Evaporative Cooling

1.0

Generated by Doxygen 1.7.6.1

Fri Feb 10 2012 05:02:58

Contents

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Class Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 insilico Namespace Reference

Classes

- class is_classified
- class do_to_upper
- class do_to_lower

Functions

- template<typename InputIt , typename OutputIt , typename Comp >
  void best_n (InputIt begin, InputIt end, OutputIt out, size_t n, Comp comp)

  Get the best n values with ties keeping same original order.
- template<typename stringT >
  stringT trim_left (const stringT &s, const std::locale &loc=std::locale())
- template<typename stringT >
  stringT trim_right (const stringT &s, const std::locale &loc=std::locale())
- template<typename stringT >
  stringT trim (const stringT &s, const std::locale &loc=std::locale())
- template<typename Container , typename stringT >
  void split (Container &cont, const stringT &s, const std::locale &loc=std::locale())
- template<typename Container , typename stringT >
  void split (Container &cont, const stringT &s, const stringT &delim)
- template<typename Container , typename stringT , typename Pred >
  void split_if (Container &cont, const stringT &s, const Pred &pred)
- template<typename It , typename stringT >
  stringT join (const It &begin, const It &end, const stringT &delim)
- template<typename stringT >
  stringT to_upper (const stringT &str, const std::locale &loc=std::locale())
- template<typename stringT >
  stringT to_lower (const stringT &str, const std::locale &loc=std::locale())

- std::string trim_left (const char ∗s, const std::locale &loc=std::locale())
- std::wstring trim_left (const wchar_t ∗s, const std::locale &loc=std::locale())
- std::string trim_right (const char ∗s, const std::locale &loc=std::locale())
- std::wstring trim_right (const wchar_t ∗s, const std::locale &loc=std::locale())
- std::string trim (const char ∗s, const std::locale &loc=std::locale())
- std::wstring trim (const wchar_t ∗s, const std::locale &loc=std::locale())
- template<typename Container >
  void split (Container &cont, const char ∗s, const std::locale &loc=std::locale())
- template<typename Container >
  void split (Container &cont, const wchar_t ∗s, const std::locale &loc=std::locale())
- template<typename Container >
  void split (Container &cont, const std::string &s, const char ∗delim)
- template<typename Container >
  void split (Container &cont, const char ∗s, const std::string &delim)
- template<typename Container >
  void split (Container &cont, const char ∗s, const char ∗delim)
- template<typename Container >
  void split (Container &cont, const std::wstring &s, const wchar_t ∗delim)
- template<typename Container >
  void split (Container &cont, const wchar_t ∗s, const std::wstring &delim)
- template<typename Container >
  void split (Container &cont, const wchar_t ∗s, const wchar_t ∗delim)
- template<typename Container , typename Pred >
  void split_if (Container &cont, const char ∗s, const Pred &pred)
- template<typename Container , typename Pred >
  void split_if (Container &cont, const wchar_t ∗s, const Pred &pred)
- template<typename It >
  std::string join (const It &begin, const It &end, const char ∗delim)
- template<typename It >
  std::wstring join (const It &begin, const It &end, const wchar_t ∗delim)
- std::string to_upper (const char ∗s, const std::locale &loc=std::locale())
- std::wstring to_upper (const wchar_t ∗s, const std::locale &loc=std::locale())
- std::string to_lower (const char ∗s, const std::locale &loc=std::locale())
- std::wstring to_lower (const wchar_t ∗s, const std::locale &loc=std::locale())
- template<typename T >
  std::string get_bits (T value)
- template<typename T >
  std::string zeroPadNumber (T num, int padSize)

5.1.1 Function Documentation

5.1.1.1 template<typename InputIt , typename OutputIt , typename Comp > void insilico::best_n (InputIt *begin,* InputIt *end,* OutputIt *out,* size_t *n,* Comp *comp*)

Get the best n values with ties keeping same original order.

Parameters

`in`	*begin*	iterator of the beginning of a input container
`in`	*end*	iterator of the end of a input container
`out`	*out*	iterator of the beginning of a output container
`in`	*size*	best n value
`in`	*comp*	compare functor

Returns

path/filename without extension

Definition at line 30 of file best_n.h.

5.1.1.2 template<typename T > std::string **insilico::get_bits** (T *value*)

Definition at line 324 of file StringUtils.h.

5.1.1.3 template<typename It , typename stringT > stringT **insilico::join** (const It & *begin,* const It & *end,* const stringT & *delim*)

Definition at line 198 of file StringUtils.h.

5.1.1.4 template<typename It > std::string **insilico::join** (const It & *begin,* const It & *end,* const char ∗ *delim*) `[inline]`

Definition at line 300 of file StringUtils.h.

5.1.1.5 template<typename It > std::wstring **insilico::join** (const It & *begin,* const It & *end,* const wchar_t ∗ *delim*) `[inline]`

Definition at line 304 of file StringUtils.h.

5.1.1.6 template<typename Container , typename stringT > void **insilico::split** (Container & *cont,* const stringT & *s,* const std::locale & *loc =* `std::locale()`) `[inline]`

Definition at line 148 of file StringUtils.h.

5.1.1.7 template<typename Container , typename stringT > void **insilico::split** (Container & *cont,* const stringT & *s,* const stringT & *delim*)

Definition at line 156 of file StringUtils.h.

5.1.1.8 template<typename Container > void insilico::split (Container & *cont*, const char * *s*, const std::locale & *loc* = `std::locale()`) `[inline]`

Definition at line 258 of file StringUtils.h.

5.1.1.9 template<typename Container > void insilico::split (Container & *cont*, const wchar_t * *s*, const std::locale & *loc* = `std::locale()`) `[inline]`

Definition at line 263 of file StringUtils.h.

5.1.1.10 template<typename Container > void insilico::split (Container & *cont*, const std::string & *s*, const char * *delim*) `[inline]`

Definition at line 268 of file StringUtils.h.

5.1.1.11 template<typename Container > void insilico::split (Container & *cont*, const char * *s*, const std::string & *delim*) `[inline]`

Definition at line 272 of file StringUtils.h.

5.1.1.12 template<typename Container > void insilico::split (Container & *cont*, const char * *s*, const char * *delim*) `[inline]`

Definition at line 276 of file StringUtils.h.

5.1.1.13 template<typename Container > void insilico::split (Container & *cont*, const std::wstring & *s*, const wchar_t * *delim*) `[inline]`

Definition at line 280 of file StringUtils.h.

5.1.1.14 template<typename Container > void insilico::split (Container & *cont*, const wchar_t * *s*, const std::wstring & *delim*) `[inline]`

Definition at line 284 of file StringUtils.h.

5.1.1.15 template<typename Container > void insilico::split (Container & *cont*, const wchar_t * *s*, const wchar_t * *delim*) `[inline]`

Definition at line 288 of file StringUtils.h.

5.1.1.16 template<typename Container , typename stringT , typename Pred > void insilico::split_if (Container & *cont*, const stringT & *s*, const Pred & *pred*)

Definition at line 178 of file StringUtils.h.

5.1.1.17 template<typename Container , typename Pred > void insilico::split_if (Container & *cont*, const char * *s*, const Pred & *pred*) `[inline]`

Definition at line 292 of file StringUtils.h.

5.1.1.18 template<typename Container , typename Pred > void insilico::split_if (Container & *cont*, const wchar_t * *s*, const Pred & *pred*) `[inline]`

Definition at line 296 of file StringUtils.h.

5.1.1.19 template<typename stringT > stringT insilico::to_lower (const stringT & *str*, const std::locale & *loc* = `std::locale()`)

Definition at line 224 of file StringUtils.h.

5.1.1.20 std::string insilico::to_lower (const char * *s*, const std::locale & *loc* = `std::locale()`) `[inline]`

Definition at line 315 of file StringUtils.h.

