1 Detailed Description

Clones with metadata.

Definition at line 37 of file Data.h.

4.5.2 Member Function Documentation

4.5.2.1 bool CloneVsGl::operator() (CloneVsGl *lhs*, CloneVsGl *rhs*) [inline]

Definition at line 43 of file Data.h.

4.5.3 Member Data Documentation

4.5.3.1 String CloneVsGl::diff

Definition at line 40 of file Data.h.

4.5.3.2 int CloneVsGl::mutNum

Definition at line 42 of file Data.h.

4.5.3.3 std::vector<int> CloneVsGl::mutPos

Definition at line 41 of file Data.h.

4.5.3.4 String CloneVsGl::name

Definition at line 38 of file Data.h.

4.5.3.5 String CloneVsGl::seq

Definition at line 39 of file Data.h.

The documentation for this struct was generated from the following file:

- Data.h

4.6 FileWriter Class Reference

Class for wrapping all functions for creating output file.

`#include <FileWriter.h>`

Public Member Functions

- FileWriter (CloneCalculations *cloneCalcs, const String saveFilePath, bool writePmHierarchy, bool writeClonalMetadata, bool writeClonalGroups)

  Creates an instance of FileWriter.

- ~FileWriter ()
- void run ()
- const String & getSaveFilePath () const

Private Member Functions

- void writeHeader ()
- void writeClonalMetadata ()
- void writeMutationOnPosition ()
- void writeIntermediateSequencesUnique ()
- void writeIntermediateSequencesCalculated ()
- void writeFile ()

Private Attributes

- CloneCalculations * m_cloneCalculations
- String m_saveFilePath
- String m_stream
- bool m_writeClonalMetadata
- bool m_writePmHierarchy
- bool m_writeClonalGroups

4.6.1 Detailed Description

Class for wrapping all functions for creating output file.

Author

Peter Vasil, Technical University, Berlin
Martin Wartenberg, Technical University, Munich

Definition at line 30 of file FileWriter.h.

4.6.2 Constructor & Destructor Documentation

4.6.2.1 FileWriter::FileWriter (CloneCalculations * *cloneCalcs*, const String *saveFilePath*, bool *writePmHierarchy*, bool *writeClonalMetadata*, bool *writeClonalGroups*)

Creates an instance of FileWriter.

Parameters

cloneCalcs Pointer to CloneCalculation instance.

saveFilePath save location path String.

writePmHierarchy true for outputting pm hierarchy.

writeClonalMetadata true for outputting clonal metadata.

writeClonalGroups true for outputting clonal groups

Definition at line 21 of file FileWriter.cpp.

4.6.2.2 FileWriter::~FileWriter ()

Definition at line 37 of file FileWriter.cpp.

4.6.3 Member Function Documentation

4.6.3.1 const String& FileWriter::getSaveFilePath () const `[inline]`

Definition at line 50 of file FileWriter.h.

4.6.3.2 void FileWriter::run ()

Definition at line 41 of file FileWriter.cpp.

4.6.3.3 void FileWriter::writeClonalMetadata () `[private]`

Definition at line 94 of file FileWriter.cpp.

4.6.3.4 void FileWriter::writeFile () `[private]`

Definition at line 525 of file FileWriter.cpp.

4.6.3.5 void FileWriter::writeHeader () `[private]`

Definition at line 69 of file FileWriter.cpp.

4.6.3.6 void FileWriter::writeIntermediateSequencesCalculated () `[private]`

Definition at line 458 of file FileWriter.cpp.

4.6.3.7 void FileWriter::writeIntermediateSequencesUnique () [private]

Definition at line 404 of file FileWriter.cpp.

4.6.3.8 void FileWriter::writeMutationOnPosition () [private]

Definition at line 164 of file FileWriter.cpp.

4.6.4 Member Data Documentation

4.6.4.1 CloneCalculations* FileWriter::m_cloneCalculations [private]

Definition at line 61 of file FileWriter.h.

4.6.4.2 String FileWriter::m_saveFilePath [private]

Definition at line 62 of file FileWriter.h.

4.6.4.3 String FileWriter::m_stream [private]

Definition at line 63 of file FileWriter.h.

4.6.4.4 bool FileWriter::m_writeClonalGroups [private]

Definition at line 66 of file FileWriter.h.

4.6.4.5 bool FileWriter::m_writeClonalMetadata [private]

Definition at line 64 of file FileWriter.h.

4.6.4.6 bool FileWriter::m_writePmHierarchy [private]

Definition at line 65 of file FileWriter.h.

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

- FileWriter.h
- FileWriter.cpp

4.7 IntermediateClones Struct Reference

HPCs.

`#include <Data.h>`

Public Member Functions

- bool operator() (IntermediateClones lhs, IntermediateClones rhs)

Public Attributes

- int pos
- int numMut
- String intermediateClone
- std::vector< CloneVsGl > clones
- double pm
- int z
- bool shouldDisplay

4.7.1 Detailed Description

HPCs.

Definition at line 66 of file Data.h.

4.7.2 Member Function Documentation

4.7.2.1 bool IntermediateClones::operator() (IntermediateClones *lhs*, IntermediateClones *rhs*) [inline]

Definition at line 74 of file Data.h.

4.7.3 Member Data Documentation

4.7.3.1 std::vector<CloneVsGl> IntermediateClones::clones

Definition at line 70 of file Data.h.

4.7.3.2 String IntermediateClones::intermediateClone

Definition at line 69 of file Data.h.

4.7.3.3 int IntermediateClones::numMut

Definition at line 68 of file Data.h.

4.7.3.4 double IntermediateClones::pm

Definition at line 71 of file Data.h.

4.7.3.5 int IntermediateClones::pos

Definition at line 67 of file Data.h.

4.7.3.6 bool IntermediateClones::shouldDisplay

Definition at line 73 of file Data.h.

4.7.3.7 int IntermediateClones::z

Definition at line 72 of file Data.h.

The documentation for this struct was generated from the following file:

- Data.h

4.8 MainAppWindow Class Reference

Main application window.

`#include <MainAppWindow.h>`

Public Member Functions

- MainAppWindow (ApplicationCommandManager *commandManager)
- ~MainAppWindow ()
- void closeButtonPressed ()

  called when the close button is pressed or esc is pushed

4.8.1 Detailed Description

Main application window.

Author

Peter Vasil, Technical University, Berlin
Martin Wartenberg, Technical University, Munich

Definition at line 29 of file MainAppWindow.h.

4.8.2 Constructor & Destructor Documentation

4.8.2.1 MainAppWindow::MainAppWindow (ApplicationCommandManager * *commandManager*)

Definition at line 24 of file MainAppWindow.cpp.

4.8.2.2 MainAppWindow::~MainAppWindow ()

Definition at line 48 of file MainAppWindow.cpp.

4.8.3 Member Function Documentation

4.8.3.1 void MainAppWindow::closeButtonPressed ()

called when the close button is pressed or esc is pushed

Definition at line 55 of file MainAppWindow.cpp.

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

- MainAppWindow.h
- MainAppWindow.cpp

4.9 MainComponent Class Reference

Main GUI component.

`#include <MainComponent.h>`

Public Member Functions

- MainComponent (DocumentWindow *mainWindow, ApplicationCommandManager *commandManager)
- ~MainComponent ()
- juce_UseDebuggingNewOperator void resized ()
- void paint (Graphics &g)
- void buttonClicked (Button *button)
- bool getGlFile ()
- bool getCloneFiles ()
- bool checkIfGlHasSeqLength ()
- bool checkIfClonesHaveSeqLength ()
- bool saveFile ()
- void selectedFileChanged (const File &newSelectedFile)
- void textEditorTextChanged (TextEditor &editor)
- void textEditorReturnKeyPressed (TextEditor &editor)
- void textEditorEscapeKeyPressed (TextEditor &editor)
- void textEditorFocusLost (TextEditor &editor)
- const StringArray getMenuBarNames ()
- const PopupMenu getMenuForIndex (int menuIndex, const String &menuName)
- void menuItemSelected (int menuItemID, int topLevelMenuIndex)
- ApplicationCommandTarget * getNextCommandTarget ()
- void getAllCommands (Array< CommandID > &commands)
- void getCommandInfo (CommandID commandID, ApplicationCommandInfo &result)
- bool perform (const InvocationInfo &info)

Private Types

- enum CommandIDs {
  pmhierarchy = 0x2000, clonalmetadata = 0x2001, clonalgroups = 0x2002, usage = 0x2003,
  basicfunction = 0x2004, about = 0x2005, web = 0x2006 }

Private Member Functions

- MainComponent (const MainComponent &)

  Prevent copy constructor being generated.

- const MainComponent & operator= (const MainComponent &)

  Prevent operator= being generated.

Private Attributes

- DocumentWindow * m_mainWindow
- ApplicationCommandManager * m_commandManager
- TooltipWindow tooltipWindow
- TextButton * m_btGl
- TextButton * m_btKlone
- TextButton * m_btCloneVsGl
- TextButton * m_btRestart
- TextButton * m_btSaveFile
- String m_statusText
- int m_seqLength
- TextEditor * m_txSeqLen
- CloneCalculations * m_cloneClaculations
- PropertiesFile * m_props
- String m_lastGlFolderPath
- String m_lastClonesFolderPath
- String m_lastSaveFolderPath
- bool m_writePmHierarchy
- bool m_writeClonalMetadata
- bool m_writeClonalGroups

4.9.1 Detailed Description

Main GUI component.

Author

Peter Vasil, Technical University, Berlin
Martin Wartenberg, Technical University, Munich

Definition at line 29 of file MainComponent.h.

4.9.2 Member Enumeration Documentation

4.9.2.1 enum MainComponent::CommandIDs [private]

Enumerator:

pmhierarchy

clonalmetadata

clonalgroups

usage

basicfunction

about Show about dialog.

web

Definition at line 35 of file MainComponent.h.

4.9.3 Constructor & Destructor Documentation

4.9.3.1 MainComponent::MainComponent (DocumentWindow * *mainWindow*, ApplicationCommandManager * *commandManager*)

Definition at line 24 of file MainComponent.cpp.

4.9.3.2 MainComponent::~MainComponent ()

Definition at line 98 of file MainComponent.cpp.

4.9.3.3 MainComponent::MainComponent (const MainComponent &) [private]

Prevent copy constructor being generated.

4.9.4 Member Function Documentation

4.9.4.1 void MainComponent::buttonClicked (Button * *button*)

Definition at line 132 of file MainComponent.cpp.

4.9.4.2 bool MainComponent::checkIfClonesHaveSeqLength ()

Definition at line 255 of file MainComponent.cpp.

4.9.4.3 bool MainComponent::checkIfGlHasSeqLength ()

Definition at line 245 of file MainComponent.cpp.

4.9.4.4 void MainComponent::getAllCommands (Array< CommandID > & *commands*) [inline]

Definition at line 122 of file MainComponent.h.

4.9.4.5 bool MainComponent::getCloneFiles ()

Definition at line 218 of file MainComponent.cpp.

4.9.4.6 void MainComponent::getCommandInfo (CommandID *commandID*, ApplicationCommandInfo & *result*) [inline]

Definition at line 136 of file MainComponent.h.

4.9.4.7 bool MainComponent::getGlFile ()

Definition at line 190 of file MainComponent.cpp.

4.9.4.8 const StringArray MainComponent::getMenuBarNames () [inline]

Definition at line 83 of file MainComponent.h.

4.9.4.9 const PopupMenu MainComponent::getMenuForIndex (int *menuIndex*, const String & *menuName*) [inline]

Definition at line 91 of file MainComponent.h.

4.9.4.10 ApplicationCommandTarget* MainComponent::getNextCommandTarget () [inline]

Definition at line 117 of file MainComponent.h.

4.9.4.11 void MainComponent::menuItemSelected (int *menuItemID*, int *topLevelMenuIndex*) [inline]

Definition at line 113 of file MainComponent.h.

4.9.4.12 const MainComponent& MainComponent::operator= (const MainComponent &) [private]

Prevent operator= being generated.

4.9.4.13 void MainComponent::paint (Graphics & *g*)

Definition at line 126 of file MainComponent.cpp.

4.9.4.14 bool MainComponent::perform (const InvocationInfo & *info*) [inline]

Definition at line 177 of file MainComponent.h.

4.9.4.15 void MainComponent::resized ()

Definition at line 117 of file MainComponent.cpp.

4.9.4.16 bool MainComponent::saveFile ()

Definition at line 273 of file MainComponent.cpp.

4.9.4.17 void MainComponent::selectedFileChanged (const File & *newSelectedFile*)

Definition at line 318 of file MainComponent.cpp.

4.9.4.18 void MainComponent::textEditorEscapeKeyPressed (TextEditor & *editor*)

Called when the user presses the escape key.

Definition at line 328 of file MainComponent.cpp.

4.9.4.19 void MainComponent::textEditorFocusLost (TextEditor & *editor*)

Called when the text editor loses focus.

Definition at line 333 of file MainComponent.cpp.

4.9.4.20 void MainComponent::textEditorReturnKeyPressed (TextEditor & *editor*)

Called when the user presses the return key.

Definition at line 323 of file MainComponent.cpp.

4.9.4.21 void MainComponent::textEditorTextChanged (TextEditor & *editor*)

Called when the user changes the text in some way.

Definition at line 310 of file MainComponent.cpp.

4.9.5 Member Data Documentation

4.9.5.1 TextButton* MainComponent::m_btCloneVsGl [private]

Definition at line 306 of file MainComponent.h.

4.9.5.2 TextButton* MainComponent::m_btGl [private]

Definition at line 304 of file MainComponent.h.

4.9.5.3 TextButton* MainComponent::m_btKlone [private]

Definition at line 305 of file MainComponent.h.

4.9.5.4 TextButton* MainComponent::m_btRestart [private]

Definition at line 307 of file MainComponent.h.

4.9.5.5 TextButton* MainComponent::m_btSaveFile [private]

Definition at line 308 of file MainComponent.h.

4.9.5.6 CloneCalculations* MainComponent::m_cloneClaculations [private]

Definition at line 315 of file MainComponent.h.

4.9.5.7 ApplicationCommandManager* MainComponent::m_commandManager [private]

Definition at line 295 of file MainComponent.h.

4.9.5.8 String MainComponent::m_lastClonesFolderPath [private]

Definition at line 319 of file MainComponent.h.

4.9.5.9 String MainComponent::m_lastGlFolderPath [private]

Definition at line 318 of file MainComponent.h.

4.9.5.10 String MainComponent::m_lastSaveFolderPath [private]

Definition at line 320 of file MainComponent.h.

4.9.5.11 DocumentWindow* MainComponent::m_mainWindow [private]

Definition at line 294 of file MainComponent.h.

4.9.5.12 PropertiesFile* MainComponent::m_props [private]

Definition at line 317 of file MainComponent.h.

4.9.5.13 int MainComponent::m_seqLength [private]

Definition at line 311 of file MainComponent.h.

4.9.5.14 String MainComponent::m_statusText [private]

Definition at line 310 of file MainComponent.h.

4.9.5.15 TextEditor* MainComponent::m_txSeqLen [private]

Definition at line 313 of file MainComponent.h.