5.1.1.21 std::wstring insilico::to_lower (const wchar_t * *s*, const std::locale & *loc* = `std::locale()`) `[inline]`

Definition at line 319 of file StringUtils.h.

5.1.1.22 template<typename stringT > stringT insilico::to_upper (const stringT & *str*, const std::locale & *loc* = `std::locale()`)

Definition at line 214 of file StringUtils.h.

5.1.1.23 std::string insilico::to_upper (const char * *s*, const std::locale & *loc* = `std::locale()`) `[inline]`

Definition at line 307 of file StringUtils.h.

5.1.1.24 std::wstring insilico::to_upper (const wchar_t * *s*, const std::locale & *loc* = `std::locale()`) `[inline]`

Definition at line 311 of file StringUtils.h.

5.1.1.25 template<typename stringT > stringT insilico::trim (const stringT & *s*, const std::locale & *loc* = `std::locale()`)

Definition at line 123 of file StringUtils.h.

5.1.1.26 std::string insilico::trim (const char * *s*, const std::locale & *loc* = `std::locale()`) `[inline]`

Definition at line 249 of file StringUtils.h.

5.1.1.27 std::wstring insilico::trim (const wchar_t * *s*, const std::locale & *loc* = `std::locale()`) `[inline]`

Definition at line 253 of file StringUtils.h.

5.1.1.28 template<typename stringT > stringT insilico::trim_left (const stringT & *s*, const std::locale & *loc* = `std::locale()`)

Definition at line 101 of file StringUtils.h.

5.1.1.29 std::string insilico::trim_left (const char * *s*, const std::locale & *loc* = `std::locale()`) `[inline]`

Definition at line 233 of file StringUtils.h.

5.1.1.30 std::wstring insilico::trim_left (const wchar_t * *s*, const std::locale & *loc* = `std::locale()`) `[inline]`

Definition at line 237 of file StringUtils.h.

5.1.1.31 template<typename stringT > stringT insilico::trim_right (const stringT & *s*, const std::locale & *loc* = `std::locale()`)

Definition at line 112 of file StringUtils.h.

5.1.1.32 std::string insilico::trim_right (const char * *s*, const std::locale & *loc* = `std::locale()`) `[inline]`

Definition at line 241 of file StringUtils.h.

5.1.1.33 std::wstring insilico::trim_right (const wchar_t * *s*, const std::locale & *loc* = `std::locale()`) `[inline]`

Definition at line 245 of file StringUtils.h.

5.1.1.34 template<typename T > std::string insilico::zeroPadNumber (T *num*, int *padSize*)

Definition at line 333 of file StringUtils.h.

Chapter 6

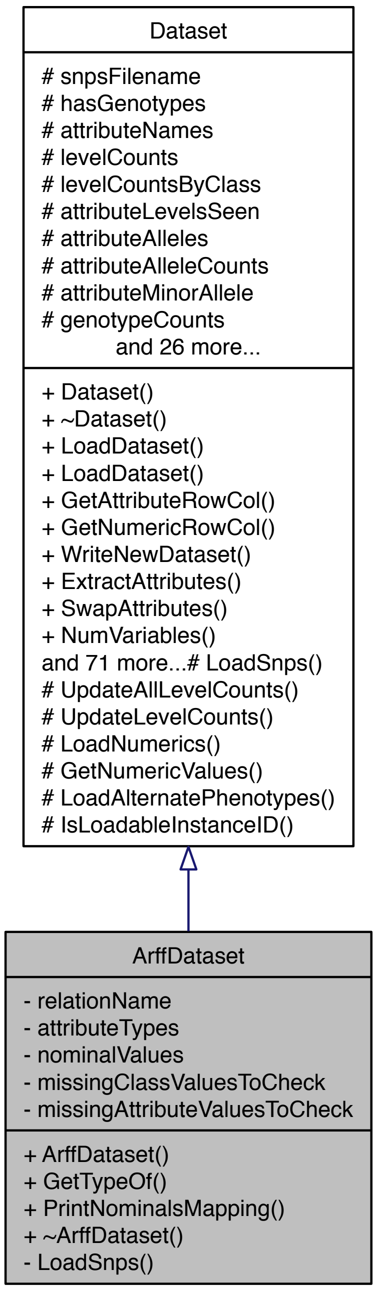
Class Documentation

6.1 ArffDataset Class Reference

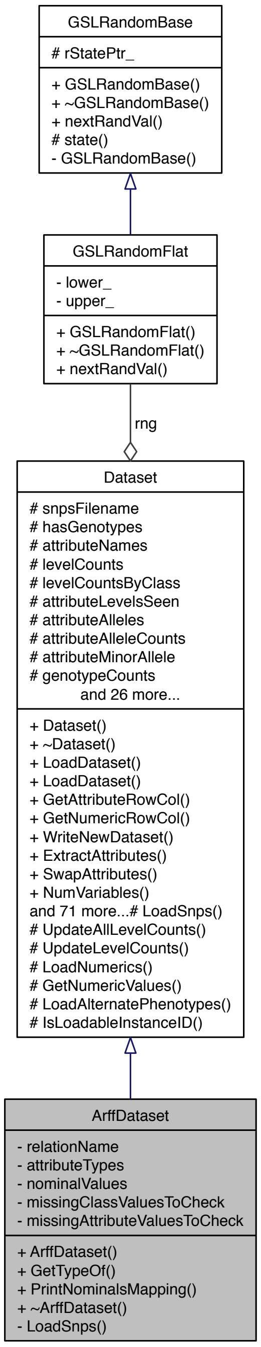
ARFF file format reader.

```
#include <ArffDataset.h>
```

Inheritance diagram for ArffDataset:



Collaboration diagram for ArffDataset:



Public Member Functions

- ArffDataset ()
- ArffAttributeType GetTypeOf (unsigned int columnIndex)
- void PrintNominalsMapping ()
- ∼ArffDataset ()

Private Member Functions

- bool LoadSnps (std::string filename)

  Load SNPs from file using the data set filename.

Private Attributes

- std::string relationName

  ARFF relation name.
- std::vector< ArffAttributeType > attributeTypes

  vector of attribute types
- std::map< std::string, std::vector< std::string > > nominalValues

  map of attribute names to valid nominal values
- std::vector< std::string > missingClassValuesToCheck

  missing class values
- std::vector< std::string > missingAttributeValuesToCheck

  missing attribute values

6.1.1 Detailed Description

ARFF file format reader.

`http://www.cs.waikato.ac.nz/ml/weka/arff.html`

See also

Dataset

Author

Bill White

Version

1.0

Contact: `bill.c.white@gmail.com` Created on: 2/24/11

Definition at line 38 of file ArffDataset.h.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 ArffDataset::ArffDataset ( )

Definition at line 25 of file ArffDataset.cpp.

6.1.2.2 ArffDataset::∼ArffDataset ( ) `[inline]`

Definition at line 54 of file ArffDataset.h.

6.1.3 Member Function Documentation

6.1.3.1 ArffAttributeType ArffDataset::GetTypeOf (unsigned int *columnIndex*)

Definition at line 374 of file ArffDataset.cpp.

6.1.3.2 bool ArffDataset::LoadSnps (std::string *filename*) `[private, virtual]`

Load SNPs from file using the data set filename.

Parameters

`in`	*filename*	SNPs filename
`in`	*deRecodeA*	perform a recodeA operation after reading raw data?

Returns

success

------------ Beginning of private methods ------------------ Open the data file and read line-by-line

Detect the class type

Reimplemented from Dataset.

Definition at line 31 of file ArffDataset.cpp.

6.1.3.3 void ArffDataset::PrintNominalsMapping ( )

Definition at line 381 of file ArffDataset.cpp.

6.1.4 Member Data Documentation

6.1.4.1 std::vector<ArffAttributeType> ArffDataset::attributeTypes `[private]`

vector of attribute types

Definition at line 61 of file ArffDataset.h.

6.1.4.2 std::vector<std::string> ArffDataset::missingAttributeValuesToCheck `[private]`

missing attribute values

Definition at line 68 of file ArffDataset.h.

6.1.4.3 std::vector<std::string> ArffDataset::missingClassValuesToCheck `[private]`

missing class values

Definition at line 66 of file ArffDataset.h.

6.1.4.4 std::map<std::string, std::vector<std::string> > ArffDataset::nominalValues `[private]`

map of attribute names to valid nominal values

Definition at line 63 of file ArffDataset.h.

6.1.4.5 std::string ArffDataset::relationName `[private]`

ARFF relation name.

Definition at line 59 of file ArffDataset.h.

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

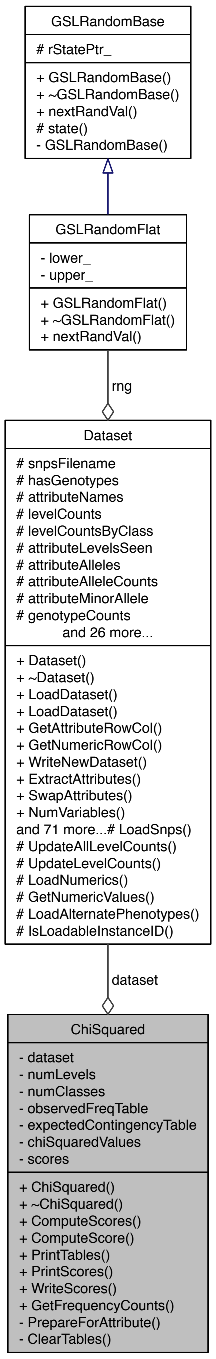
- src/library/ArffDataset.h
- src/library/ArffDataset.cpp

6.2 ChiSquared Class Reference

Chi-squared attribute ranking algorithm.

`#include <ChiSquared.h>`

Collaboration diagram for ChiSquared:



Public Member Functions

- ChiSquared (Dataset ∗ds)

  Construct an chi-squared algorithm object.
- ∼ChiSquared ()
- const std::vector< std::pair < double, double > > & ComputeScores ()

  For each attribue, calculate chi-squared and associated p-value.
- std::pair< double, double > ComputeScore (unsigned int index)

  For the attribue at the specified index, calculate the chi-squared and associated p-value.
- void PrintTables ()

  Print calculation tables.
- void PrintScores (std::ofstream &outStream, unsigned int topN=0)

  Print the scores to a stream.
- void WriteScores (std::string outFilename, unsigned int topN=0)

  Print the scores to a stream.
- std::vector< std::vector < double > > GetFrequencyCounts ()

  Get the observed frequencies table as a vector of vector of doubles.

Private Member Functions

- void PrepareForAttribute (unsigned int attributeIndex)

  Private method to setup the chi-squared contingency tables for a particular attribute.
- void ClearTables ()

  Clear calculation tables.

Private Attributes

- Dataset ∗ dataset

  pointer to a Dataset object
- unsigned int numLevels

  number of levels in the attributes
- unsigned int numClasses

  number of classes in the instances
- std::vector< std::vector < double > > observedFreqTable

  observed frequencies
- std::vector< std::vector < double > > expectedContingencyTable
- std::vector< std::vector < double > > chiSquaredValues

  chi squared computed values
- std::vector< std::pair< double, double > > scores

  chi-squared value, p-value for each attribute

6.2.1 Detailed Description

Chi-squared attribute ranking algorithm.

ChiSquared algorithm interface. For performing chi-squared tests of association between an attribute and its class across all instances in a data set.

Author

Bill White

Version

1.0

Contact: `bill.c.white@gmail.com` Created on: 6/15/05

Definition at line 25 of file ChiSquared.h.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 ChiSquared::ChiSquared (Dataset $*$ *ds*)

Construct an chi-squared algorithm object.

Parameters

`in`	*ds*	pointer to a Dataset object

Definition at line 21 of file ChiSquared.cpp.

6.2.2.2 ChiSquared::∼ChiSquared ( )

Definition at line 31 of file ChiSquared.cpp.

6.2.3 Member Function Documentation

6.2.3.1 void ChiSquared::ClearTables ( ) `[private]`

Clear calculation tables.

Definition at line 230 of file ChiSquared.cpp.

6.2.3.2 pair< double, double > ChiSquared::ComputeScore (unsigned int *index*)

For the attribue at the specified index, calculate the chi-squared and associated p-value.

Return as a pair.

Parameters

`in`	*index*	index into the attributes of the data set

Returns

pairs of chi-squared score and associated p-value for the attribute

Definition at line 46 of file ChiSquared.cpp.

6.2.3.3 const vector< pair< double, double > > & **ChiSquared::ComputeScores** ( )

For each attribue, calculate chi-squared and associated p-value.

Return in a vector of pairs indexed by attribute index.

Returns

vector of pairs of chi-squared scores and associated p-values

Definition at line 34 of file ChiSquared.cpp.

6.2.3.4 std::vector<std::vector<double> > **ChiSquared::GetFrequencyCounts** ( ) `[inline]`

Get the observed frequencies table as a vector of vector of doubles.

Definition at line 62 of file ChiSquared.h.

6.2.3.5 void **ChiSquared::PrepareForAttribute** (unsigned int *attributeIndex*) `[private]`

Private method to setup the chi-squared contingency tables for a particular attribute.

Parameters

`in`	*attribute-Index*	attribute index

Definition at line 209 of file ChiSquared.cpp.

6.2.3.6 void **ChiSquared::PrintScores** (std::ofstream & *outStream*, unsigned int *topN* = `0`)

Print the scores to a stream.

Parameters

`in`	*outStream*	reference to an output stream
`in`	*topN*	top number of attributes to print

Definition at line 176 of file ChiSquared.cpp.

6.2.3.7 void **ChiSquared::PrintTables** ( )

Print calculation tables.

Definition at line 145 of file ChiSquared.cpp.

6.2.3.8 void **ChiSquared::WriteScores** (std::string *outFilename*, unsigned int *topN* = `0`)

Print the scores to a stream.

Parameters

`in`	*outFilename*	filename to write scores to
`in`	*topN*	top number of attributes to print

Definition at line 193 of file ChiSquared.cpp.

6.2.4 Member Data Documentation

6.2.4.1 std::vector<std::vector<double> > **ChiSquared::chiSquaredValues** `[private]`

chi squared computed values

Definition at line 86 of file ChiSquared.h.

6.2.4.2 Dataset* **ChiSquared::dataset** `[private]`

pointer to a Dataset object

Definition at line 76 of file ChiSquared.h.

6.2.4.3 std::vector<std::vector<double> > **ChiSquared::expectedContingencyTable** `[private]`

Definition at line 84 of file ChiSquared.h.

6.2.4.4 unsigned int **ChiSquared::numClasses** `[private]`

number of classes in the instances

Definition at line 80 of file ChiSquared.h.

6.2.4.5 unsigned int **ChiSquared::numLevels** `[private]`

number of levels in the attributes

Definition at line 78 of file ChiSquared.h.

6.2.4.6 std::vector<std::vector<double> > **ChiSquared::observedFreqTable** `[private]`

observed frequencies

Definition at line 82 of file ChiSquared.h.

6.2.4.7 std::vector<std::pair<double, double> > **ChiSquared::scores** `[private]`

chi-squared value, p-value for each attribute

Definition at line 88 of file ChiSquared.h.