4.9.5.16 bool MainComponent::m_writeClonalGroups [private]

Definition at line 324 of file MainComponent.h.

4.9.5.17 bool MainComponent::m_writeClonalMetadata [private]

Definition at line 323 of file MainComponent.h.

4.9.5.18 bool MainComponent::m_writePmHierarchy [private]

Definition at line 322 of file MainComponent.h.

4.9.5.19 TooltipWindow MainComponent::tooltipWindow [private]

Definition at line 296 of file MainComponent.h.

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

- MainComponent.h
- MainComponent.cpp

4.10 PVLogger Class Reference

Logger class for easy debugging.

`#include <PVLogger.h>`

Static Public Member Functions

- static void outputError (const tchar *logType, const char *function, const String &message)
- static void logMessage (const tchar *logType, const char *function, const String &name, const String &message)
- static void logMessage (const tchar *logType, const char *function, const String &name, int message)
- static void logMessage (const tchar *logType, const char *function, const String &name, float message)
- static void logMessage (const tchar *logType, const char *function, const String &name, double message)
- static void logMessage (const tchar *logType, const char *function, const String &name, bool message)
- static void logMessage (const tchar *logType, const char *function)

Private Member Functions

- PVLogger ()
- ~PVLogger ()

Static Private Member Functions

- static int64 currentTimeMillisHiRes ()
- static void getCurrentTime (int64 seconds, struct tm &currTime)
- static const String getCurrenTimeString ()

4.10.1 Detailed Description

Logger class for easy debugging.

Definition at line 57 of file PVLogger.h.

4.10.2 Constructor & Destructor Documentation

4.10.2.1 PVLogger::PVLogger () [inline, private]

Definition at line 59 of file PVLogger.h.

4.10.2.2 PVLogger::~PVLogger () [inline, private]

Definition at line 62 of file PVLogger.h.

4.10.3 Member Function Documentation

4.10.3.1 static int64 PVLogger::currentTimeMillisHiRes () [inline, static, private]

Definition at line 179 of file PVLogger.h.

4.10.3.2 static const String PVLogger::getCurrenTimeString () [inline, static, private]

Definition at line 226 of file PVLogger.h.

4.10.3.3 static void PVLogger::getCurrentTime (int64 *seconds*, struct tm & *currTime*) [inline, static, private]

Definition at line 216 of file PVLogger.h.

4.10.3.4 static void PVLogger::logMessage (const tchar * *logType*, const char * *function*) [inline, static]

Definition at line 161 of file PVLogger.h.

4.10.3.5 static void PVLogger::logMessage (const tchar * *logType*, const char * *function*, const String & *name*, bool *message*) [inline, static]

Definition at line 145 of file PVLogger.h.

4.10.3.6 static void PVLogger::logMessage (const tchar * *logType*, const char * *function*, const String & *name*, double *message*) [inline, static]

Definition at line 129 of file PVLogger.h.

4.10.3.7 static void PVLogger::logMessage (const tchar * *logType*, const char * *function*, const String & *name*, float *message*) [inline, static]

Definition at line 113 of file PVLogger.h.

4.10.3.8 static void PVLogger::logMessage (const tchar * *logType*, const char * *function*, const String & *name*, int *message*) [inline, static]

Definition at line 97 of file PVLogger.h.

4.10.3.9 static void PVLogger::logMessage (const tchar * *logType*, const char * *function*, const String & *name*, const String & *message*) [inline, static]

Definition at line 81 of file PVLogger.h.

4.10.3.10 static void PVLogger::outputError (const tchar * *logType*, const char * *function*, const String & *message*) [inline, static]

Definition at line 67 of file PVLogger.h.

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

- PVLogger.h

Chapter 5

File Documentation

5.1 CloneCalculations.cpp File Reference

```
#include "CloneCalculations.h"
#include "AppConfig.h"
#include <cstdlib>
#include <cstdarg>
#include <climits>
#include <limits>
#include <cmath>
#include <cwchar>
#include <stdexcept>
#include <typeinfo>
#include <cstring>
#include <cstdio>
#include <iostream>
#include <sys/time.h>
#include <sys/types.h>
#include <stdlib.h>
#include <sys/timeb.h>
#include <time.h>
#include <vector>
#include <algorithm>
```

Functions

- bool compareCloneRefs (std::vector< CloneVsGl * > lhs, std::vector< CloneVsGl * > rhs)
- bool compareIntermediates (IntermediateClones lhs, IntermediateClones rhs)

5.1.1 Function Documentation

5.1.1.1 bool compareCloneRefs (std::vector< CloneVsGl * > *lhs*, std::vector< CloneVsGl * > *rhs*)

Definition at line 37 of file CloneCalculations.cpp.

5.1.1.2 bool compareIntermediates (IntermediateClones *lhs*, IntermediateClones *rhs*)

Definition at line 42 of file CloneCalculations.cpp.

5.2 CloneCalculations.cpp

```
00001 /*=========================================================================
00002  * Copyright (C) 2010 Peter Vasil
00003  *
00004  * This program is free software; you can redistribute it and/or modify
00005  * it under the terms of the GNU General Public License as published by
00006  * the Free Software Foundation; either version 3 of the License, or
00007  * (at your option) any later version.
00008  *
00009  * This program is distributed in the hope that it will be useful,
00010  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00012  * GNU General Public License for more details.
00013  *
00014  * You should have received a copy of the GNU General Public License
00015  * along with this program; if not, see <http://www.gnu.org/licenses/>.
00016  *
00017  =========================================================================*/
00018
00019 #include "CloneCalculations.h"
00020 #include <algorithm>
00021
00022
00023 CloneCalculations::CloneCalculations()
00024 :
00025 m_gl(String::empty),
00026 m_seqLength(195),
00027 m_statusText(String::empty)
00028 {
00029     //ctor
00030 }
00031
00032 CloneCalculations::~CloneCalculations()
00033 {
00034     //dtor
00035 }
00036
00037 bool compareCloneRefs(std::vector<CloneVsGl*> lhs, std::vector<CloneVsGl*> rhs)
00038 {
00039     return lhs.size() < rhs.size();
00040 }
00041
00042 bool compareIntermediates(IntermediateClones lhs, IntermediateClones rhs)
00043 {
00044     return lhs.numMut >= rhs.numMut;
00045 }
00046
00047 void CloneCalculations::checkCloneVsGl()
00048 {
00049     for (unsigned int i = 0; i < m_cloneVec.size(); ++i)
00050     {
00051         CloneVsGl clgl;
00052         clgl.name = m_cloneVec[i].name;
00053         clgl.seq = m_cloneVec[i].seq;
00054         clgl.mutPos.clear();
00055         String tmpDif = T("");
00056         for (int j = 0; j < m_seqLength; ++j)
00057         {
00058             if (m_cloneVec[i].seq[j] == m_gl[j])
00059             {
00060                 tmpDif += T("-");
00061             }
00062             else
00063             {
00064                 tmpDif += m_cloneVec[i].seq[j];
00065                 clgl.mutPos.push_back(j);
```

```
00066             }
00067         }
00068         clgl.diff = tmpDif;
00069         clgl.mutNum = clgl.mutPos.size();
00070         m_cloneVsGl.push_back(clgl);
00071     }
00072
00073     m_statusText = T("Compared clones against gl!");
00074 }
00075
00076 void CloneCalculations::sortClones()
00077 {
00078     // Sorting of clones.
00079     CloneVsGl tmpClgl;
00080     m_cloneVsGlSorted = m_cloneVsGl;
00081     std::sort(m_cloneVsGlSorted.begin(), m_cloneVsGlSorted.end(), tmpClgl);
00082     for (unsigned int i = 0; i < m_cloneVsGlSorted.size(); ++i)
00083     {
00084         DBG_VAL(m_cloneVsGlSorted[i].name);
00085         DBG_VAL(m_cloneVsGlSorted[i].mutNum);
00086     }
00087 }
00088
00089 void CloneCalculations::countMutationOnPosition()
00090 {
00091     for(int i = 0; i < m_seqLength; ++i)
00092     {
00093         m_diffTimesA.push_back(0);
00094         m_diffTimesC.push_back(0);
00095         m_diffTimesT.push_back(0);
00096         m_diffTimesG.push_back(0);
00097         m_diffTimesARelative.push_back(0);
00098         m_diffTimesCRelative.push_back(0);
00099         m_diffTimesTRelative.push_back(0);
00100         m_diffTimesGRelative.push_back(0);
00101     }
00102
00103     // count number of mutation for a position in sequence.
00104     m_cloneRefs.clear();
00105     for(int i = 0; i < m_seqLength; ++i)
00106     {
00107         std::vector<CloneVsGl> tmpRefs;
00108         String sign = T("-");
00109         for(unsigned int j = 0; j < m_cloneVsGl.size(); ++j)
00110         {
00111             if (m_cloneVsGl[j].diff[i] != '-')
00112             {
00113                 tmpRefs.push_back(m_cloneVsGl[j]);
00114                 switch(m_cloneVsGl[j].diff[i])
00115                 {
00116                 case 'A':
00117                     m_diffTimesA[i] += 1;
00118                     break;
00119                 case 'C':
00120                     m_diffTimesC[i] += 1;
00121                     break;
00122                 case 'T':
00123                     m_diffTimesT[i] += 1;
00124                     break;
00125                 case 'G':
00126                     m_diffTimesG[i] += 1;
00127                     break;
00128                 }
00129             }
00130         }
00131         m_cloneRefs.push_back(tmpRefs);
00132     }
```

```
00133     for(unsigned int i = 0; i < m_cloneRefs.size(); ++i)
00134     {
00135         for(unsigned int j = 0; j < m_cloneRefs[i].size(); ++j)
00136         {
00137             DBG_VAL(m_cloneRefs[i][j].name);
00138         }
00139         DBG_VAL((int)i);
00140     }
00141 
00142     for(unsigned int i = 0; i < m_cloneRefs.size(); ++i)
00143     {
00144         CloneRefsAndPosition crap;
00145         crap.pos = i;
00146         crap.clones = m_cloneRefs[i];
00147         m_cloneRefsSorted.push_back(crap);
00148     }
00149     CloneRefsAndPosition crapcompare;
00150     //m_cloneRefsSorted = m_cloneRefs;
00151     std::sort( m_cloneRefsSorted.begin(), m_cloneRefsSorted.end(), crapcompare );

00152 
00153     for(unsigned int i = 0; i < m_cloneRefsSorted.size(); ++i)
00154     {
00155         DBG_VAL(m_cloneRefsSorted[i].pos);
00156         for(unsigned int j = 0; j < m_cloneRefsSorted[i].clones.size(); ++j)
00157         {
00158             DBG_VAL(m_cloneRefsSorted[i].clones[j].name);
00159         }
00160 
00161     }
00162     // relative number of mutation
00163     for(int i = 0; i < m_seqLength; ++i)
00164     {
00165         m_diffTimesARelative[i] = m_diffTimesA[i]/(double)m_cloneVec.size();
00166         m_diffTimesCRelative[i] = m_diffTimesC[i]/(double)m_cloneVec.size();
00167         m_diffTimesTRelative[i] = m_diffTimesT[i]/(double)m_cloneVec.size();
00168         m_diffTimesGRelative[i] = m_diffTimesG[i]/(double)m_cloneVec.size();
00169 
00170 
00171     }
00172 
00173 }
00174 
00175 void CloneCalculations::calculateIntermediateClones()
00176 {
00177     for(unsigned int i = 0; i < m_cloneRefsSorted.size(); ++i)
00178     {
00179         String intermediateStr = String::empty;
00180         int mutNumber = 0;
00181         unsigned int numClones = m_cloneRefsSorted[i].clones.size();
00182         if(numClones != 0)
00183         {
00184             for(int j = 0; j < m_seqLength ; ++j)
00185             {
00186                 bool res = false;
00187                 wchar_t c = m_cloneRefsSorted[i].clones[0].diff[j];
00188                 //DBG_VAL(c);
00189                 if( c != '-')
00190                 {
00191                                         for(unsigned int k = 0; k < numClones; ++
      k)
00192                                         {
00193                                                 if (m_cloneRefsSorted[i].clones[k
      ].diff[j] == c)
00194                                                 {
00195                                                         res = true;
00196                                                 }
```

```
00197                                                   else
00198                                                   {
00199                                                           res = false;
00200                                                           break;
00201                                                   }
00202                                           }
00203                   }
00204                   if (res)
00205                   {
00206                       intermediateStr += c;
00207                       ++mutNumber;
00208                   }
00209                   else
00210                   {
00211                       intermediateStr += '-';
00212                   }
00213               }
00214           }
00215           IntermediateClones intCls;
00216           intCls.pos = m_cloneRefsSorted[i].pos;
00217           intCls.numMut = mutNumber;
00218           intCls.clones = m_cloneRefsSorted[i].clones;
00219           intCls.intermediateClone = intermediateStr;
00220           if(intCls.clones.size() != 0)
00221           {
00222               m_intermediatClones.push_back(intCls);
00223               m_intermediatClonesSorted.push_back(intCls);
00224           }
00225       }
00226       for(unsigned int i = 0; i < m_intermediatClones.size(); ++i)
00227       {
00228           DBG_VAL(m_intermediatClones[i].pos);
00229           DBG_VAL(m_intermediatClones[i].intermediateClone);
00230           for(unsigned int j = 0; j < m_intermediatClones[i].clones.size(); ++j)
00231           {
00232               DBG_VAL(m_intermediatClones[i].clones[j].name);
00233           }
00234       }
00235
00236
00237       IntermediateClones tmpIntCl;
00238       std::sort( m_intermediatClonesSorted.begin(), m_intermediatClonesSorted.end()
      , tmpIntCl );
00239
00240       for(unsigned int i = 0; i < m_intermediatClonesSorted.size(); ++i)
00241       {
00242           DBG_VAL(m_intermediatClonesSorted[i].pos);
00243           DBG_VAL(m_intermediatClonesSorted[i].intermediateClone);
00244           DBG_VAL(m_intermediatClonesSorted[i].numMut);
00245           DBG_VAL((int)m_intermediatClonesSorted[i].clones.size());
00246 //          for(unsigned int j = 0; j < m_intermediatClonesSorted[i].clones.size();
       ++j)
00247 //          {
00248 //              DBG_VAL(m_intermediatClonesSorted[i].clones[j]->name);
00249 //          }
00250       }
00251 }
00252
00253 void CloneCalculations::calculateIntermediateClonesUnique()
00254 {
00255
00256           for (unsigned int i = 0; i < m_intermediatClones.size(); ++i)
00257           {
00258                   std::vector<String> newIntermediates;
00259                   std::vector<String> tmpIntermediate;
00260                   for (unsigned int tmpI = 0; tmpI < 4; ++tmpI)
00261                   {
```

```
00262                         tmpIntermediate.push_back(m_intermediatClones[i].intermed
      iateClone);
00263                 }
00264
00265         for (unsigned int j = 0; j < m_intermediatClones[i].clones.size(); ++j)
00266         {
00267             switch(m_intermediatClones[i].clones[j].diff[m_intermediatClones[i].p
      os])
00268             {
00269             case 'A':
00270                 tmpIntermediate[0][m_intermediatClones[i].pos] = 'A';
00271                 break;
00272             case 'C':
00273                 tmpIntermediate[1][m_intermediatClones[i].pos] = 'C';
00274                 break;
00275             case 'T':
00276                 tmpIntermediate[2][m_intermediatClones[i].pos] = 'T';
00277                 break;
00278             case 'G':
00279                 tmpIntermediate[3][m_intermediatClones[i].pos] = 'G';
00280                 break;
00281             }
00282         }
00283         bool firstIntermediate = true;
00284                 for (unsigned int tmpII = 0; tmpII < tmpIntermediate.size(); ++tm
      pII)
00285                 {
00286                         if (tmpIntermediate[tmpII].compare(m_intermediatClones[i]
      .intermediateClone) != 0)
00287                         {
00288                                 newIntermediates.push_back(tmpIntermediate[tmpII]
      );
00289                         }
00290                 }
00291         if (newIntermediates.size() == 0 )
00292         {
00293             if (m_intermediatClones[i].intermediateClone.length() != 0)
00294                 newIntermediates.push_back(m_intermediatClones[i].intermediateClo
      ne);
00295         }
00296                 for (unsigned int j = 0; j < newIntermediates.size(); ++j)
00297                 {
00298
00299                         IntermediateClones intCls;
00300
00301                         std::vector<CloneVsGl> tmpClones;
00302                         for (unsigned int numCl = 0; numCl < m_intermediatClones[
      i].clones.size(); ++numCl)
00303                         {
00304                                 bool result = false;
00305                                 for (int k = 0; k < m_seqLength; ++k)
00306                                 {
00307                                         if(newIntermediates[j][k] != '-' && newIn
      termediates[j][k] == m_intermediatClones[i].clones[numCl].diff[k])
00308                                         {
00309                                                 result = true;
00310                                         }
00311                                         else if (newIntermediates[j][k] != '-' &&
       newIntermediates[j][k] != m_intermediatClones[i].clones[numCl].diff[k])
00312                                         {
00313                                                 result = false;
00314                                                 break;
00315                                         }
00316                                 }
00317                                 if (result)
00318                                 {
00319                                         tmpClones.push_back(m_intermediatClones[i
```

```
    ].clones[numCl]);
00320                                   }
00321                               }
00322             //-------------------------------------------------------------------
    --------
00323                               for (int k = 0; k < m_seqLength; ++k)
00324                               {
00325                   if(tmpClones.size() > 0)
00326                   {
00327                       bool res = false;
00328                       wchar_t c = tmpClones[0].diff[k];
00329 
00330                       if(c != '-')
00331                       {
00332                           for(unsigned int kk = 0; kk < tmpClones.size(); ++kk)
00333                           {
00334                               if(tmpClones[kk].diff[k] == c)
00335                               {
00336                                   res = true;
00337                               }
00338                               else
00339                               {
00340                                   res = false;
00341                                   break;
00342                               }
00343                           }
00344                           if(res)
00345                           {
00346                               newIntermediates[j][k] = c;
00347                           }
00348                       }
00349 
00350                   }
00351                               }
00352 
00353                               int numMut = 0;
00354                               //LOG_VAL(newIntermediates[j].length());
00355                               for (int k = 0; k < m_seqLength; ++k)
00356                               {
00357                                       if (newIntermediates[j][k] != '-')
00358                                       {
00359                                               ++numMut;
00360                                       }
00361                               }
00362 
00363                               intCls.pos = m_intermediatClones[i].pos;
00364                               intCls.numMut = numMut;
00365                               intCls.clones = tmpClones;
00366                               intCls.intermediateClone = newIntermediates[j];
00367                               intCls.pm = 0.0;
00368                               intCls.shouldDisplay = true;
00369                               if(intCls.clones.size() >= 2)
00370                               {
00371                   m_intermediateClonesUnique.push_back(intCls);
00372                               }
00373                   }
00374           }
00375 
00376 }
00377 
00378 void CloneCalculations::calculatePm()
00379 {
00380     for(unsigned int i = 0; i < m_intermediateClonesUnique.size(); ++i)
00381     {
00382         int t = (int)m_cloneVec.size();
00383         int r = (int)m_intermediateClonesUnique[i].clones.size();
00384         int mutNum = m_intermediateClonesUnique[i].numMut;
```

```
00385         int z = 0;
00386         for(unsigned int j = 0; j < m_intermediateClonesUnique.size(); ++j)
00387         {
00388             if(m_intermediateClonesUnique[i].intermediateClone.compare(
      m_intermediateClonesUnique[j].intermediateClone) == 0)
00389             {
00390                 ++z;
00391                 if ( i != j && m_intermediateClonesUnique[i].shouldDisplay)
00392                 {
00393                     m_intermediateClonesUnique[j].shouldDisplay = false;
00394                 }
00395             }
00396         }
00397         double pk = 0.0;
00398         for(int j = 0; j < m_seqLength; ++j)
00399         {
00400             wchar_t c = m_intermediateClonesUnique[i].intermediateClone[j];
00401             switch ( c )
00402             {
00403             case 'A':
00404                 if(pk == 0.0)
00405                     pk = m_diffTimesARelative[j];
00406                 else
00407                     pk *= m_diffTimesARelative[j];
00408                 break;
00409             case 'C':
00410                 if(pk == 0.0)
00411                     pk = m_diffTimesCRelative[j];
00412                 else
00413                     pk *= m_diffTimesCRelative[j];
00414                 break;
00415             case 'T':
00416                 if(pk == 0.0)
00417                     pk = m_diffTimesTRelative[j];
00418                 else
00419                     pk *= m_diffTimesTRelative[j];
00420                 break;
00421             case 'G':
00422                 if(pk == 0.0)
00423                     pk = m_diffTimesGRelative[j];
00424                 else
00425                     pk *= m_diffTimesGRelative[j];
00426                 break;
00427             default:
00428                 break;
00429             }
00430
00431         }
00432         double pm = 0.0;
00433         pm = pow(pk,(1.0 / ((t*1.0 - r*1.0 + 1.0) * (double)mutNum * (double)z)))
      ;
00434
00435         m_intermediateClonesUnique[i].pm = pm;
00436         m_intermediateClonesUnique[i].z = z;
00437     }
00438
00439     m_intermediateClonesUniqueSorted = m_intermediateClonesUnique;
00440     IntermediateClones compareIntermediate;
00441     std::sort(m_intermediateClonesUniqueSorted.begin(),
      m_intermediateClonesUniqueSorted.end(), compareIntermediate);
00442
00443 }
00444
00445 void CloneCalculations::calculate()
00446 {
00447     this->checkCloneVsGl();
00448
```

```
00449     this->sortClones();
00450
00451     this->countMutationOnPosition();
00452
00453     this->calculateIntermediateClones();
00454
00455         this->calculateIntermediateClonesUnique();
00456
00457         this->calculatePm();
00458 }
00459
00460 void CloneCalculations::clearValues()
00461 {
00462     m_gl = String::empty;
00463     m_cloneVec.clear();
00464     m_cloneVsGl.clear();
00465     m_cloneVsGlSorted.clear();
00466
00467     for(unsigned int i = 0; i < m_cloneRefs.size(); ++i)
00468     {
00469         m_cloneRefs[i].clear();
00470     }
00471     m_cloneRefs.clear();
00472     m_cloneRefsSorted.clear();
00473     m_intermediatClones.clear();
00474     m_intermediatClonesSorted.clear();
00475         m_intermediateClonesUnique.clear();
00476
00477     m_seqLength = 0;
00478     m_diffTimesA.clear();
00479     m_diffTimesC.clear();
00480     m_diffTimesT.clear();
00481     m_diffTimesG.clear();
00482     m_statusText = String::empty;
00483
00484 }
```

5.3 CloneCalculations.h File Reference

```
#include "FLER_Headers.h"
#include "Data.h"
```

Classes

- class CloneCalculations

  Class for clone calculations.

5.4 CloneCalculations.h

```
00001 /*=============================================================================
00002  * Copyright (C) 2010 Peter Vasil
00003  *
00004  * This program is free software; you can redistribute it and/or modify
00005  * it under the terms of the GNU General Public License as published by
00006  * the Free Software Foundation; either version 3 of the License, or
00007  * (at your option) any later version.
00008  *
00009  * This program is distributed in the hope that it will be useful,
00010  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00012  * GNU General Public License for more details.
00013  *
00014  * You should have received a copy of the GNU General Public License
00015  * along with this program; if not, see <http://www.gnu.org/licenses/>.
00016  *
00017  ============================================================================*/
00018
00019 #ifndef _CLONECALCULATIONS_H_
00020 #define _CLONECALCULATIONS_H_
00021
00022 #include "FLER_Headers.h"
00023 #include "Data.h"
00024
00030 class CloneCalculations
00031 {
00032 public:
00033     CloneCalculations();
00034     ~CloneCalculations();
00035
00036     //-------------------------------------------------------------
00041     void setSeqLength(const int seqLength) { m_seqLength = seqLength; }
00046     void setGlFile(const String& glFilePath) { m_gl = File(glFilePath).loadFileAs
      String(); }
00051     void setClones(const std::vector<Clone>& clones) { m_cloneVec = clones; }
00052     //-------------------------------------------------------------
00053     const int getSeqLength() const { return m_seqLength; }
00054     const std::vector<Clone>& getClones() const { return m_cloneVec; }
00055     const String& getGlString() const { return m_gl; }
00056     const std::vector<CloneVsGl>& getCloneVsGl() const { return m_cloneVsGl; }
00057     const std::vector<CloneVsGl>& getCloneVsGlSorted() const { return
      m_cloneVsGlSorted; }
00058     const std::vector<std::vector<CloneVsGl> >& getCloneRefs() const { return
      m_cloneRefs; }
00059     const std::vector<CloneRefsAndPosition>& getCloneRefsSorted() const { return
      m_cloneRefsSorted; }
00060     const std::vector<IntermediateClones>& getIntermediateClones() const { return
       m_intermediatClones; }
00061     const std::vector<IntermediateClones>& getIntermediateClonesSorted() const {
      return m_intermediatClonesSorted; }
00062         const std::vector<IntermediateClones>& getIntermediateClonesUnique() cons
      t { return m_intermediateClonesUnique; }
00063     const std::vector<IntermediateClones>& getIntermediateClonesUniqueSorted() co
      nst { return m_intermediateClonesUniqueSorted; }
00064     const std::vector<int>& getDiffTimesA() const { return m_diffTimesA; }
00065     const std::vector<int>& getDiffTimesC() const { return m_diffTimesC; }
00066     const std::vector<int>& getDiffTimesT() const { return m_diffTimesT; }
00067     const std::vector<int>& getDiffTimesG() const { return m_diffTimesG; }
00068     const std::vector<double>& getDiffTimesARelative() const { return
      m_diffTimesARelative; }
00069     const std::vector<double>& getDiffTimesCRelative() const { return
      m_diffTimesCRelative; }
00070     const std::vector<double>& getDiffTimesTRelative() const { return
      m_diffTimesTRelative; }
00071     const std::vector<double>& getDiffTimesGRelative() const { return
```

```
      m_diffTimesGRelative; }
00072     //-------------------------------------------------------------
00073     const String& getStatusMessage() const { return m_statusText; }
00074
00078     void calculate();
00079
00083     void clearValues();
00084
00085 private:
00086
00090     void checkCloneVsGl();
00091
00095     void sortClones();
00096
00100     void countMutationOnPosition();
00101
00105     void calculateIntermediateClones();
00106
00110         void calculateIntermediateClonesUnique();
00111
00115         void calculatePm();
00116
00117 private:
00118     String m_gl;
00119     std::vector<Clone> m_cloneVec;
00120     std::vector<CloneVsGl> m_cloneVsGl;
00121     std::vector<CloneVsGl> m_cloneVsGlSorted;
00122     std::vector<std::vector<CloneVsGl> > m_cloneRefs;
00123     std::vector<CloneRefsAndPosition> m_cloneRefsSorted;
00124     std::vector<IntermediateClones> m_intermediatClones;
00125     std::vector<IntermediateClones> m_intermediatClonesSorted;
00126         std::vector<IntermediateClones> m_intermediateClonesUnique;
00127         std::vector<IntermediateClones> m_intermediateClonesUniqueSorted;
00128
00129     int m_seqLength;
00130     std::vector<int> m_diffTimesA;
00131     std::vector<int> m_diffTimesC;
00132     std::vector<int> m_diffTimesT;
00133     std::vector<int> m_diffTimesG;
00134     std::vector<double> m_diffTimesARelative;
00135     std::vector<double> m_diffTimesCRelative;
00136     std::vector<double> m_diffTimesTRelative;
00137     std::vector<double> m_diffTimesGRelative;
00138     String m_statusText;
00139
00140         //========================================================================
      =======
00142         CloneCalculations(const CloneCalculations&);
00144         const CloneCalculations& operator=(const CloneCalculations&);
00145
00146 };
00147
00148 #endif // _CLONECALCULATIONS_H_
```

5.5 Data.h File Reference

```
#include <vector>
```

Classes

- struct Clone

  Struct for name and sequence.

- struct CloneVsGl

  Clones with metadata.

- struct CloneRefsAndPosition

  Clones with locus.

- struct IntermediateClones

  HPCs.