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

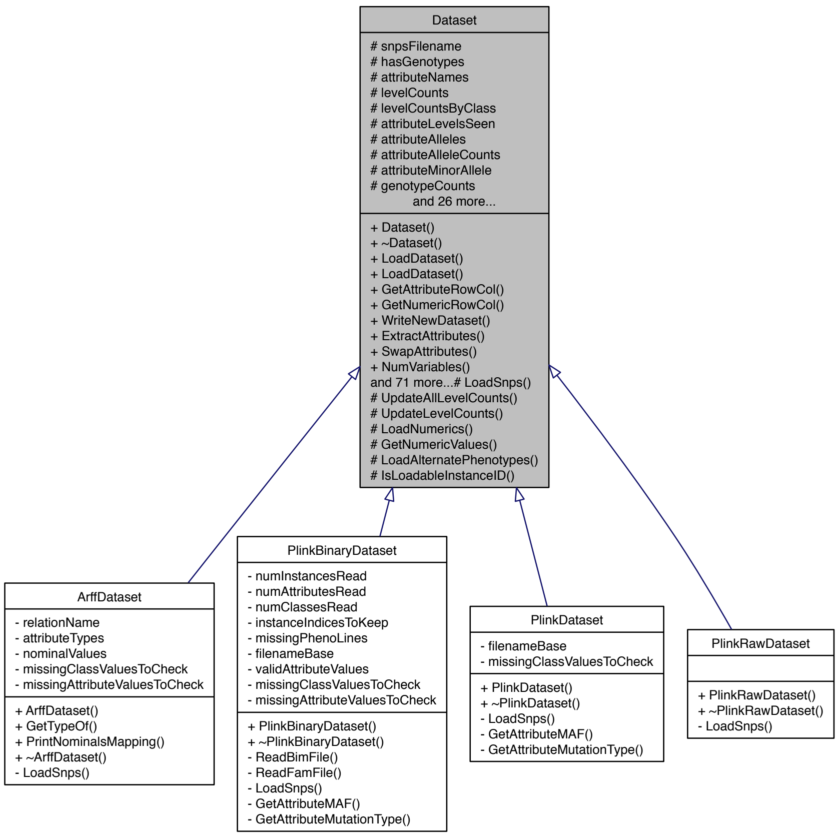
- src/library/ChiSquared.h
- src/library/ChiSquared.cpp

6.3 Dataset Class Reference

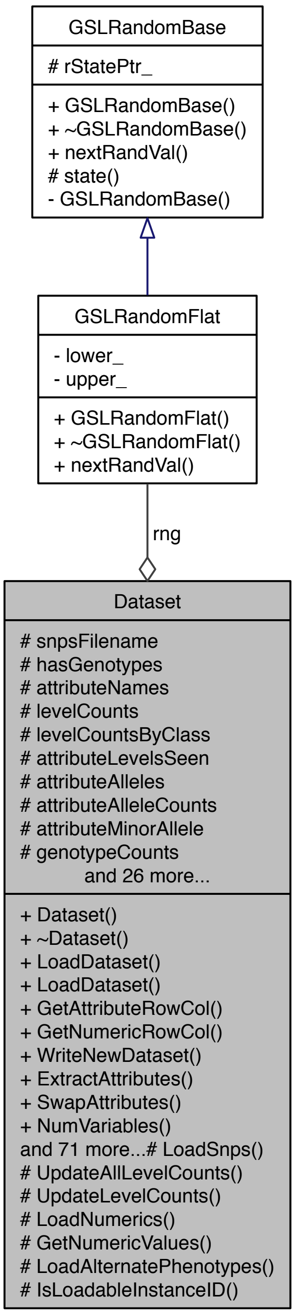
Base class for collections of instances containing attributea and class.

`#include <Dataset.h>`

Inheritance diagram for Dataset:



Collaboration diagram for Dataset:



Public Member Functions

- Dataset ()

  Construct a default data set.
- virtual ∼Dataset ()

  Destruct all dynamically allocated memory.
- bool LoadDataset (std::string snpsFilename, std::string numericsFilename, std::string altPhenoFilename, std::vector< std::string > ids)

  Load the dataset from files passed as parameters.
- bool LoadDataset (DgeData ∗dgeData)

  Load the dataset from DGE data.
- bool GetAttributeRowCol (unsigned int row, unsigned int col, AttributeLevel &attrVal)

  Get the attribute value at row, column.
- bool GetNumericRowCol (unsigned int row, unsigned int col, NumericLevel &numVal)

  Get the numeric value at row, column.
- bool WriteNewDataset (std::string newDatasetFilename, OutputDatasetType outputDatasetType)

  Write the dataset to a new filename, respecting masked attributes and numerics and class/phenotype data type.
- bool ExtractAttributes (std::string scoresFilename, unsigned int topN, std::string newDatasetFilename)

  Extracts top N attributes based on a file of attribute scores and writes a new dataset.
- bool SwapAttributes (unsigned int a1, unsigned int a2)

  Swap two attributes/columns in the dataset.
- unsigned int NumVariables ()

  Return the number of discrete plus continuous variables in the data set.
- std::vector< std::string > GetVariableNames ()

  Returns the names of discrete and continuous variables in the data set.
- virtual unsigned int NumInstances ()

  Returns the number of instances in the data set.
- DatasetInstance ∗ GetInstance (unsigned int index)

  Returns a pointer to a dataset instance selected by index.
- DatasetInstance ∗ GetRandomInstance ()

  Returns a pointer to a randomly chosen data set instance.
- std::vector< std::string > GetInstanceIds ()

  Get all instance IDs.
- bool GetInstanceIndexForID (std::string ID, unsigned int &instanceIndex)

  Get the instance index from the instance ID.
- virtual unsigned int NumAttributes ()

  Return the number of unmasked discrete attributes in the data set.
- std::vector< std::string > GetAttributeNames ()

  Return the discrete (SNP) attribute names.

- bool GetAttributeValues (unsigned int attributeIndex, std::vector< AttributeLevel > &attributeValues)

  Loads the referenced vector with an attribute's values (column).
- bool GetAttributeValues (std::string attributeName, std::vector< AttributeLevel > &attributeValues)

  Loads the referenced vector with an attribute's values (column) from the dataset.
- std::string GetSnpsFilename ()

  Get the filename SNPs were read from.
- unsigned int GetAttributeIndexFromName (std::string attributeName)

  Looks up original attribute index from attribute name.
- bool HasGenotypes ()

  Does the data set have genotype variables?
- AttributeLevel GetAttribute (unsigned instanceIndex, std::string name)

  Get attribute value for attribute name at instance index.
- virtual std::pair< char, double > GetAttributeMAF (unsigned int attributeIndex)

  Get attribute minor allele and frequency.
- virtual AttributeMutationType GetAttributeMutationType (unsigned int attribute-Index)

  Get attribute mutation type.
- bool GetIntForGenotype (std::string genotype, AttributeLevel &newAttr)

  Get integer value for string genotype.
- unsigned int NumLevels (unsigned int index)

  Returns the number of levels in a given attribute index.
- unsigned int NumNumerics ()

  Return the number of unmasked discrete attributes in the data set.
- std::vector< std::string > GetNumericsNames ()

  Return the numeric attribute names.
- std::pair< double, double > GetMinMaxForNumeric (unsigned int numericIdx)

  Get the minimum and maximum values for a numeric at index.
- double GetMeanForNumeric (unsigned int numericIdx)

  Get the mean/average of numeric at index.
- bool HasNumerics ()

  Does the data set have numeric variables? setter/getter.
- void HasNumerics (bool setHasNumerics)
- NumericLevel GetNumeric (unsigned int instanceIndex, std::string name)

  Get numeric value for numeric name at instance index.
- bool GetNumericValues (std::string numericName, std::vector< NumericLevel > &numericValues)

  Loads the referenced vector with a numeric's values (column) from the dataset.
- std::string GetNumericsFilename ()

  Get the filename numerics were read from.
- unsigned int GetNumericIndexFromName (std::string numericName)

  Looks up original numeric index from numeric name.
- unsigned int NumClasses ()

Get the number of classes in the data set.