5.6 Data.h

```
00001 /*=============================================================================
00002  * Copyright (C) 2010 Peter Vasil
00003  *
00004  * This program is free software; you can redistribute it and/or modify
00005  * it under the terms of the GNU General Public License as published by
00006  * the Free Software Foundation; either version 3 of the License, or
00007  * (at your option) any later version.
00008  *
00009  * This program is distributed in the hope that it will be useful,
00010  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00012  * GNU General Public License for more details.
00013  *
00014  * You should have received a copy of the GNU General Public License
00015  * along with this program; if not, see <http://www.gnu.org/licenses/>.
00016  *
00017  ============================================================================*/
00018
00019 #ifndef _DATA_H_
00020 #define _DATA_H_
00021
00022 #include <vector>
00023
00028 struct Clone {
00029     String name;
00030     String seq;
00031 };
00032
00037 struct CloneVsGl {
00038     String name;
00039     String seq;
00040     String diff;
00041     std::vector<int> mutPos;
00042     int mutNum;
00043     bool operator()(CloneVsGl lhs, CloneVsGl rhs)
00044     {
00045         return lhs.mutNum < rhs.mutNum;
00046     }
00047 };
00048
00053 struct CloneRefsAndPosition {
00054     int pos;
00055     std::vector<CloneVsGl> clones;
00056     bool operator()(CloneRefsAndPosition lhs, CloneRefsAndPosition rhs)
00057     {
00058         return lhs.clones.size() > rhs.clones.size();
00059     }
00060 };
00061
00066 struct IntermediateClones {
00067     int pos;
00068     int numMut;
00069     String intermediateClone;
00070     std::vector<CloneVsGl> clones;
00071     double pm;
00072     int z;
00073     bool shouldDisplay;
00074     bool operator()(IntermediateClones lhs, IntermediateClones rhs)
00075     {
00076         return lhs.pm > rhs.pm;
00077     }
00078 };
00079
00080 #endif // _DATA_H_
```

5.7 FileWriter.cpp File Reference

```
#include "FileWriter.h"
#include "FLER_Headers.h"
#include "CloneCalculations.h"
```

5.8 FileWriter.cpp

```
00001 /*=========================================================================
00002  * Copyright (C) 2010 Peter Vasil
00003  *
00004  * This program is free software; you can redistribute it and/or modify
00005  * it under the terms of the GNU General Public License as published by
00006  * the Free Software Foundation; either version 3 of the License, or
00007  * (at your option) any later version.
00008  *
00009  * This program is distributed in the hope that it will be useful,
00010  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00012  * GNU General Public License for more details.
00013  *
00014  * You should have received a copy of the GNU General Public License
00015  * along with this program; if not, see <http://www.gnu.org/licenses/>.
00016  *
00017  =========================================================================*/
00018
00019 #include "FileWriter.h"
00020
00021 FileWriter::FileWriter(CloneCalculations* cloneCalculations,
00022                        const String saveFilePath,
00023                        bool writePmHierarchy,
00024                        bool writeClonalMetadata,
00025                        bool writeClonalGroups)
00026 :
00027 ThreadWithProgressWindow(T("Writing file..."), true, false),
00028 m_cloneCalculations(cloneCalculations),
00029 m_saveFilePath(saveFilePath),
00030 m_stream(String::empty),
00031 m_writePmHierarchy(writePmHierarchy),
00032 m_writeClonalMetadata(writeClonalMetadata),
00033 m_writeClonalGroups(writeClonalGroups)
00034 {
00035 }
00036
00037 FileWriter::~FileWriter()
00038 {
00039 }
00040
00041 void FileWriter::run()
00042 {
00043     setProgress(0.0);
00044     //------------------------------------------------------------
00045     writeHeader();
00046     //------------------------------------------------------------------------------

00047     if (m_writePmHierarchy)
00048     {
00049         writeIntermediateSequencesCalculated();
00050         writeMutationOnPosition();
00051         setProgress(0.3);
00052     }
00053     if (m_writeClonalMetadata)
00054     {
00055         writeClonalMetadata();
00056         setProgress(0.6);
00057     }
00058     if (m_writeClonalGroups)
00059     {
00060         writeIntermediateSequencesUnique();
00061         setProgress(0.9);
00062     }
00063     //------------------------------------------------------------------------------
```

```
00064     writeFile();
00065     setProgress(1.0);
00066     //--------------------------------------------------------------------------

00067 }
00068
00069 void FileWriter::writeHeader()
00070 {
00071     m_stream += T("\n");
00072     m_stream += T("            <<<____________________FLER____________________>>>\n");

00073     m_stream += T("            (Follicular Lymphoma Evolution Reconstructor)\n");

00074     m_stream += T("                                result file                \n");

00075     int charNum = 70;
00076     String currTime = T("[");
00077     currTime += Time::getCurrentTime().toString(true, true, true, true);
00078     currTime += T("]");
00079     juce_wchar padChar = *String(" ");
00080     int timeStrLength = currTime.length();
00081     currTime = currTime.paddedLeft(padChar, charNum/2 + timeStrLength/2);
00082     m_stream += currTime;
00083     m_stream += T("\n\n");
00084     m_stream += T("*****************************************************************
    ********\n");
00085     m_stream += T("* Code:    Peter Vasil, Technical University, Berlin.
        *\n");
00086     m_stream += T("*          peter.vasil@campus.tu-berlin.de
        *\n");
00087     m_stream += T("* Method:  Martin Wartenberg, Technical University, Munich.
        *\n");
00088     m_stream += T("*          moatlcombat@gmx.de
        *\n");
00089     m_stream += T("*
        *\n");
00090     m_stream += T("*****************************************************************
    ********\n");
00091
00092 }
00093
00094 void FileWriter::writeClonalMetadata()
00095 {
00096     m_stream += T("\n\n\n");
00097     m_stream += T("***********************************************************\n");

00098     m_stream += T("*                                                         *\n");

00099     m_stream += T("* CLONAL METADATA                                         *\n");

00100     m_stream += T("*                                                         *\n");

00101     m_stream += T("***********************************************************\n\n"
    );
00102     m_stream += T("germline: \n");
00103     for(int ii = 0; ii < m_cloneCalculations->getGlString().length(); ++ii)
00104     {
00105         m_stream += m_cloneCalculations->getGlString()[ii];
00106         m_stream += T("\t");
00107     }
00108 //    m_stream += m_cloneCalculations->getGlString();
00109     m_stream += T("\n");
00110     for (unsigned int i = 0; i < m_cloneCalculations->getCloneVsGl().size(); ++i)

00111     {
00112         m_stream += T("clone: \t");
00113         m_stream += m_cloneCalculations->getCloneVsGl()[i].name;
```

```
00114         m_stream += T("\n");
00115         m_stream += T("sequence: \n");
00116         for (int ii = 0; ii < m_cloneCalculations->getCloneVsGl()[i].seq.length()
      ; ++ii)
00117         {
00118             m_stream += m_cloneCalculations->getCloneVsGl()[i].seq[ii];
00119             m_stream += T("\t");
00120         }
00121 //        m_stream += m_cloneCalculations->getCloneVsGl()[i].seq;
00122         m_stream += T("\n");
00123         m_stream += T("mutation pattern: \n");
00124         for (int ii = 0; ii < m_cloneCalculations->getCloneVsGl()[i].diff.length(
      ); ++ii)
00125         {
00126             m_stream += m_cloneCalculations->getCloneVsGl()[i].diff[ii];
00127             m_stream += T("\t");
00128         }
00129
00130 //        m_stream += m_cloneCalculations->getCloneVsGl()[i].diff;
00131         m_stream += T("\n\n");
00132         m_stream += T("mutation position: ");
00133         for(unsigned int k = 0; k < m_cloneCalculations->getCloneVsGl()[i].mutPos
      .size(); ++k)
00134         {
00135             m_stream += String(m_cloneCalculations->getCloneVsGl()[i].mutPos[k] +
       1);
00136             if (k != m_cloneCalculations->getCloneVsGl()[i].mutPos.size() - 1)
      m_stream += T(", ");
00137         }
00138         m_stream += T("\n");
00139         m_stream += T("number of mutations: ");
00140         m_stream += String(m_cloneCalculations->getCloneVsGl()[i].mutNum);
00141         m_stream += T("\n");
00142         m_stream += T("*****************\n");
00143         m_stream += T("\n");
00144     }
00145     m_stream += T("\n");
00146     m_stream += T("================================================================
      ================================================================\n");
00147 //    m_stream += T("Sequences sorted regarding their number of mutation:\n");
00148 //    m_stream += T("\n");
00149 //    setProgress(0.1);
00150 //    for (unsigned int i = 0; i < m_cloneCalculations->getCloneVsGlSorted().size
      (); ++i)
00151 //    {
00152 //        m_stream += m_cloneCalculations->getCloneVsGlSorted()[i].name;
00153 //        m_stream += T(" :\t");
00154 //        m_stream += m_cloneCalculations->getCloneVsGlSorted()[i].diff;
00155 //        m_stream += T("\n");
00156 //    }
00157 //    m_stream += T("\n");
00158 //    m_stream += T("================\n");
00159 //    m_stream += T("\n");
00160
00161
00162 }
00163
00164 void FileWriter::writeMutationOnPosition()
00165 {
00166     m_stream += T("\n\n\n");
00167     m_stream += T("**********************************************************\n");

00168     m_stream += T("*                                                        *\n");

00169     m_stream += T("* MUTATIONAL METADATA                                    *\n");

00170     m_stream += T("*                                                        *\n");
```

```
00171     m_stream += T("************************************************************\n\n"
      );
00172     m_stream += T("sequence length: ");
00173     m_stream += String(m_cloneCalculations->getSeqLength());
00174     m_stream += T("\n");
00175 
00176     m_stream += T("locus-specific number of mutations:\n");
00177     m_stream += T("\n");
00178     m_stream += T("locus: \n");
00179     setProgress(0.2);
00180     for (unsigned int i = 0; i < m_cloneCalculations->getCloneRefs().size(); ++i)

00181     {
00182         m_stream += String((int)(i+1));
00183         if (i < m_cloneCalculations->getCloneRefs().size())
00184         {
00185             m_stream += T("\t");
00186         };
00187     }
00188     m_stream += T("\n");
00189     m_stream += T("number of mutations: \n");
00190     for (unsigned int i = 0; i < m_cloneCalculations->getCloneRefs().size(); ++i)

00191     {
00192         if ((int)m_cloneCalculations->getCloneRefs()[i].size() != 0)
00193         {
00194             m_stream += String((int)m_cloneCalculations->getCloneRefs()[i].size()
      );
00195             if (i < m_cloneCalculations->getCloneRefs().size()-1)
00196             {
00197                 m_stream += T("\t");
00198             };
00199         }
00200         else
00201         {
00202             m_stream += T("-");
00203             if (i < m_cloneCalculations->getCloneRefs().size()-1)
00204             {
00205                 m_stream += T("\t");
00206             };
00207         }
00208 
00209     }
00210     m_stream += T("\n");
00211     // ------------------------------------------------------------------------
     ---
00212     m_stream += T("\n");
00213     m_stream += T("detailed mutations:\n");
00214     m_stream += T("\n");
00215     for(int i = 0; i < m_cloneCalculations->getSeqLength(); ++i)
00216     {
00217         m_stream += T("A");
00218         if (i < m_cloneCalculations->getSeqLength()-1) {m_stream += T("\t");};
00219     }
00220     m_stream += T("\n");
00221     for(int i = 0; i < m_cloneCalculations->getSeqLength(); ++i)
00222     {
00223         if(m_cloneCalculations->getDiffTimesA()[i] == 0)
00224         {
00225             m_stream += T("-");
00226         }
00227         else
00228         {
00229             m_stream += String(m_cloneCalculations->getDiffTimesA()[i]);
00230         }
00231         if (i < m_cloneCalculations->getSeqLength()-1) {m_stream += T("\t");};
```

```
00232     }
00233     m_stream += T("\n");
00234     for(int i = 0; i < m_cloneCalculations->getSeqLength(); ++i)
00235     {
00236         m_stream += T("C");
00237         if (i < m_cloneCalculations->getSeqLength()-1) {m_stream += T("\t");};
00238     }
00239     m_stream += T("\n");
00240     for(int i = 0; i < m_cloneCalculations->getSeqLength(); ++i)
00241     {
00242         if(m_cloneCalculations->getDiffTimesC()[i] == 0)
00243         {
00244             m_stream += T("-");
00245         }
00246         else
00247         {
00248             m_stream += String(m_cloneCalculations->getDiffTimesC()[i]);
00249         }
00250         if (i < m_cloneCalculations->getSeqLength()-1) {m_stream += T("\t");};
00251     }
00252     m_stream += T("\n");
00253 
00254     for(int i = 0; i < m_cloneCalculations->getSeqLength(); ++i)
00255     {
00256         m_stream += T("T");
00257         if (i < m_cloneCalculations->getSeqLength()-1) {m_stream += T("\t");};
00258     }
00259     m_stream += T("\n");
00260     for(int i = 0; i < m_cloneCalculations->getSeqLength(); ++i)
00261     {
00262         if(m_cloneCalculations->getDiffTimesT()[i] == 0)
00263         {
00264             m_stream += T("-");
00265         }
00266         else
00267         {
00268             m_stream += String(m_cloneCalculations->getDiffTimesT()[i]);
00269         }
00270         if (i < m_cloneCalculations->getSeqLength()-1) {m_stream += T("\t");};
00271     }
00272     m_stream += T("\n");
00273 
00274     for(int i = 0; i < m_cloneCalculations->getSeqLength(); ++i)
00275     {
00276         m_stream += T("G");
00277         if (i < m_cloneCalculations->getSeqLength()-1) {m_stream += T("\t");};
00278     }
00279     m_stream += T("\n");
00280     for(int i = 0; i < m_cloneCalculations->getSeqLength(); ++i)
00281     {
00282         if(m_cloneCalculations->getDiffTimesG()[i] == 0)
00283         {
00284             m_stream += T("-");
00285         }
00286         else
00287         {
00288             m_stream += String(m_cloneCalculations->getDiffTimesG()[i]);
00289         }
00290         if (i < m_cloneCalculations->getSeqLength()-1) {m_stream += T("\t");};
00291     }
00292     m_stream += T("\n");
00293 
00294     //-------------------------------------------------------------------------

00295 
00296     m_stream += T("\n");
00297     m_stream += T("relative freuency of mutations P:\n");
```

```
00298     m_stream += T("\n");
00299     for(int i = 0; i < m_cloneCalculations->getSeqLength(); ++i)
00300     {
00301         m_stream += T("A");
00302         if (i < m_cloneCalculations->getSeqLength()-1) {m_stream += T("\t");};
00303     }
00304     m_stream += T("\n");
00305     for(int i = 0; i < m_cloneCalculations->getSeqLength(); ++i)
00306     {
00307         if(m_cloneCalculations->getDiffTimesARelative()[i] == 0)
00308         {
00309             m_stream += T("-");
00310             m_stream += T("\t");
00311         }
00312         else
00313         {
00314             m_stream += String(m_cloneCalculations->getDiffTimesARelative()[i],2)
      ;
00315 //             if (String(m_cloneCalculations->getDiffTimesARelative()[i]).length(
      ) < 4)
00316 //             {
00317 //                 m_stream += T("\t");
00318 //             }
00319         }
00320 //         if (i < m_cloneCalculations->getSeqLength()-1) {m_stream += T("\t");};
00321     }
00322     m_stream += T("\n");
00323     for(int i = 0; i < m_cloneCalculations->getSeqLength(); ++i)
00324     {
00325         m_stream += T("C");
00326         if (i < m_cloneCalculations->getSeqLength()-1) {m_stream += T("\t");};
00327     }
00328     m_stream += T("\n");
00329     for(int i = 0; i < m_cloneCalculations->getSeqLength(); ++i)
00330     {
00331         if(m_cloneCalculations->getDiffTimesCRelative()[i] == 0)
00332         {
00333             m_stream += T("-");
00334             m_stream += T("\t");
00335         }
00336         else
00337         {
00338             m_stream += String(m_cloneCalculations->getDiffTimesCRelative()[i],2)
      ;
00339 //             if (String(m_cloneCalculations->getDiffTimesCRelative()[i]).length(
      ) < 4)
00340 //             {
00341 //                 m_stream += T("\t");
00342 //             }
00343 
00344         }
00345 //         if (i < m_cloneCalculations->getSeqLength()-1) {m_stream += T("\t");};
00346     }
00347     m_stream += T("\n");
00348 
00349     for(int i = 0; i < m_cloneCalculations->getSeqLength(); ++i)
00350     {
00351         m_stream += T("T");
00352         if (i < m_cloneCalculations->getSeqLength()-1) {m_stream += T("\t");};
00353     }
00354     m_stream += T("\n");
00355     for(int i = 0; i < m_cloneCalculations->getSeqLength(); ++i)
00356     {
00357         if(m_cloneCalculations->getDiffTimesTRelative()[i] == 0)
00358         {
00359             m_stream += T("-");
00360             m_stream += T("\t");
```

```
00361         }
00362         else
00363         {
00364             m_stream += String(m_cloneCalculations->getDiffTimesTRelative()[i],2)
      ;
00365 //            if (String(m_cloneCalculations->getDiffTimesTRelative()[i]).length(
      ) < 4)
00366 //            {
00367 //                m_stream += T("\t");
00368 //            }
00369 
00370         }
00371 //        if (i < m_cloneCalculations->getSeqLength()-1) {m_stream += T("\t");};
00372     }
00373     m_stream += T("\n");
00374 
00375     for(int i = 0; i < m_cloneCalculations->getSeqLength(); ++i)
00376     {
00377         m_stream += T("G");
00378         if (i < m_cloneCalculations->getSeqLength()-1) {m_stream += T("\t");};
00379     }
00380     m_stream += T("\n");
00381     for(int i = 0; i < m_cloneCalculations->getSeqLength(); ++i)
00382     {
00383         if(m_cloneCalculations->getDiffTimesGRelative()[i] == 0)
00384         {
00385             m_stream += T("-");
00386             m_stream += T("\t");
00387         }
00388         else
00389         {
00390             m_stream += String(m_cloneCalculations->getDiffTimesGRelative()[i],2)
      ;
00391 //            if (String(m_cloneCalculations->getDiffTimesGRelative()[i]).length(
      ) < 4)
00392 //            {
00393 //                m_stream += T("\t");
00394 //            }
00395 
00396         }
00397 //        if (i < m_cloneCalculations->getSeqLength()-1) {m_stream += T("\t");};
00398     }
00399     m_stream += T("\n\n\n");
00400     m_stream += T("=================================================================
      ================================================================\n");
00401 
00402 }
00403 
00404 void FileWriter::writeIntermediateSequencesUnique()
00405 {
00406         m_stream += T("\n\n\n");
00407         m_stream += T("**********************************************************
      ***********************************************\n");
00408         m_stream += T("*
                                                *\n");
00409         m_stream += T("* HPC DETERMINATION BY CREATION OF CLONAL GROUPS DELINEATE
      D FROM LOCUS-SPECIFIC, IDENTICAL MUTATIONS      *\n");
00410         m_stream += T("*
                                                *\n");
00411         m_stream += T("**********************************************************
      ***********************************************\n");
00412         m_stream += T("\n\n");
00413 
00414         for(unsigned int i = 0; i < m_cloneCalculations->
      getIntermediateClonesUnique().size(); ++i)
00415         {
00416                 m_stream += T("Derived from locus ");
```

```
00417                 m_stream += String(m_cloneCalculations->
      getIntermediateClonesUnique()[i].pos + 1);
00418                 //m_stream += T(":\n");
00419                 m_stream += T("\n");
00420                 m_stream += T("number of clones r, theoretically evolving from th
      is HPC: ");
00421                 m_stream += String((int)m_cloneCalculations->
      getIntermediateClonesUnique()[i].clones.size());
00422                 m_stream += T("\n");
00423 //              m_stream += T("Sequences sharing the intermediate: ");
00424 //              m_stream += T("\n");
00425                 m_stream += T("number of mutations i: ");
00426                 m_stream += String(m_cloneCalculations->
      getIntermediateClonesUnique()[i].numMut);
00427                 m_stream += T("\n\n");
00428                 m_stream += T("HPC sequence: \n");
00429 //              String intermediate = m_cloneCalculations->getIntermediateClonesU
      nique()[i].intermediateClone;
00430 //              for(unsigned int ii = 0; ii < intermediate.length(); ++ii)
00431 //              {
00432 //            m_stream += intermediate[ii];
00433 //            m_stream += T("\t");
00434 //              }
00435                 m_stream += m_cloneCalculations->getIntermediateClonesUnique()[i]
      .intermediateClone;
00436                 m_stream += T("\n\n");
00437                 for(unsigned int j = 0; j < m_cloneCalculations->
      getIntermediateClonesUnique()[i].clones.size(); ++j)
00438                 {
00439                         m_stream += m_cloneCalculations->
      getIntermediateClonesUnique()[i].clones[j].name;
00440                         m_stream += T(":\n");
00441 //                      String diff = m_cloneCalculations->getIntermediateClonesU
      nique()[i].clones[j].diff;
00442 //                      for(unsigned int ii = 0; ii < diff.length(); ++ii)
00443 //                      {
00444 //                m_stream += diff[ii];
00445 //                      }
00446                         m_stream += m_cloneCalculations->
      getIntermediateClonesUnique()[i].clones[j].diff;
00447                         m_stream += T("\n");
00448                 }
00449                 m_stream += T("**********************\n");
00450                 m_stream += T("\n");
00451         }
00452
00453         m_stream += T("\n\n");
00454         m_stream += T("==========================================================
      ==========================================================================\n");
00455
00456 }
00457
00458 void FileWriter::writeIntermediateSequencesCalculated()
00459 {
00460     m_stream += T("\n\n\n");
00461     m_stream += T("**********************************************************
      ************************************************************\n");
00462     m_stream += T("*
                                                                *\n");
00463         m_stream += T("* INPUT PARAMETERS OF PM-FORMULA , PM-HIRACHY, AND MUTATIO
      N PATTERNS OF HYPOTHETICAL PREDECESSOR CLONES (HPC)               *\n");
00464         m_stream += T("*
                                                                *\n");
00465         m_stream += T("**********************************************************
      ************************************************************\n");
00466         m_stream += T("\n\n");
00467         m_stream += T("For pm-formula see http://www.petervasil.net/fler\n");
```

```
00468         m_stream += T("\n");
00469         m_stream += T("\t\tpm = probability measure\n");
00470         m_stream += T("\t\tt = total number of operational taxonomic units (OTUs)
       / input clones\n");
00471         m_stream += T("\t\tP = relative frequency of a considered mutation among
      the whole population of OTUs\n");
00472         m_stream += T("\t\tr = number of OTUs in an OTU subgroup defining the con
      sidered HPC\n");
00473         m_stream += T("\t\tz = number of reapeats of a HPC with the identical mut
      ation patterns among the abundance of HPCs\n");
00474         m_stream += T("\t\ti = nuber of mutations of an HPC (number of mutations
      shared by all clones of the HPC-defining subgroup)\n");
00475         m_stream += T("\n");
00476         m_stream += T("number of OTUs / input clones t: ");
00477         m_stream += (int)m_cloneCalculations->getClones().size();
00478         m_stream += T("\n\n");
00479     m_stream += T("pm          |     r |     z |     i | locus | (only one locus
      is shown, in case z > 1, see section \"HPC DETERMINATION BY CREATION OF CLONAL GR
      OUPS DELINEATED FROM LOCUS-SPECIFIC, IDENTICAL MUTATIONS\" for further loci.)\n")
      ;
00480     m_stream += T("-----------------------------------------\n");
00481     for(unsigned int i = 0; i < m_cloneCalculations->
      getIntermediateClonesUniqueSorted().size(); ++i)
00482     {
00483         if(m_cloneCalculations->getIntermediateClonesUniqueSorted()[i].shouldDisp
      lay)
00484         {
00485             String pm_ = String(m_cloneCalculations->
      getIntermediateClonesUniqueSorted()[i].pm, 9);
00486             String r_ = String((int)m_cloneCalculations->
      getIntermediateClonesUniqueSorted()[i].clones.size());
00487             String z_ = String((int)m_cloneCalculations->
      getIntermediateClonesUniqueSorted()[i].z);
00488             String i_ = String((int)m_cloneCalculations->
      getIntermediateClonesUniqueSorted()[i].numMut);
00489             String locus = String((int)m_cloneCalculations->
      getIntermediateClonesUniqueSorted()[i].pos + 1);
00490
00491             juce_wchar padChar = *String(" ");
00492             m_stream += pm_;
00493             m_stream += T(" | ");
00494             m_stream += r_.paddedLeft(padChar, 5);
00495             m_stream += T(" | ");
00496             m_stream += z_.paddedLeft(padChar,5);
00497             m_stream += T(" | ");
00498             m_stream += i_.paddedLeft(padChar,5);
00499             m_stream += T(" | ");
00500             m_stream += locus.paddedLeft(padChar,5);
00501             m_stream += T(" |\n");
00502             for (int ii = 0; ii < m_cloneCalculations->
      getIntermediateClonesUniqueSorted()[i].intermediateClone.length(); ++ii)
00503             {
00504                 m_stream += m_cloneCalculations->
      getIntermediateClonesUniqueSorted()[i].intermediateClone[ii];
00505                 m_stream += T("\t");
00506             }
00507
00508 //            m_stream += m_cloneCalculations->getIntermediateClonesUniqueSorted(
      )[i].intermediateClone;
00509 //            m_stream += T("\n");
00510 //            m_stream += T("clones: ");
00511 //            for(unsigned int j = 0; j < m_cloneCalculations->getIntermediateClo
      nesUniqueSorted()[i].clones.size(); ++j)
00512 //            {
00513 //                m_stream += m_cloneCalculations->getIntermediateClonesUniqueSor
      ted()[i].clones[j].name;
00514 //                m_stream += T(" ");
```

```
00515 //             }
00516             m_stream += T("\n\n");
00517 
00518         }
00519     }
00520 
00521         m_stream += T("\n\n");
00522         m_stream += T("================================================================
      ================================================================\n");
00523 }
00524 
00525 void FileWriter::writeFile()
00526 {
00527     FileOutputStream* fileOutStream;
00528     File resultFile(m_saveFilePath);
00529 
00530     fileOutStream =  new FileOutputStream(resultFile);
00531     fileOutStream->writeText(m_stream,true, true);
00532     fileOutStream->flush();
00533     SAFE_DELETE(fileOutStream);
00534     m_stream = String::empty;
00535 }
```