- unsigned int GetClassColumn ()

  Get the class column as read from the file.
- bool GetClassValues (std::vector< ClassLevel > &classValues)

  Loads the referenced vector with the dataset's class labels.
- const std::map< ClassLevel, std::vector< unsigned int > > & GetClassIndexes ()

  Get a map from class levels to a vector of instance indices.
- bool HasAlternatePhenotypes ()

  Does the data set have alternate phenotypes loaded?
- void HasAlternatePhenotypes (bool setHasAlternatePhenotypes)
- std::string GetAlternatePhenotypesFilename ()

  Get the alternate phenotype filename.
- bool HasContinuousPhenotypes ()
- std::pair< double, double > GetMinMaxForContinuousPhenotype ()

  Get the minumum and maximum values for the continuous phenotype.
- void Print ()

  Print the entire data set in compact format.
- void PrintRecodeMap (std::vector< std::map< unsigned int, unsigned int > > recodeMap)

  Print the passed recode map to stdout.
- void PrintStats ()

  Print basic statstics abou the data set - discrete/SNPs only.
- void PrintNumericsStats ()

  Print statistics about the data set including numerics.
- void PrintStatsSimple ()

  Print very simple statistics abou the data set with no formatting.
- void PrintClassIndexInfo ()

  Print class index information.
- void PrintMissingValuesStats ()

  Print missing value statistics.
- void PrintLevelCounts ()

  Prit attribute level counts.
- void WriteLevelCounts (std::string levelsFilename)

  Write attribute level counts to a text file.
- void PrintAttributeLevelsSeen ()

  Print unique attribute levels seen.
- bool MaskRemoveVariable (std::string variableName)

  Removes the variable name from consideration in any data set operations.
- bool MaskRemoveVariableType (std::string variableName, AttributeType varType)

  Removes the attribute name from consideration in any data set operations.
- bool MaskSearchVariableType (std::string variableName, AttributeType attrType)

Determines if the named variable is in the current masked data set.
- bool MaskIncludeAllAttributes (AttributeType attrType)

  Mark all attributes for inclusion in data set operations.
- std::vector< unsigned int > MaskGetAttributeIndices (AttributeType attrType)

  Return a vector of all the attribute indices under consideration.
- const std::map< std::string, unsigned int > & MaskGetAttributeMask (AttributeType attrType)

  Return a map of attribute name to attribute index of attributes to include.
- std::vector< std::string > MaskGetAllVariableNames ()

  Return a vector of all the variable names under consideration.
- bool MaskRemoveInstance (std::string instanceId)

  Removes the instance from consideration in any data set operations.
- bool MaskSearchInstance (std::string instanceId)

  Determines if the names Instance is in the current masked dataaset.
- bool MaskIncludeAllInstances ()

  Mark all instances for inclusion in algorithms.
- std::vector< unsigned int > MaskGetInstanceIndices ()

  Return a vector of all the instance indices under consideration.
- std::vector< std::string > MaskGetInstanceIds ()

  Return a vector of all the instance ids under consideration.
- const std::map< std::string, unsigned int > & MaskGetInstanceMask ()

  Return a map of instance name to instance index of instances to include.
- bool MaskPushAll ()

  Save the current masks for later restore.
- bool MaskPopAll ()

  Restore the masks previously pushed.
- bool MaskWriteNewDataset (std::string newDatasetFilename)

  Saved the unmasked attributes as a tab-delimited text file.
- void PrintMaskStats ()

  Print mask statistics.
- void RunSnpDiagnosticTests (std::string logFilename, double globalGenotypeThreshold=0.01, unsigned int cellThreshold=5)

  Perform and report SNP diagnostic test information.
- bool CheckHardyWeinbergEquilibrium (std::vector< unsigned int > genotypeCounts)

  Calculate whether passed genotype counts are in HWE.
- double SNPHWE (int obs_hets, int obs_hom1, int obs_hom2)

  This code implements an exact SNP test of Hardy-Weinberg Equilibrium.
- double GetClassProbability (ClassLevel thisClass)

  Get the probability of a class value in the data set.
- double GetProbabilityValueGivenClass (unsigned int attributeIndex, AttributeLevel A, ClassLevel classValue)

  Get the probability of an attribute value at an attribute index.
- void AttributeInteractionInformation ()

Calculate and display interaction information for all attribute combinations.
- void CalculateInteractionInformation (std::map< std::pair< int, int >, std::map< std::string, double > > &results)

  Calculate all the information needed to construct the interaction diagram.
- bool CalculateGainMatrix (double **gainMatrix)

  Calculate the GAIN matrix to run snprank on this data set.

Protected Member Functions

- virtual bool LoadSnps (std::string filename)

  Load SNPs from file using the data set filename.
- void UpdateAllLevelCounts ()

  Update level counts for all instances by calling UpdateLevelCounts(inst)
- void UpdateLevelCounts (DatasetInstance *dsi)

  Update all attribute level counts from one data set instance.
- bool LoadNumerics (std::string filename)

  Load numerics (continuous attributes) from a file set in the constructor.
- bool GetNumericValues (unsigned int numericIndex, std::vector< NumericLevel > &numericValues)

  Loads the referenced vector with an numeric's values (column).
- bool LoadAlternatePhenotypes (std::string filename)

  Load alternate phenotype/class values from a plink covariate .cov file.
- bool IsLoadableInstanceID (std::string ID)

  Is the passed instance ID loadable (not filtered).

Protected Attributes

- std::string snpsFilename

  file from which the discrete attributes (SNPSs) were read
- bool hasGenotypes

  does the data set contain any genotypes?
- std::vector< std::string > attributeNames

  discrete attribute names read from file
- std::vector< std::map < AttributeLevel, unsigned int > > levelCounts

  attribute values/levels counts
- std::vector< std::map < std::pair< AttributeLevel, ClassLevel >, unsigned int > > levelCountsByClass

  attribute values/levels counts by discrete class
- std::vector< std::set < std::string > > attributeLevelsSeen

  unique attribute values/levels read from file
- std::vector< std::pair< char, char > > attributeAlleles

  allele1, allele2
- std::vector< std::map< char, unsigned int > > attributeAlleleCounts

allele->count
- std::vector< std::pair< char, double > > attributeMinorAllele

  minor allele, minor allele frequency
- std::vector< std::map < std::string, unsigned int > > genotypeCounts

  genotype->count
- std::vector < AttributeMutationType > attributeMutationTypes

  Keep mutation type for all attributes.
- std::map< std::pair< char, char >, AttributeMutationType > attributeMutation-Map

  Lookup table for mutation type.
- std::string numericsFilename

  file from which the continuous attributes were read
- bool hasNumerics

  does the data set contain any continuous attributes?
- std::vector< std::string > numericsIds

  IDs associated with the numerics read from file.
- std::vector< std::pair < NumericLevel, NumericLevel > > numericsMinMax

  the minimum and maximum value for each continuous attribute
- std::vector< std::string > numericsNames

  continuous attribute names read from file
- std::string alternatePhenotypesFilename

  file from which the alternate phenotypes (class labels) were read
- bool hasAlternatePhenotypes

  does the data set contain alternate phenotypes?
- std::vector< std::string > phenotypesIds

  IDs associated with the phenotypes/classes read from file.
- bool hasContinuousPhenotypes

  does the data set contain continuous phenotypes?
- std::pair< NumericLevel, NumericLevel > continuousPhenotypeMinMax

  the minimum and maximum value for each continuous phenotype
- std::vector< DatasetInstance ∗ > instances

  vector of pointers to all instances in the data set
- std::vector< std::string > instanceIds

  IDs associated with the instances read from file.
- std::vector< std::string > instanceIdsToLoad

  IDs of instances to load from numeric and/or phenotype files.
- std::map< std::string, std::vector< unsigned int > > missingValues

  missing discrete values and their instance indices
- std::map< std::string, std::vector< unsigned int > > missingNumericValues

  missing continuous values and their instance indices
- unsigned int classColumn

  class column from the original data set
- std::map< ClassLevel, std::vector< unsigned int > > classIndexes

class values mapped to instance indices

- std::map< std::string, unsigned int > attributesMask
- std::map< std::string, unsigned int > numericsMask
- std::map< std::string, unsigned int > instancesMask
- std::map< std::string, unsigned int > attributesMaskPushed

  masks can be temporarily pushed and popped
- std::map< std::string, unsigned int > numericsMaskPushed
- std::map< std::string, unsigned int > instancesMaskPushed
- bool maskIsPushed
- GSLRandomFlat ∗ rng

  random number generator classes use GNU Scienitifc Library (GSL)

6.3.1 Detailed Description

Base class for collections of instances containing attributea and class.