5.9 FileWriter.h File Reference

```
#include "FLER_Headers.h"
#include "CloneCalculations.h"
```

Classes

- class FileWriter

  Class for wrapping all functions for creating output file.

5.10 FileWriter.h

```
00001 /*=========================================================================
00002  * Copyright (C) 2010 Peter Vasil
00003  *
00004  * This program is free software; you can redistribute it and/or modify
00005  * it under the terms of the GNU General Public License as published by
00006  * the Free Software Foundation; either version 3 of the License, or
00007  * (at your option) any later version.
00008  *
00009  * This program is distributed in the hope that it will be useful,
00010  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00012  * GNU General Public License for more details.
00013  *
00014  * You should have received a copy of the GNU General Public License
00015  * along with this program; if not, see <http://www.gnu.org/licenses/>.
00016  *
00017  =========================================================================*/
00018
00019 #ifndef _FILEWRITER_H_
00020 #define _FILEWRITER_H_
00021
00022 #include "FLER_Headers.h"
00023 #include "CloneCalculations.h"
00024
00030 class FileWriter : public ThreadWithProgressWindow
00031 {
00032 public:
00041     FileWriter(CloneCalculations* cloneCalcs,
00042                const String saveFilePath,
00043                bool writePmHierarchy,
00044                bool writeClonalMetadata,
00045                bool writeClonalGroups);
00046     ~FileWriter();
00047
00048     void run();
00049
00050     const String& getSaveFilePath() const { return m_saveFilePath; }
00051
00052 private:
00053     void writeHeader();
00054     void writeClonalMetadata();
00055     void writeMutationOnPosition();
00056         void writeIntermediateSequencesUnique();
00057         void writeIntermediateSequencesCalculated();
00058     void writeFile();
00059
00060 private:
00061     CloneCalculations* m_cloneCalculations;
00062     String m_saveFilePath;
00063     String m_stream;
00064     bool m_writeClonalMetadata;
00065     bool m_writePmHierarchy;
00066     bool m_writeClonalGroups;
00067 };
00068
00069 #endif // _FILEWRITER_H_
```

5.11 FLER_Headers.h File Reference

```
#include "../JuceLibraryCode/JuceHeader.h"
#include "PVLogger.h"
```

Defines

- #define RC_COMPANY_STR "Peter Vasil"
- #define SAFE_DELETE(p) {if(p) {delete (p); (p)=NULL;}}
- #define SAFE_DELETE_ARRAY(p) {if(p) {delete[ ] (p); (p)=NULL;}}

5.11.1 Define Documentation

5.11.1.1 #define RC_COMPANY_STR "Peter Vasil"

Definition at line 22 of file FLER_Headers.h.

5.11.1.2 #define SAFE_DELETE(p) {if(p) {delete (p); (p)=NULL;}}

Definition at line 27 of file FLER_Headers.h.

5.11.1.3 #define SAFE_DELETE_ARRAY(p) {if(p) {delete[ ] (p); (p)=NULL;}}

Definition at line 28 of file FLER_Headers.h.

5.12 FLER_Headers.h

```
00001 /*==============================================================================
00002  * Copyright (C) 2010 Peter Vasil
00003  *
00004  * This program is free software; you can redistribute it and/or modify
00005  * it under the terms of the GNU General Public License as published by
00006  * the Free Software Foundation; either version 3 of the License, or
00007  * (at your option) any later version.
00008  *
00009  * This program is distributed in the hope that it will be useful,
00010  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00012  * GNU General Public License for more details.
00013  *
00014  * You should have received a copy of the GNU General Public License
00015  * along with this program; if not, see <http://www.gnu.org/licenses/>.
00016  *
00017  ==============================================================================*/
00018
00019 #ifndef _FLER_HEADERS_H_
00020 #define _FLER_HEADERS_H_
00021
00022 #define RC_COMPANY_STR "Peter Vasil"
00023 //#define RC_APPDATE_STR "2010-05-09"
00024
00025 //------------------------------------------------------------
00026
00027 #define SAFE_DELETE(p) {if(p) {delete (p); (p)=NULL;}}
00028 #define SAFE_DELETE_ARRAY(p) {if(p) {delete[] (p); (p)=NULL;}}
00029
00030 //------------------------------------------------------------
00031
00032 #ifdef _WIN32
00033 # include <windows.h>
00034 #endif
00035
00036 #include "../JuceLibraryCode/JuceHeader.h"
00037
00038 #include "PVLogger.h"
00039
00040 #endif // _FLER_HEADERS_H_
```

5.13 Main.cpp File Reference

```
#include "FLER_Headers.h"
#include "MainAppWindow.h"
```

Classes

- class AppClass

  Application class.