Added interaction infomation week of 4/18-26/06 Totally redone for McKinney Lab. - February 2011.

Author

Bill White

Version

1.0

Contact: `bill.c.white@gmail.com` Created on: 6/14/05

Definition at line 115 of file Dataset.h.

6.3.2 Constructor & Destructor Documentation

6.3.2.1 Dataset::Dataset ( )

Construct a default data set.

Set private data to defaults.

Load attribute mutation map for transitions/transversions.

Definition at line 46 of file Dataset.cpp.

6.3.2.2 Dataset::∼Dataset ( ) `[virtual]`

Destruct all dynamically allocated memory.

Definition at line 78 of file Dataset.cpp.

6.3.3 Member Function Documentation

6.3.3.1 void **Dataset::AttributeInteractionInformation** ( )

Calculate and display interaction information for all attribute combinations.

get the column sum

display results detail; I(A;B|C) column as percentage

Definition at line 1581 of file Dataset.cpp.

6.3.3.2 bool **Dataset::CalculateGainMatrix** (double ** *gainMatrix*)

Calculate the GAIN matrix to run snprank on this data set.

Uses OpenMP to calculate matrix entries in parallel threads.

Parameters

`out`	*gainMatrix*	pointer to an allocated n x n matrix, n = number of attributes

Returns

success

Calculate the interaction information from entropies

Populate the GAIN matrix

Definition at line 1753 of file Dataset.cpp.

6.3.3.3 void **Dataset::CalculateInteractionInformation** (std::map< std::pair< int, int >, std::map< std::string, double > > & *results*)

Calculate all the information needed to construct the interaction diagram.

Parameters

`out`	*results*	map of attribute combinations to results

Insure only discrete values

Get the class values once

for all possible (unique) interactions, ie nCk

load attribute values (columns) into vectors for Statistics routines

construct a new attribute with a and b

compute all information theoretic quantities and save the results

Definition at line 1635 of file Dataset.cpp.

6.3.3.4 bool Dataset::CheckHardyWeinbergEquilibrium (std::vector< unsigned int > *genotypeCounts*)

Calculate whether passed genotype counts are in HWE.

Parameters

genotype-Counts	vector of genotype counts: AA, Aa, aa

Returns

counts are in HWE?

observered counts

HWE probabilities

expected values

perform Pearson's chi-squared test

one degree of freedom (# genotypes - # alleles), 5% significance level

Definition at line 1405 of file Dataset.cpp.

6.3.3.5 bool Dataset::ExtractAttributes (std::string *scoresFilename,* unsigned int *topN,* std::string *newDatasetFilename*)

Extracts top N attributes based on a file of attribute scores and writes a new dataset.

Revised 10/3/11 for numerics and continuous class/phenotypes.

Parameters

`in`	*scores-Filename*	filename of attribute scores and names
`in`	*topN*	top N attributes
`in`	*newDataset-Filename*	filename of new dataset

Returns

success

Definition at line 413 of file Dataset.cpp.

6.3.3.6 string Dataset::GetAlternatePhenotypesFilename ( )

Get the alternate phenotype filename.

Definition at line 791 of file Dataset.cpp.

6.3.3.7 AttributeLevel Dataset::GetAttribute (unsigned *instanceIndex,* std::string *name*)

Get attribute value for attribute name at instance index.

Parameters

`in`	*instance-Index*	instance index
`in`	*name*	attribute name

Returns

attributevalue

Definition at line 621 of file Dataset.cpp.

6.3.3.8 unsigned int Dataset::GetAttributeIndexFromName (std::string *attributeName*)

Looks up original attribute index from attribute name.

Parameters

`in`	*attribute-Name*	attribute name

Returns

attribute index or INVALID_INDEX

Definition at line 676 of file Dataset.cpp.

6.3.3.9 pair$<$ char, double $>$ Dataset::GetAttributeMAF (unsigned int *attributeIndex*) `[virtual]`

Get attribute minor allele and frequency.

Parameters

`in`	*attribute*	index

Returns

pair (minor allele, minor allele frequency)

An Intriduction to Genetic Analysis by Griffiths, Miller, Suzuki, Lewontin and Gelbart, 2000, page 715.

Reimplemented in PlinkDataset, and PlinkBinaryDataset.

Definition at line 637 of file Dataset.cpp.

6.3.3.10 AttributeMutationType Dataset::GetAttributeMutationType (unsigned int *attributeIndex*) [virtual]

Get attribute mutation type.

Parameters

in	*attribute*	index

Returns

mutation type (transition, transversion, unknown)

Reimplemented in PlinkDataset, and PlinkBinaryDataset.

Definition at line 661 of file Dataset.cpp.

6.3.3.11 vector< string > Dataset::GetAttributeNames ( )

Return the discrete (SNP) attribute names.

Returns

vector of attribute names

Definition at line 566 of file Dataset.cpp.

6.3.3.12 bool Dataset::GetAttributeRowCol (unsigned int *row*, unsigned int *col*, AttributeLevel & *attrVal*)

Get the attribute value at row, column.

Same as instance index, attribute index.

Parameters

in	*row*	instance row
in	*col*	attribute column
out	*attrVal*	attribute value

Returns

success

Definition at line 224 of file Dataset.cpp.

6.3.3.13 bool **Dataset::GetAttributeValues** (unsigned int *attributeIndex,* std::vector< **AttributeLevel** > & *attributeValues*)

Loads the referenced vector with an attribute's values (column).

from the dataset

Parameters

`in`	*attribute-Index*	attribute index
`out`	*attribute-Values*	reference to a a vector allocated by the caller

Returns

success

6.3.3.14 bool **Dataset::GetAttributeValues** (std::string *attributeName,* std::vector< **AttributeLevel** > & *attributeValues*)

Loads the referenced vector with an attribute's values (column) from the dataset.

Parameters

`in`	*attribute-Name*	attribute name
`out`	*attribute-Values*	reference to a a vector allocated by the caller

Returns

success

6.3.3.15 unsigned int **Dataset::GetClassColumn** ( )

Get the class column as read from the file.

Definition at line 765 of file Dataset.cpp.

6.3.3.16 const std::map< **ClassLevel**, std::vector< unsigned int > > & **Dataset::GetClassIndexes** ( )

Get a map from class levels to a vector of instance indices.

Returns

map of class => instance indices

Definition at line 779 of file Dataset.cpp.

6.3.3.17 double **Dataset::GetClassProbability** (**ClassLevel** *thisClass*)

Get the probability of a class value in the data set.

Parameters

thisClass	class value

Returns

probability

Definition at line 1559 of file Dataset.cpp.

6.3.3.18 bool **Dataset::GetClassValues** (std::vector< **ClassLevel** > & *classValues*)

Loads the referenced vector with the dataset's class labels.

Parameters

`out`	*classValues*	reference to a a vector allocated by the caller

Returns

success

Definition at line 769 of file Dataset.cpp.