5.14 Main.cpp

```
00001 /*==============================================================================
00002  * Copyright (C) 2010 Peter Vasil
00003  *
00004  * This program is free software; you can redistribute it and/or modify
00005  * it under the terms of the GNU General Public License as published by
00006  * the Free Software Foundation; either version 3 of the License, or
00007  * (at your option) any later version.
00008  *
00009  * This program is distributed in the hope that it will be useful,
00010  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00012  * GNU General Public License for more details.
00013  *
00014  * You should have received a copy of the GNU General Public License
00015  * along with this program; if not, see <http://www.gnu.org/licenses/>.
00016  *
00017  ==============================================================================*/
00018
00019 #include "FLER_Headers.h"
00020 #include "MainAppWindow.h"
00021
00107 //==============================================================================
00112 class AppClass : public JUCEApplication
00113         {
00114                 MainAppWindow* theMainWindow;
00115
00116                 ApplicationCommandManager* m_commandManager;
00117
00118         public:
00119                 //==========================================================================
    ===============
00120                 AppClass()
00121         : theMainWindow (0),
00122                   m_commandManager(0)
00123                 {
00124                 }
00125
00126                 ~AppClass()
00127                 {
00128                 }
00129
00130                 //==========================================================================
    ===============
00131                 void initialise (const String& commandLine)
00132                 {
00133                         m_commandManager = new ApplicationCommandManager();
00134                         theMainWindow = new MainAppWindow(m_commandManager);
00135                         m_commandManager->registerAllCommandsForTarget(JUCEApplic
    ation::getInstance());
00136                         theMainWindow->centreWithSize (300, 450);   // [*] (see b
    elow for a tip on this)
00137                         theMainWindow->setVisible (true);
00138
00139                 }
00140
00141                 void shutdown()
00142                 {
00143                         deleteAndZero (theMainWindow);
00144                         deleteAndZero(m_commandManager);
00145                 }
00146
00147                 //==========================================================================
    ===============
00148                 const String getApplicationName()
00149                 {
```

```
00150                           return String(ProjectInfo::projectName);
00151                   }
00152
00153                   const String getApplicationVersion()
00154                   {
00155                           return String(ProjectInfo::versionString);
00156                   }
00157
00158                   bool moreThanOneInstanceAllowed()
00159                   {
00160                           return true;
00161                   }
00162
00163                   void anotherInstanceStarted (const String& commandLine)
00164                   {
00165                   }
00166           };
00167
00168
00169 //==============================================================================
00170 // This macro creates the application's main() function..
00171 START_JUCE_APPLICATION(AppClass)
```

5.15 MainAppWindow.cpp File Reference

```
#include "FLER_Headers.h"
#include "MainAppWindow.h"
#include "MainComponent.h"
#include "CloneCalculations.h"
```

5.16 MainAppWindow.cpp

```
00001 /*==============================================================================
00002  * Copyright (C) 2010 Peter Vasil
00003  *
00004  * This program is free software; you can redistribute it and/or modify
00005  * it under the terms of the GNU General Public License as published by
00006  * the Free Software Foundation; either version 3 of the License, or
00007  * (at your option) any later version.
00008  *
00009  * This program is distributed in the hope that it will be useful,
00010  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00012  * GNU General Public License for more details.
00013  *
00014  * You should have received a copy of the GNU General Public License
00015  * along with this program; if not, see <http://www.gnu.org/licenses/>.
00016  *
00017  ==============================================================================*/
00018
00019 #include "FLER_Headers.h"
00020 #include "MainAppWindow.h"
00021 #include "MainComponent.h"
00022
00023 //==============================================================================
00024 MainAppWindow::MainAppWindow(ApplicationCommandManager* commandManager)
00025 :
00026 DocumentWindow ( String(ProjectInfo::projectName),
00027                                 Colours::lightgrey,
00028                                 101,
00029                                 true
00030                                 )
00031 {
00032     setResizable (true, false);
00033
00034         setTitleBarHeight (25);
00035
00036         MainComponent* contentComponent = new MainComponent (this, commandManager
      );
00037     setContentComponent (contentComponent);
00038
00039     setMenuBar (contentComponent);
00040     this->addKeyListener (commandManager->getKeyMappings());
00041
00042         //setMenuBar(setMacMainMenu(contentComponent));
00043     setUsingNativeTitleBar(false);
00044     setResizable(false, false);
00045     //setTitleBarButtonsRequired();
00046 }
00047
00048 MainAppWindow::~MainAppWindow()
00049 {
00050     setMenuBar(0);
00051
00052         setContentComponent(0, true);
00053 }
00054
00055 void MainAppWindow::closeButtonPressed()
00056 {
00057         JUCEApplication::getInstance()->systemRequestedQuit();
00058 }
```

5.17 MainAppWindow.h File Reference

Classes

- class MainAppWindow

  Main application window.

5.18 MainAppWindow.h

```
00001 /*==============================================================================
00002  * Copyright (C) 2010 Peter Vasil
00003  *
00004  * This program is free software; you can redistribute it and/or modify
00005  * it under the terms of the GNU General Public License as published by
00006  * the Free Software Foundation; either version 3 of the License, or
00007  * (at your option) any later version.
00008  *
00009  * This program is distributed in the hope that it will be useful,
00010  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00012  * GNU General Public License for more details.
00013  *
00014  * You should have received a copy of the GNU General Public License
00015  * along with this program; if not, see <http://www.gnu.org/licenses/>.
00016  *
00017  ==============================================================================*/
00018
00019 #ifndef _MainAppWindow_H__
00020 #define _MainAppWindow_H__
00021
00028 //==============================================================================
00029 class MainAppWindow  : public DocumentWindow
00030         {
00031         public:
00032                 //==========================================================
      ===============
00033                 MainAppWindow(ApplicationCommandManager* commandManager);
00034                 ~MainAppWindow();
00035
00036                 //==========================================================
      ===============
00038                 void closeButtonPressed();
00039         };
00040
00041
00042 #endif
```

5.19 MainComponent.cpp File Reference

```
#include "FLER_Headers.h"
#include "MainComponent.h"
#include "Data.h"
#include "FileWriter.h"
```

5.20 MainComponent.cpp

```
00001 /*==============================================================================
00002  * Copyright (C) 2010 Peter Vasil
00003  *
00004  * This program is free software; you can redistribute it and/or modify
00005  * it under the terms of the GNU General Public License as published by
00006  * the Free Software Foundation; either version 3 of the License, or
00007  * (at your option) any later version.
00008  *
00009  * This program is distributed in the hope that it will be useful,
00010  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00012  * GNU General Public License for more details.
00013  *
00014  * You should have received a copy of the GNU General Public License
00015  * along with this program; if not, see <http://www.gnu.org/licenses/>.
00016  *
00017  ==============================================================================*/
00018
00019 #include "FLER_Headers.h"
00020 #include "MainComponent.h"
00021 #include "Data.h"
00022 #include "FileWriter.h"
00023
00024 MainComponent::MainComponent(DocumentWindow* mainWindow, ApplicationCommandManage
      r* commandManager)
00025     :
00026     m_mainWindow(mainWindow),
00027     m_commandManager(commandManager),
00028     m_btGl(0),
00029     m_btKlone(0),
00030     m_btCloneVsGl(0),
00031     m_btRestart(0),
00032     m_btSaveFile(0),
00033     m_statusText(String::empty),
00034     m_seqLength(195),
00035     m_txSeqLen(0),
00036     m_cloneClaculations(NULL),
00037     m_writePmHierarchy(true),
00038     m_writeClonalMetadata(true),
00039     m_writeClonalGroups(true)
00040 {
00041     m_commandManager->registerAllCommandsForTarget(this);
00042     setApplicationCommandManagerToWatch(m_commandManager);
00043
00044     m_btGl = new TextButton(T("GERMLINE"));
00045     m_btGl->addButtonListener(this);
00046     addAndMakeVisible(m_btGl);
00047
00048     m_btKlone = new TextButton(T("CLONES"));
00049     m_btKlone->addButtonListener(this);
00050     addAndMakeVisible(m_btKlone);
00051     m_btKlone->setEnabled(false);
00052
00053     m_statusText = T("Follicular Lymphoma Evolution Reconstructor!");
00054     m_btCloneVsGl = new TextButton("RUN FLER");
00055     m_btCloneVsGl->addButtonListener(this);
00056     addAndMakeVisible(m_btCloneVsGl);
00057     m_btCloneVsGl->setEnabled(false);
00058
00059     m_btRestart = new TextButton(T("RESET"));
00060     m_btRestart->addButtonListener(this);
00061     addAndMakeVisible(m_btRestart);
00062
00063     m_btSaveFile = new TextButton("SAVE FILE");
00064     m_btSaveFile->setColour (TextButton::buttonColourId, Colour (0xffab5757));
```

```
00065     m_btSaveFile->setColour (TextButton::buttonOnColourId, Colour (0xff003c00));
00066     m_btSaveFile->addButtonListener(this);
00067     m_btSaveFile->setEnabled(false);
00068     addAndMakeVisible(m_btSaveFile);
00069
00070     m_txSeqLen = new TextEditor("Sequence length input");
00071     m_txSeqLen->addListener(this);
00072     m_txSeqLen->setMultiLine(false);
00073     m_txSeqLen->setInputRestrictions(3, T("1234567890"));
00074     addAndMakeVisible(m_txSeqLen);
00075
00076     m_cloneClaculations = new CloneCalculations();
00077     m_cloneClaculations->setSeqLength(m_seqLength);
00078
00079     m_props = new PropertiesFile(PropertiesFile::getDefaultAppSettingsFile(String
      (ProjectInfo::projectName),
00080                                  T("settings"),
00081                                  String::empty,
00082                                  false),
00083                                  3000,
00084                                  PropertiesFile::storeAsXML);
00085
00086     m_lastGlFolderPath = m_props->getValue(T("LastGlFolder"), File::getSpecialLoc
      ation(File::userHomeDirectory).getFullPathName());
00087     m_lastClonesFolderPath = m_props->getValue(T("LastClonesFolder"), File::getSp
      ecialLocation(File::userHomeDirectory).getFullPathName());
00088     m_lastSaveFolderPath = m_props->getValue(T("LastSaveFolder"), File::getSpecia
      lLocation(File::userHomeDirectory).getFullPathName());
00089     m_seqLength = m_props->getIntValue(T("SequenceLength"), 195);
00090     m_writePmHierarchy = m_props->getBoolValue(T("PmHierarchy"), true);
00091     m_writeClonalMetadata = m_props->getBoolValue(T("ClonalMetadata"), true);
00092     m_writeClonalGroups = m_props->getBoolValue(T("ClonalGroups"), true);
00093
00094
00095     m_txSeqLen->setText(String(m_seqLength));
00096 }
00097
00098 MainComponent::~MainComponent()
00099 {
00100     m_props->createDefaultAppPropertiesFile(String(ProjectInfo::projectName),
00101                                             T("settings"),
00102                                             String::empty, false, 3000, 2);
00103
00104     m_props->setValue(T("LastGlFolder"), m_lastGlFolderPath);
00105     m_props->setValue(T("LastClonesFolder"), m_lastClonesFolderPath);
00106     m_props->setValue(T("LastSaveFolder"), m_lastSaveFolderPath);
00107     m_props->setValue(T("SequenceLength"), m_seqLength);
00108     m_props->setValue(T("PmHierarchy"), m_writePmHierarchy);
00109     m_props->setValue(T("ClonalMetadata"), m_writeClonalMetadata);
00110     m_props->setValue(T("ClonalGroups"), m_writeClonalGroups);
00111     m_props->save();
00112
00113     deleteAllChildren();
00114     SAFE_DELETE(m_cloneClaculations);
00115 }
00116
00117 void MainComponent::resized()
00118 {
00119     m_btGl->setBounds(getWidth() / 2 - 50,60,100,30);
00120     m_btKlone->setBounds(getWidth() / 2 - 50,110,100,30);
00121     m_btCloneVsGl->setBounds(getWidth() / 2 - 60, 170, 120, 40);
00122     m_btRestart->setBounds(getWidth() / 2 - 50, getHeight() - 70, 100, 30);
00123     m_btSaveFile->setBounds(getWidth() / 2 - 50, getHeight() -120, 100, 30);
00124     m_txSeqLen->setBounds (getWidth() / 2 - 25, 15, 50, 24);
00125 }
00126 void MainComponent::paint(Graphics& g)
00127 {
```

```
00128     g.drawSingleLineText(T("sequece length:"), 5, 33);
00129     g.drawSingleLineText(m_statusText, 5, getHeight() - 5);
00130 }
00131
00132 void MainComponent::buttonClicked(Button* button)
00133 {
00134     if (button == m_btGl)
00135     {
00136         if(getGlFile())
00137         {
00138             m_btGl->setEnabled(false);
00139             m_btKlone->setEnabled(true);
00140             repaint();
00141         }
00142     }
00143     else if (button == m_btKlone)
00144     {
00145         if(getCloneFiles())
00146         {
00147             m_btKlone->setEnabled(false);
00148             m_btCloneVsGl->setEnabled(true);
00149             repaint();
00150         }
00151     }
00152     else if (button == m_btCloneVsGl)
00153     {
00154         if (checkIfGlHasSeqLength() && checkIfClonesHaveSeqLength())
00155         {
00156             m_cloneClaculations->calculate();
00157             m_btCloneVsGl->setEnabled(false);
00158             m_btSaveFile->setEnabled(true);
00159             repaint();
00160         }
00161         else
00162         {
00163             String title = T("Error");
00164             String message =  T("The given sequence length is to long!\n");
00165             message += T("Please check length of germline or clones.");
00166             AlertWindow::showMessageBox(AlertWindow::WarningIcon, title, message,
      T("Ok"));
00167         }
00168     }
00169     else if (button == m_btRestart)
00170     {
00171         m_btGl->setEnabled(true);
00172         m_btKlone->setEnabled(false);
00173         m_btCloneVsGl->setEnabled(false);
00174         m_btSaveFile->setEnabled(false);
00175         m_cloneClaculations->clearValues();
00176         m_cloneClaculations->setSeqLength(m_seqLength);
00177         m_statusText = T("Follicular Lymphoma Evolution Reconstructor!");
00178         repaint();
00179     }
00180     else if (button == m_btSaveFile)
00181     {
00182         if(saveFile())
00183         {
00184             m_btSaveFile->setEnabled(false);
00185             repaint();
00186         }
00187     }
00188 }
00189
00190 bool MainComponent::getGlFile()
00191 {
00192     bool res = false;
00193     FileChooser myChooser ("Please select the germline file...", File(
```

```
      m_lastGlFolderPath), "*.txt", true);
00194     if (myChooser.browseForFileToOpen())
00195     {
00196         File glFile (myChooser.getResult());
00197         if(glFile.existsAsFile())
00198         {
00199             //        m_gl = glFile.loadFileAsString();
00200             String glFilePath = glFile.getFullPathName();
00201             m_cloneClaculations->setGlFile(glFilePath);
00202             m_lastGlFolderPath = glFile.getParentDirectory().getFullPathName();
00203             m_statusText = T("Got the gl!");
00204             res = true;
00205         }
00206         else
00207         {
00208             res = false;
00209         }
00210     }
00211     else
00212     {
00213         res = false;
00214     }
00215     return res;
00216 }
00217
00218 bool MainComponent::getCloneFiles()
00219 {
00220     bool res = false;
00221     FileChooser myChooser ("Please select the clone files...", File(
      m_lastClonesFolderPath), "*.txt", true);
00222     std::vector<Clone> cloneVec;
00223     if (myChooser.browseForMultipleFilesToOpen())
00224     {
00225         unsigned int fileNum = myChooser.getResults().size();
00226         for(unsigned int i = 0; i < fileNum; ++i )
00227         {
00228             File cloneFile (myChooser.getResults()[i]);
00229             String resStr = cloneFile.loadFileAsString();
00230             Clone resClone = { cloneFile.getFileNameWithoutExtension(), resStr};
00231             cloneVec.push_back(resClone);
00232         }
00233         m_lastClonesFolderPath = myChooser.getResults().getFirst().getParentDirec
      tory().getFullPathName();
00234         m_cloneClaculations->setClones(cloneVec);
00235         m_statusText = T("Got ") + String((int)m_cloneClaculations->getClones().s
      ize()) + T(" clones!");
00236         res = true;
00237     }
00238     else
00239     {
00240         res = false;
00241     }
00242     return res;
00243 }
00244
00245 bool MainComponent::checkIfGlHasSeqLength()
00246 {
00247     bool result = false;
00248     if(m_cloneClaculations->getGlString().length() >= m_seqLength)
00249         result = true;
00250     else
00251         result = false;
00252     return result;
00253 }
00254
00255 bool MainComponent::checkIfClonesHaveSeqLength()
00256 {
```

```
00257     bool result = false;
00258     for(unsigned int i = 0; i < m_cloneClaculations->getClones().size(); ++i)
00259     {
00260         if(m_cloneClaculations->getClones()[i].seq.length() >= m_seqLength)
00261         {
00262             result = true;
00263         }
00264         else
00265         {
00266             result = false;
00267             break;
00268         }
00269     }
00270     return result;
00271 }
00272
00273 bool MainComponent::saveFile()
00274 {
00275     bool res = false;
00276     String saveFilePath = String::empty;
00277     FileChooser saveDialog ("Please select the file to save results to...", File(
      m_lastSaveFolderPath), "*.txt", true);
00278     if (saveDialog.browseForFileToSave(true))
00279     {
00280         File mooseFile (saveDialog.getResult());
00281
00282         saveFilePath = mooseFile.getFullPathName();
00283         m_lastSaveFolderPath = mooseFile.getParentDirectory().getFullPathName();
00284         FileWriter fileWriter(m_cloneClaculations,
00285                               saveFilePath,
00286                               m_writePmHierarchy,
00287                               m_writeClonalMetadata,
00288                               m_writeClonalGroups);
00289         if (mooseFile.existsAsFile())
00290         {
00291             mooseFile.deleteFile();
00292         }
00293         if (fileWriter.runThread())
00294         {
00295             m_statusText = T("Saved results to ") + fileWriter.getSaveFilePath();

00296         }
00297         else
00298         {
00299             m_statusText = T("");
00300         }
00301         res = true;
00302     }
00303     else
00304     {
00305         m_statusText = T("Saving file canceled!");
00306     }
00307     return res;
00308 }
00309
00310 void MainComponent::textEditorTextChanged (TextEditor& editor)
00311 {
00312     m_seqLength = editor.getText().getIntValue();
00313     m_cloneClaculations->setSeqLength(m_seqLength);
00314     DBG_VAL(m_seqLength);
00315
00316 }
00317
00318 void MainComponent::selectedFileChanged(const File& newSelectedFile)
00319 {
00320
00321 }
```

```
00322
00323 void MainComponent::textEditorReturnKeyPressed (TextEditor& editor)
00324 {
00325
00326 }
00327
00328 void MainComponent::textEditorEscapeKeyPressed (TextEditor& editor)
00329 {
00330
00331 }
00332
00333 void MainComponent::textEditorFocusLost (TextEditor& editor)
00334 {
00335
00336 }
```

5.21 MainComponent.h File Reference

```
#include "CloneCalculations.h"
```

Classes

- class MainComponent

  Main GUI component.

5.22 MainComponent.h

```
00001 /*==============================================================================
00002  * Copyright (C) 2010 Peter Vasil
00003  *
00004  * This program is free software; you can redistribute it and/or modify
00005  * it under the terms of the GNU General Public License as published by
00006  * the Free Software Foundation; either version 3 of the License, or
00007  * (at your option) any later version.
00008  *
00009  * This program is distributed in the hope that it will be useful,
00010  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00012  * GNU General Public License for more details.
00013  *
00014  * You should have received a copy of the GNU General Public License
00015  * along with this program; if not, see <http://www.gnu.org/licenses/>.
00016  *
00017  ==============================================================================*/
00018
00019 #ifndef _MAINCOMPONENT_H_
00020 #define _MAINCOMPONENT_H_
00021
00022 #include "CloneCalculations.h"
00029 class MainComponent  : public Component,
00030                                             public MenuBarModel,
00031                                             public ApplicationCommandTarget,
00032                                             public ButtonListener,
00033                                             public TextEditorListener
00034 {
00035         enum CommandIDs
00036         {
00037             pmhierarchy         = 0x2000,
00038         clonalmetadata      = 0x2001,
00039         clonalgroups        = 0x2002,
00040                 usage               = 0x2003,
00041                 basicfunction       = 0x2004,
00042                 about               = 0x2005,
00043                 web                 = 0x2006,
00044         };
00045
00046 public:
00047                 //==============================================================
    ===============
00048         MainComponent (DocumentWindow* mainWindow, ApplicationCommandManager* com
    mandManager);
00049
00050         ~MainComponent ();
00051                 //==============================================================
    ===============
00052         juce_UseDebuggingNewOperator
00053
00054         void resized ();
00055
00056         void paint (Graphics& g);
00057
00058         void buttonClicked (Button* button);
00059
00060     bool getGlFile();
00061
00062     bool getCloneFiles();
00063
00064     bool checkIfGlHasSeqLength();
00065
00066     bool checkIfClonesHaveSeqLength();
00067
00068     bool saveFile();
```

```
00069
00070     void selectedFileChanged (const File& newSelectedFile);
00072     void textEditorTextChanged (TextEditor& editor);
00073
00075     void textEditorReturnKeyPressed (TextEditor& editor);
00076
00078     void textEditorEscapeKeyPressed (TextEditor& editor);
00079
00081     void textEditorFocusLost (TextEditor& editor);
00082
00083         const StringArray getMenuBarNames()
00084         {
00085                 const tchar* names[] = { T("save options"), T("help"), 0 };
00086
00087                 return StringArray(names);
00088
00089         }
00090
00091         const PopupMenu getMenuForIndex (int menuIndex, const String& menuName)
00092         {
00093                 PopupMenu menu;
00094                 switch (menuIndex)
00095                 {
00096                     case 0:
00097                 menu.addCommandItem(m_commandManager, pmhierarchy);
00098                 menu.addCommandItem(m_commandManager, clonalmetadata);
00099                 menu.addCommandItem(m_commandManager, clonalgroups);
00100                 break;
00101                         case 1:
00102                 menu.addCommandItem(m_commandManager, usage);
00103                 menu.addCommandItem(m_commandManager, basicfunction);
00104                                 menu.addCommandItem(m_commandManager, about);
00105                                 menu.addCommandItem(m_commandManager, web);
00106                                 break;
00107                         default:
00108                                 break;
00109                 }
00110                 return menu;
00111         }
00112
00113         void menuItemSelected (int menuItemID, int topLevelMenuIndex)
00114         {
00115         }
00116
00117         ApplicationCommandTarget* getNextCommandTarget()
00118         {
00119                 return findFirstTargetParentComponent();
00120         }
00121
00122         void getAllCommands (Array <CommandID>& commands)
00123         {
00124                 const CommandID ids[] = {
00125                     pmhierarchy,
00126                     clonalmetadata,
00127                     clonalgroups,
00128                     usage,
00129                     basicfunction,
00130                         about,
00131                         web
00132                 };
00133                 commands.addArray(ids, numElementsInArray(ids));
00134         }
00135
00136         void getCommandInfo (CommandID commandID, ApplicationCommandInfo& result)

00137         {
00138                 switch (commandID)
```

```
00139                 {
00140                     case pmhierarchy:
00141                 result.setInfo(T("pm hierarchy"), T("Writes pm hierarchy."), T("S
      ave options"), 0);
00142                 result.setTicked(m_writePmHierarchy);
00143                 break;
00144                     case clonalmetadata:
00145                 result.setInfo(T("clonal metadata"), T("Writes clonal metadata.")
      , T("Save options"), 0);
00146                 result.setTicked(m_writeClonalMetadata);
00147                 break;
00148                     case clonalgroups:
00149                 result.setInfo(T("clonal groups"), T("Writes clonal groups."), T(
      "Save options"), 0);
00150                 result.setTicked(m_writeClonalGroups);
00151                 break;
00152                         case usage:
00153                                 result.setInfo(T("user guide"), T("Displays usage
       information."), T("Help"), 0);
00154                 result.addDefaultKeypress(T('h'), 0);
00155                                 break;
00156                         case basicfunction:
00157                                 result.setInfo(T("basic function"), T("Displays b
      asic functioon of FLER."), T("Help"), 0);
00158                 result.addDefaultKeypress(T('f'), 0);
00159                                 break;
00160                         case about:
00161                                 result.setInfo(T("about "),
00162                                 T("Displays information about ") + String(Project
      Info::projectName) + T(" application"),
00163                                 T("Help"), 0);
00164                 result.addDefaultKeypress(T('a'), 0);
00165                                 break;
00166                         case web:
00167                                 result.setInfo(T("web "),
00168                                 T("Open FLER website in default browser "),
00169                                 T("Help"), 0);
00170                 result.addDefaultKeypress(T('w'), 0);
00171                                 break;
00172                         default:
00173                                 break;
00174                 }
00175         }
00176
00177         bool perform (const InvocationInfo& info)
00178         {
00179                 bool result = true;
00180                 switch(info.commandID)
00181                 {
00182                     case pmhierarchy:
00183                 m_writePmHierarchy = !m_writePmHierarchy;
00184                 break;
00185                     case clonalmetadata:
00186                 m_writeClonalMetadata = !m_writeClonalMetadata;
00187                 break;
00188                     case clonalgroups:
00189                 m_writeClonalGroups = !m_writeClonalGroups;
00190                 break;
00191                     case usage:
00192                     {
00193                 String title = T("User guide");
00194                 String message = T("-------------------------------------------
      ------------------------------------------------------\n");
00195                 message += T("Input data:\n");
00196                 message += T("sequence length   - all sequences must have the ide
      ntical length\n");
00197                 message += T("clone sequence(s) - the sequence population for tha
```

```
t the evolution reconstruction\n");
00198                 message += T("                    will be implemented by FLER\n")
;
00199                 message += T("germline sequence - this sequence will be regarded
as unmutated wildtype configuration.\n");
00200                 message += T("                    Each clone sequence will be com
pared to the germline\n");
00201                 message += T("                    sequence to determine the clone
-specific mutation pattern.\n");
00202                 message += T("\n");
00203                 message += T("Input format:\n");
00204                 message += T("Individual DNA sequences have to be saved in a plai
n text file (.txt), each. The sequence\n");
00205                 message += T("itself has to be written in the first line only, wi
th no figures (lower or capital case)\n");
00206                 message += T("except for A (adenine), C (cytidine), T (thymine),
G (guanine). The name of the txt-file\n");
00207                 message += T("will be used as clone name by FLER (example: the se
quence contained in file <clone1.txt>\n");
00208                 message += T("will be addressed as <clone1>).");
00209                 message += T("\n");
00210                 message += T("Output data:\n");
00211                 message += T("The user can choose whether to display only the pm-
directed hierarchy of HPCs, the groups\n");
00212                 message += T("of OTUs defining a certain HPC, or individual clona
l metatdata like mutated loci,\n");
00213                 message += T("mutation number etc. All output data will besaved i
n a file the user can determine\n");
00214                 message += T("interactively. It is recommended to use the extensi
on <.txt>.\n");
00215                 message += T("-------------------------------------------------
-----------------------------------------------------\n");
00216                                 AlertWindow::showMessageBox(AlertWindow::InfoIcon
, title, message, T("OK"));
00217                 break;
00218                     }
00219                     case basicfunction:
00220                     {
00221                 String title = T("Basic function of FLER");
00222                 String message = T("-----------------------------------------
-----------------------------------------------------------\n");
00223                                 message += T("FLER is the implementation of a mat
hematic/statistic algorithm to calculate the most\n");
00224                                 message += T("likely evolution of monoclonal oper
ational taxonomic units (OTU) driven by somatic\n");
00225                                 message += T("hypermutation. Thus, given a group
of OTUs defined by various mutation patterns of\n");
00226                                 message += T("a certain DNA-sequence (compared to
 an unmutated wildtype configuration (germline)),\n");
00227                                 message += T("FLER uses the number of equally mut
ated loci among subgroups of OTUs that share\n");
00228                                 message += T("different spectra of mutations as f
oundation for creation of hypothetical predecessor\n");
00229                                 message += T("clones (HPCs). Among the abundance
of HPCs the most probable HPCs are chosen to\n");
00230                                 message += T("recapitulate evolution from the wil
dtype configuration to each OTU. Probability is\n");
00231                                 message += T("assessed by calculation of the prob
ability measure \"pm\" for each HPC.\n");
00232                                 message += T("Input parameters of the pm-formula
are:\n");
00233                                 message += T("\n");
00234                                 message += T("r: number of OTUs in a OTU subgroup
 defining the considered HPC\n");
00235                                 message += T("i: nuber of mutations of an HPC (nu
mber of mutations shared by all clones of the\n");
00236                                 message += T("   HPC-defining subgroup)\n");
```

```
00237                                   message += T("z: number of reapeats of a HPC with
      the identical mutation patterns among the\n");
00238                                   message += T("   abundance of HPCs\n");
00239                                   message += T("t: total number of operational taxo
      nomic units (OTUs) / input clones\n");
00240                                   message += T("P: relative frequency of a consider
      ed mutation among the whole population of OTUs\n");
00241                                   message += T("\n");
00242                                   message += T("Unlike the neighbor joining (NJ) al
      gorithm, FLER calculates HPCs and considers the mutation\n");
00243                                   message += T("number as surrogate marker for elap
      sed time. Thus clonal interchange among different\n");
00244                                   message += T("populations of OTUs can also be ass
      essed by doing population-specific assays first,\n");
00245                                   message += T("complemented by an assay comprising
       all OTUs as one population. By comparing pm-values\n");
00246                                   message += T("of HPCs of different assays, interc
      hange among various populations rather than\n");
00247                                   message += T("population-contained evolution can
      be evaluated. Then, migration directions are\n");
00248                                   message += T("determined by the population affili
      ation of the majority of HPC-defining OTUs.\n");
00249                                   message += T("-----------------------------------
      ------------------------------------------------------------------\n");
00250 
00251                                   AlertWindow::showMessageBox(AlertWindow::InfoIcon
      , title, message, T("OK"));
00252 
00253                       }
00254                    break;
00255                           case about:
00256                           {
00257                                   String title = String(ProjectInfo::projectName);
00258                                   title += T(" Version ");
00259                                   title += String(ProjectInfo::versionString);
00260                   #if defined (RC_APPDATE_STR)
00261                   title += T(" (RC ");
00262                   title += String(RC_APPDATE_STR);
00263                   title += T(")");
00264                   #endif
00265                                   String message = T("-------------------------------
      ------------------------------------\n");
00266                   message += T("Follicular Lymphoma Evolution Reconstructor.\n");
00267                                   message += T("-----------------------------------
      -----------------------------\n");
00268                                   message += T("Coding:  ");
00269                                   message += RC_COMPANY_STR;
00270                                   message += T(", Technical University, Berlin.\n")
      ;
00271                                   message += T("Method: Martin Wartenberg, Technica
      l University, Munich.\n");
00272                                   message += T("Contact: moatlcombat@gmx.net\n");
00273                                   message += T("-----------------------------------
      -----------------------------\n");
00274                                   message += T("http://www.petervasil.net/fler\n");

00275                                   message += T("-----------------------------------
      -----------------------------\n");
00276                                   AlertWindow::showMessageBox(AlertWindow::InfoIcon
      , title, message, T("OK"));
00277                           }
00278                                   break;
00279               case web:
00280                   {
00281                       URL flerUrl(T("http://www.petervasil.net/fler"));
00282                       flerUrl.launchInDefaultBrowser();
00283                   }
```

```
00284                 break;
00285                         default:
00286                                 result = false;
00287                                 break;
00288                 }
00289                 return result;
00290         }
00291         //=============================================================================
      =======
00292 private:
00293         //=============================================================================
      =======
00294         DocumentWindow* m_mainWindow;
00295         ApplicationCommandManager* m_commandManager;
00296         TooltipWindow tooltipWindow;
00297 
00298         //=============================================================================
      =======
00300         MainComponent(const MainComponent&);
00302         const MainComponent& operator=(const MainComponent&);
00303 
00304     TextButton* m_btGl;
00305     TextButton* m_btKlone;
00306     TextButton* m_btCloneVsGl;
00307     TextButton* m_btRestart;
00308     TextButton* m_btSaveFile;
00309 
00310     String m_statusText;
00311     int m_seqLength;
00312 
00313     TextEditor* m_txSeqLen;
00314 
00315     CloneCalculations* m_cloneClaculations;
00316 
00317     PropertiesFile* m_props;
00318     String m_lastGlFolderPath;
00319     String m_lastClonesFolderPath;
00320     String m_lastSaveFolderPath;
00321 
00322     bool m_writePmHierarchy;
00323     bool m_writeClonalMetadata;
00324     bool m_writeClonalGroups;
00325 };
00326 
00327 #endif//_MAINCOMPONENT_H_
```