6.3.3.19 **DatasetInstance** * **Dataset::GetInstance** (unsigned int *index*)

Returns a pointer to a dataset instance selected by index.

Parameters

`in`	*index*	index of instance

Returns

pointer to an instance

Definition at line 528 of file Dataset.cpp.

6.3.3.20 vector< string > Dataset::GetInstanceIds ( )

Get all instance IDs.

Returns

vector of instance IDs

Definition at line 541 of file Dataset.cpp.

6.3.3.21 bool Dataset::GetInstanceIndexForID (std::string *ID*, unsigned int & *instanceIndex*)

Get the instance index from the instance ID.

Parameters

in	*ID*	string ID
out	*instance-Index*	instance index

Returns

success

Definition at line 550 of file Dataset.cpp.

6.3.3.22 bool Dataset::GetIntForGenotype (std::string *genotype*, AttributeLevel & *newAttr*)

Get integer value for string genotype.

Parameters

in	*genotype*	genotype string
out	*newAttr*	new attribute value

Returns

success

6.3.3.23 double Dataset::GetMeanForNumeric (unsigned int *numericIdx*)

Get the mean/average of numeric at index.

Parameters

`in`	*numericIdx*	numeric index

Returns

average value of numeric attribute at index

Definition at line 703 of file Dataset.cpp.

6.3.3.24 pair< double, double > **Dataset::GetMinMaxForContinuousPhenotype ( )**

Get the minumum and maximum values for the continuous phenotype.

Returns

minimum/maximum pair

Definition at line 799 of file Dataset.cpp.

6.3.3.25 pair< **NumericLevel**, **NumericLevel** > **Dataset::GetMinMaxForNumeric (** unsigned int *numericIdx*)

Get the minimum and maximum values for a numeric at index.

Parameters

`in`	*numericIdx*	numeric index

Returns

minimum/maximum pair

Definition at line 698 of file Dataset.cpp.

6.3.3.26 **NumericLevel Dataset::GetNumeric (** unsigned int *instanceIndex,* std::string *name*)

Get numeric value for numeric name at instance index.

Parameters

`in`	*instance-Index*	instance index
`in`	*name*	numeric name

Returns

numeric value at index

Definition at line 721 of file Dataset.cpp.

6.3.3.27 unsigned int **Dataset::GetNumericIndexFromName** (std::string *numericName*)

Looks up original numeric index from numeric name.

Parameters

`in`	*numeric-Name*	numeric name

Returns

attribute index or INVALID_INDEX

Definition at line 752 of file Dataset.cpp.

6.3.3.28 bool **Dataset::GetNumericRowCol** (unsigned int *row*, unsigned int *col*, **NumericLevel** & *numVal*)

Get the numeric value at row, column.

Same as instance index, numeric index.

Parameters

`in`	*row*	instance row
`in`	*col*	numeric column
`out`	*numVal*	numeric value

Returns

success

Definition at line 237 of file Dataset.cpp.

6.3.3.29 std::string **Dataset::GetNumericsFilename** ( )

Get the filename numerics were read from.

Definition at line 748 of file Dataset.cpp.

6.3.3.30 vector< string > **Dataset::GetNumericsNames** ( )

Return the numeric attribute names.

Returns

vector of attribute names

Definition at line 689 of file Dataset.cpp.

6.3.3.31 bool **Dataset::GetNumericValues** (std::string *numericName,* std::vector< **NumericLevel** > & *numericValues*)

Loads the referenced vector with a numeric's values (column) from the dataset.

Parameters

`in`	*numeric-Name*	numeric name
`out`	*numeric-Values*	reference to a a vector allocated by the caller

Returns

success

6.3.3.32 bool **Dataset::GetNumericValues** (unsigned int *numericIndex,* std::vector< **NumericLevel** > & *numericValues*) `[protected]`

Loads the referenced vector with an numeric's values (column).

from the dataset

Parameters

`in`	*numeric-Index*	numeric index
`out`	*numeric-Values*	reference to a a vector allocated by the caller

Returns

success

6.3.3.33 double **Dataset::GetProbabilityValueGivenClass** (unsigned int *attributeIndex,* **AttributeLevel** *A,* **ClassLevel** *classValue*)

Get the probability of an attribute value at an attribute index.

Parameters

in	*attribute-Index*	attribute index
in	*A*	attribute value
in	*classValue*	class value

Returns

probability of the value in attribute given class

Definition at line 1566 of file Dataset.cpp.

6.3.3.34 DatasetInstance $*$ Dataset::GetRandomInstance ( )

Returns a pointer to a randomly chosen data set instance.

The random number generator is set to give values in range of instance indexes.

Returns

pointer to a data set instance

Definition at line 536 of file Dataset.cpp.

6.3.3.35 std::string Dataset::GetSnpsFilename ( )

Get the filename SNPs were read from.

Definition at line 613 of file Dataset.cpp.

6.3.3.36 vector$<$ string $>$ Dataset::GetVariableNames ( )

Returns the names of discrete and continuous variables in the data set.

Returns

vector of names as strings

Definition at line 512 of file Dataset.cpp.

6.3.3.37 bool Dataset::HasAlternatePhenotypes ( )

Does the data set have alternate phenotypes loaded?

Definition at line 783 of file Dataset.cpp.

6.3.3.38 void Dataset::HasAlternatePhenotypes (bool *setHasAlternatePhenotypes*)

Definition at line 787 of file Dataset.cpp.

6.3.3.39 bool Dataset::HasContinuousPhenotypes ( )

Definition at line 795 of file Dataset.cpp.

6.3.3.40 bool Dataset::HasGenotypes ( )

Does the data set have genotype variables?

Definition at line 617 of file Dataset.cpp.

6.3.3.41 bool Dataset::HasNumerics ( )

Does the data set have numeric variables? setter/getter.

Definition at line 713 of file Dataset.cpp.

6.3.3.42 void Dataset::HasNumerics (bool *setHasNumerics*)

Definition at line 717 of file Dataset.cpp.

6.3.3.43 bool Dataset::IsLoadableInstanceID (std::string *ID*) `[protected]`

Is the passed instance ID loadable (not filtered).

Parameters

`in`	*ID*	instance ID

Returns

[out] success

Definition at line 2389 of file Dataset.cpp.

6.3.3.44 bool Dataset::LoadAlternatePhenotypes (std::string *filename*) `[protected]`

Load alternate phenotype/class values from a plink covariate .cov file.

Format described here: `http://pngu.mgh.harvard.edu/~purcell/plink/data.-shtml#covar` MAJOR CHANGES: for continuous phenotypes/class - 9/29/11

Parameters

`in`	*filename*	alternate phenotype data filename in PLINK covar format

Returns

success

Detect the class type

Definition at line 2205 of file Dataset.cpp.

6.3.3.45 bool Dataset::LoadDataset (std::string *snpsFilename,* std::string *numericsFilename,* std::string *altPhenoFilename,* std::vector< std::string > *ids*)

Load the dataset from files passed as parameters.

Parameters

in	*snpFilename*	discrete values (SNPs) filename
in	*doRecodeA*	perform recodeA encoding after reading
in	*numerics-Filename*	continuous values (numerics) filename or empty string
in	*altPheno-Filename*	alternate class (phenotype) filename or empty string
in	*ids*	vector of possibly empty IDs to match in auxiliary files

Returns

success

6.3.3.46 bool Dataset::LoadDataset (DgeData ∗ *dgeData*)

Load the dataset from DGE data.

Parameters

in	*dgeData*	pointer to a digital gene expression (DGE) data object

Returns

success

Definition at line 177 of file Dataset.cpp.

6.3.3.47 bool Dataset::LoadNumerics (std::string *filename*) `[protected]`

Load numerics (continuous attributes) from a file set in the constructor.