5.23 PVLogger.h File Reference

```
#include <sys/time.h>
#include <sys/types.h>
#include <stdlib.h>
#include <sys/timeb.h>
#include <time.h>
```

Classes

- class PVLogger

  Logger class for easy debugging.

Defines

- #define __PRETTY_FUNCTION__ __FUNCTION__
- #define PVL_DVAL T("DBG_VAL")
- #define PVL_DSCOPE T("DBG_SCOPE")
- #define PVL_VAL T("LOG_VAL")
- #define PVL_SCOPE T("LOG_SCOPE")
- #define PVL_DERROR T("DBG_ERROR")
- #define PVL_ERROR T("LOG_ERROR")
- #define DBG_VAL(message)
- #define DBG_SCOPE()
- #define DBG_ERROR(message)
- #define LOG_VAL(message) PVLogger::logMessage(PVL_VAL, __PRETTY_FUNCTION__-, T(#message), message)
- #define LOG_SCOPE() PVLogger::logMessage(PVL_SCOPE, __PRETTY_FUNCTION__)
- #define LOG_ERROR(message) PVLogger::outputError(PVL_ERROR, __PRETTY_-FUNCTION__, message)

5.23.1 Define Documentation

5.23.1.1 #define __PRETTY_FUNCTION__ __FUNCTION__

Definition at line 39 of file PVLogger.h.

5.23.1.2 #define DBG_ERROR(message)

Definition at line 262 of file PVLogger.h.

5.23.1.3 #define DBG_SCOPE()

Definition at line 260 of file PVLogger.h.

5.23.1.4 #define DBG_VAL(message)

Definition at line 258 of file PVLogger.h.

5.23.1.5 #define LOG_ERROR(message) PVLogger::outputError(PVL_ERROR, __PRETTY_FUNCTION__, message)

Definition at line 268 of file PVLogger.h.

5.23.1.6 #define LOG_SCOPE() PVLogger::logMessage(PVL_SCOPE, __PRETTY_FUNCTION__)

Definition at line 266 of file PVLogger.h.

5.23.1.7 #define LOG_VAL(message) PVLogger::logMessage(PVL_VAL, __PRETTY_FUNCTION__, T(#message), message)

Definition at line 264 of file PVLogger.h.

5.23.1.8 #define PVL_DERROR T("DBG_ERROR")

Definition at line 50 of file PVLogger.h.

5.23.1.9 #define PVL_DSCOPE T("DBG_SCOPE")

Definition at line 47 of file PVLogger.h.

5.23.1.10 #define PVL_DVAL T("DBG_VAL")

Definition at line 46 of file PVLogger.h.

5.23.1.11 #define PVL_ERROR T("LOG_ERROR")

Definition at line 51 of file PVLogger.h.

5.23.1.12 #define PVL_SCOPE T("LOG_SCOPE")

Definition at line 49 of file PVLogger.h.

5.23.1.13 #define PVL_VAL T("LOG_VAL")

Definition at line 48 of file PVLogger.h.

5.24 PVLogger.h

```
00001 /*=========================================================================
00002  * Copyright (C) 2010 Peter Vasil
00003  *
00004  * This program is free software; you can redistribute it and/or modify
00005  * it under the terms of the GNU General Public License as published by
00006  * the Free Software Foundation; either version 3 of the License, or
00007  * (at your option) any later version.
00008  *
00009  * This program is distributed in the hope that it will be useful,
00010  * but WITHOUT ANY WARRANTY; without even the implied warranty of
00011  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00012  * GNU General Public License for more details.
00013  *
00014  * You should have received a copy of the GNU General Public License
00015  * along with this program; if not, see <http://www.gnu.org/licenses/>.
00016  *
00017  =========================================================================*/
00018
00019 #ifndef _PVLOGGER_H_
00020 #define _PVLOGGER_H_
00021
00022 #ifdef _WIN32
00023 # define WIN32_LEAN_AND_MEAN
00024 # include <windows.h>
00025 # include <lmcons.h>
00026 # define sprintf sprintf_s
00027 # define USE_NEW_SECURE_TIME_FNS
00028 #else
00029 //# include <unistd.h>
00030 # include <sys/time.h>
00031 # include <sys/types.h>
00032 //# include <pwd.h>
00033 #endif
00034 #include <stdlib.h>
00035 #include <sys/timeb.h>
00036 #include <time.h>
00037
00038 #ifndef __GNUC__
00039 # define __PRETTY_FUNCTION__ __FUNCTION__
00040 #endif
00041
00042 #if defined __APPLE__
00043 # define UInt32 uint32
00044 #endif
00045
00046 #define PVL_DVAL                        T("DBG_VAL")
00047 #define PVL_DSCOPE                      T("DBG_SCOPE")
00048 #define PVL_VAL                         T("LOG_VAL")
00049 #define PVL_SCOPE                       T("LOG_SCOPE")
00050 #define PVL_DERROR                      T("DBG_ERROR")
00051 #define PVL_ERROR                       T("LOG_ERROR")
00052
00057 class PVLogger {
00058 private:
00059         PVLogger()
00060         {
00061         }
00062         ~PVLogger()
00063         {
00064         }
00065
00066 public:
00067         static void outputError (const tchar* logType, const char* function, cons
      t String& message)
00068         {
```

```
00069                 String tmpMsg;
00070                 tmpMsg
00071                 << logType
00072                 << T(" [")
00073                 << getCurrenTimeString()
00074                 << T("] : --> ")
00075                 << String(function)
00076                 << T(" \n")
00077                 << message
00078                 << T("\n");
00079                 Logger::outputDebugString(tmpMsg);
00080         }
00081         static void logMessage (const tchar* logType, const char* function, const
      String& name, const String& message)
00082         {
00083                 String tmpMsg;
00084                 tmpMsg
00085                 << logType
00086                 << T(" [")
00087                 << getCurrenTimeString()
00088                 << T("] : --> ")
00089                 << String(function)
00090                 << T(" \n")
00091                 << name
00092                 << T(" : ")
00093                 << message
00094                 << T("\n");// << T("<-------------------------");
00095                 Logger::outputDebugString(tmpMsg);
00096         }
00097         static void logMessage (const tchar* logType, const char* function, const
      String& name, int message)
00098         {
00099                 String tmpMsg;
00100                 tmpMsg
00101                 << logType
00102                 << T(" [")
00103                 << getCurrenTimeString()
00104                 << T("] : --> ")
00105                 << String(function)
00106                 << T(" \n")
00107                 << name
00108                 << T(" : ")
00109                 << message
00110                 << T("\n");// << T("<-------------------------");
00111                 Logger::outputDebugString(tmpMsg);
00112         }
00113         static void logMessage (const tchar* logType, const char* function, const
      String& name, float message)
00114         {
00115                 String tmpMsg;
00116                 tmpMsg
00117                 << logType
00118                 << T(" [")
00119                 << getCurrenTimeString()
00120                 << T("] : --> ")
00121                 << String(function)
00122                 << T(" \n")
00123                 << name
00124                 << T(" : ")
00125                 << message
00126                 << T("\n");// << T("<-------------------------");
00127                 Logger::outputDebugString(tmpMsg);
00128         }
00129         static void logMessage (const tchar* logType, const char* function, const
      String& name, double message)
00130         {
00131                 String tmpMsg;
```

```
00132                 tmpMsg
00133                 << logType
00134                 << T(" [")
00135                 << getCurrenTimeString()
00136                 << T("] : --> ")
00137                 << String(function)
00138                 << T(" \n")
00139                 << name
00140                 << T(" : ")
00141                 << message
00142                 << T("\n");// << T("<--------------------------");
00143                 Logger::outputDebugString(tmpMsg);
00144         }
00145         static void logMessage (const tchar* logType, const char* function, const
       String& name, bool message)
00146         {
00147                 String tmpMsg;
00148                 tmpMsg
00149                 << logType
00150                 << T(" [")
00151                 << getCurrenTimeString()
00152                 << T("] --> ")
00153                 << String(function)
00154                 << T(" \n")
00155                 << name
00156                 << T(" : ")
00157                 << (message ? T("true") : T("false"))
00158                 << T("\n");// << T("<--------------------------");
00159                 Logger::outputDebugString(tmpMsg);
00160         }
00161         static void logMessage (const tchar* logType, const char* function)
00162         {
00163                 String tmpMsg;
00164                 tmpMsg
00165                 << logType
00166                 << T(" [")
00167                 << getCurrenTimeString()
00168                 << T("] --> ")
00169                 << String(function)
00170                 << T(" \n");
00171                 Logger::outputDebugString(tmpMsg);
00172         }
00173 private:
00174         /*
00175          * Get current time in milliseconds with high resolution.
00176          * Copied and modified from the implementation in Juce without high resul
      tion.
00177          * The high resolution implementation is missing there.
00178          */
00179         static int64 currentTimeMillisHiRes()
00180         {
00181                 static uint32 lastCounterResult = 0xffffffff;
00182                 static int64 correction = 0;
00183
00184                 const uint32 now = (uint32)Time::getMillisecondCounterHiRes();
00185
00186                 // check the counter hasn't wrapped (also triggered the first tim
      e this function is called)
00187                 if (now < lastCounterResult)
00188                 {
00189                         // double-check it's actually wrapped, in case multi-cpu
      machines have timers that drift a bit.
00190                         if (lastCounterResult == 0xffffffff || now < lastCounterR
      esult - 10)
00191                         {
00192                                 // get the time once using normal library calls,
      and store the difference needed to
```

```
00193                                   // turn the millisecond counter into a real time.
00194 #if JUCE_WIN32
00195                                   struct _timeb t;
00196 #ifdef USE_NEW_SECURE_TIME_FNS
00197                                   _ftime_s (&t);
00198 #else
00199                                   _ftime (&t);
00200 #endif
00201                                   correction = (((int64) t.time) * 1000 + t.millitm
      ) - now;
00202 #else
00203                                   struct timeval tv;
00204                                   struct timezone tz;
00205                                   gettimeofday (&tv, &tz);
00206                                   correction = (((int64) tv.tv_sec) * 1000 + tv.tv_
      usec / 1000) - now;
00207 #endif
00208                           }
00209                   }
00210
00211                   lastCounterResult = now;
00212
00213                   return correction + now;
00214           }
00215
00216           static void getCurrentTime(int64 seconds, struct tm& currTime)
00217           {
00218                   time_t now = (time_t)(seconds);
00219
00220 #if JUCE_WIN32
00221                   localtime_s (&currTime, &now);
00222 #else
00223                   localtime_r (&now, &currTime);
00224 #endif
00225           }
00226           static const String getCurrenTimeString()
00227           {
00228                   int64 currTimeMillis = currentTimeMillisHiRes();
00229                   struct tm currTime;
00230                   getCurrentTime(currTimeMillis/1000,currTime);
00231
00232                   char buffer[20];
00233                   sprintf(buffer, "%02d:%02d:%02d.%03d", currTime.tm_hour, currTime
      .tm_min, currTime.tm_sec, (int)(currTimeMillis%1000));
00234                   return (String)buffer;
00235           }
00236
00237 };
00238
00239 // ------------------------------------------------------------------------------
00240 // LOG Macros
00241 // ------------------------------------------------------------------------------
00242 #if defined (_DEBUG)
00243
00244 # define DBG_VAL(message)                                   PVLogger::logMessage(PVL_
      DVAL, __PRETTY_FUNCTION__, T(#message), message)
00245
00246 # define DBG_SCOPE()                                       PVLogger::logMessage(PVL_
      DSCOPE, __PRETTY_FUNCTION__)
00247
00248 # define DBG_ERROR(message)                                PVLogger::outputError(PVL
      _DERROR, __PRETTY_FUNCTION__, message)
00249
00250 # define LOG_ERROR(message)                                PVLogger::outputError(PVL
      _ERROR, __PRETTY_FUNCTION__, message)
00251
```

```
00252 # define LOG_VAL(message)                                    PVLogger::logMessage(PVL_
      VAL, __PRETTY_FUNCTION__, T(#message), message)
00253
00254
00255 // ---------------------------------------------------------------------------
00256 #else
00257
00258 # define DBG_VAL(message)
00259
00260 # define DBG_SCOPE()
00261
00262 # define DBG_ERROR(message)
00263
00264 # define LOG_VAL(message)                                    PVLogger::logMessage(PVL_
      VAL, __PRETTY_FUNCTION__, T(#message), message)
00265
00266 # define LOG_SCOPE()                                         PVLogger::logMessage(PVL_
      SCOPE, __PRETTY_FUNCTION__)
00267
00268 # define LOG_ERROR(message)                                  PVLogger::outputError(PVL
      _ERROR, __PRETTY_FUNCTION__, message)
00269
00270 #endif
00271
00272 #if defined __APPLE__
00273 # undef UInt32
00274 #endif
00275
00276 #if defined _WIN32
00277 # undef sprintf
00278 #endif
00279
00280 #endif // _PVLOGGER_H_
```