Parameters

in	*filename*	numerics data filename in PLINK covar format

Returns

success

Definition at line 2040 of file Dataset.cpp.

6.3.3.48 bool **Dataset::LoadSnps** (std::string *filename*) `[protected, virtual]`

Load SNPs from file using the data set filename.

Parameters

`in`	*filename*	SNPs filename
`in`	*deRecodeA*	perform a recodeA operation after reading raw data?

Returns

success

------------ Beginning of private methods ------------------ Open the data file and read line-by-line

Detect the class type

Reimplemented in ArffDataset, PlinkDataset, PlinkBinaryDataset, and PlinkRaw-Dataset.

Definition at line 1782 of file Dataset.cpp.

6.3.3.49 vector< string > **Dataset::MaskGetAllVariableNames** ( )

Return a vector of all the variable names under consideration.

Returns

vector of discrete and numeric variable

Definition at line 1128 of file Dataset.cpp.

6.3.3.50 vector< unsigned int > **Dataset::MaskGetAttributeIndices** (**AttributeType** *attrType*)

Return a vector of all the attribute indices under consideration.

Parameters

attrType	attribute type

Returns

vector of indices into currently considered discrete attributes

Definition at line 1103 of file Dataset.cpp.

6.3.3.51 const map< string, unsigned int > & Dataset::MaskGetAttributeMask (AttributeType *attrType*)

Return a map of attribute name to attribute index of attributes to include.

Parameters

in	*attrType*	attribute type

Returns

attributes mask: name->index

Definition at line 1120 of file Dataset.cpp.

6.3.3.52 vector< string > Dataset::MaskGetInstanceIds ( )

Return a vector of all the instance ids under consideration.

Returns

vector of ids of currently included instances

Definition at line 1183 of file Dataset.cpp.

6.3.3.53 vector< unsigned int > Dataset::MaskGetInstanceIndices ( )

Return a vector of all the instance indices under consideration.

vector of indices into current instances

Definition at line 1174 of file Dataset.cpp.

6.3.3.54 const map< string, unsigned int > & Dataset::MaskGetInstanceMask ( )

Return a map of instance name to instance index of instances to include.

Returns

instances mask: instance ID=>vector of instance indices

Definition at line 1192 of file Dataset.cpp.

6.3.3.55 bool Dataset::MaskIncludeAllAttributes (AttributeType *attrType*)

Mark all attributes for inclusion in data set operations.

Parameters

`in`	*attrType*	attribute type

Returns

success

Definition at line 1081 of file Dataset.cpp.

6.3.3.56 bool Dataset::MaskIncludeAllInstances ( )

Mark all instances for inclusion in algorithms.

Returns

success

Definition at line 1162 of file Dataset.cpp.

6.3.3.57 bool Dataset::MaskPopAll ( )

Restore the masks previously pushed.

Returns

success

Definition at line 1210 of file Dataset.cpp.

6.3.3.58 bool Dataset::MaskPushAll ( )

Save the current masks for later restore.

Returns

success

Definition at line 1196 of file Dataset.cpp.

6.3.3.59 bool Dataset::MaskRemoveInstance (std::string *instanceId*)

Removes the instance from consideration in any data set operations.

Parameters

`in`	*instanceId*	instance ID

Returns

success

Definition at line 1141 of file Dataset.cpp.

6.3.3.60 bool **Dataset::MaskRemoveVariable** (std::string *variableName*)

Removes the variable name from consideration in any data set operations.

Parameters

`in`	*variable-Name*	variable name

Returns

success

Definition at line 1023 of file Dataset.cpp.

6.3.3.61 bool **Dataset::MaskRemoveVariableType** (std::string *variableName,* **AttributeType** *varType*)

Removes the attribute name from consideration in any data set operations.

Parameters

`in`	*attribute-Name*	attribute name
`in`	*attrType*	attribute type

Returns

success

Definition at line 1035 of file Dataset.cpp.

6.3.3.62 bool **Dataset::MaskSearchInstance** (std::string *instanceId*)

Determines if the names Instance is in the current masked dataaset.

Parameters

in	*instanceID*	instance ID

Returns

true if instance ID is in the dataset, considering instance mask

Definition at line 1153 of file Dataset.cpp.

6.3.3.63 bool Dataset::MaskSearchVariableType (std::string *variableName*, AttributeType *attrType*)

Determines if the named variable is in the current masked data set.

Parameters

in	*attribute-Name*	attribute name
in	*attributeType*	attribute type

Returns

true if discrete attribute name is being considered in operations.

Definition at line 1061 of file Dataset.cpp.

6.3.3.64 bool Dataset::MaskWriteNewDataset (std::string *newDatasetFilename*)

Saved the unmasked attributes as a tab-delimited text file.

Parameters

in	*newDataset-Filename*	new data set filename

Returns

success

Definition at line 1223 of file Dataset.cpp.

6.3.3.65 unsigned int Dataset::NumAttributes ( ) [virtual]

Return the number of unmasked discrete attributes in the data set.

Definition at line 562 of file Dataset.cpp.

6.3.3.66 unsigned int **Dataset::NumClasses** ( )

Get the number of classes in the data set.

Definition at line 761 of file Dataset.cpp.

6.3.3.67 unsigned int **Dataset::NumInstances** ( ) `[virtual]`

Returns the number of instances in the data set.

Definition at line 524 of file Dataset.cpp.

6.3.3.68 unsigned int **Dataset::NumLevels** (unsigned int *index*)

Returns the number of levels in a given attribute index.

Parameters

`in`	*index*	attribute index

Returns

number of levels

Definition at line 666 of file Dataset.cpp.

6.3.3.69 unsigned int **Dataset::NumNumerics** ( )

Return the number of unmasked discrete attributes in the data set.

Definition at line 685 of file Dataset.cpp.

6.3.3.70 unsigned int **Dataset::NumVariables** ( )

Return the number of discrete plus continuous variables in the data set.

The number does not include masked variables removed.

Returns

number of discrete plus continuous variables

Definition at line 508 of file Dataset.cpp.

6.3.3.71 void **Dataset::Print** ( )

Print the entire data set in compact format.

Definition at line 803 of file Dataset.cpp.

6.3.3.72 void **Dataset::PrintAttributeLevelsSeen** ( )

Print unique attribute levels seen.

Definition at line 1008 of file Dataset.cpp.

6.3.3.73 void **Dataset::PrintClassIndexInfo** ( )

Print class index information.

Definition at line 912 of file Dataset.cpp.

6.3.3.74 void **Dataset::PrintLevelCounts** ( )

Prit attribute level counts.

Definition at line 949 of file Dataset.cpp.

6.3.3.75 void **Dataset::PrintMaskStats** ( )

Print mask statistics.

Definition at line 1271 of file Dataset.cpp.

6.3.3.76 void **Dataset::PrintMissingValuesStats** ( )

Print missing value statistics.

Definition at line 923 of file Dataset.cpp.

6.3.3.77 void **Dataset::PrintNumericsStats** ( )

Print statistics about the data set including numerics.

Definition at line 845 of file Dataset.cpp.

6.3.3.78 void **Dataset::PrintRecodeMap** (std::vector< std::map< unsigned int, unsigned int > > *recodeMap*)

Print the passed recode map to stdout.

See also

DoRecodeA()

Parameters

`in`	*recodeMap*	recoding map

Definition at line 990 of file Dataset.cpp.

6.3.3.79 void **Dataset::PrintStats** ( )

Print basic statstics abou the data set - discrete/SNPs only.

Definition at line 811 of file Dataset.cpp.

6.3.3.80 void **Dataset::PrintStatsSimple** ( )

Print very simple statistics abou the data set with no formatting.

Definition at line 882 of file Dataset.cpp.

6.3.3.81 void **Dataset::RunSnpDiagnosticTests** (std::string *logFilename*, double *globalGenotypeThreshold* = `0.01`, unsigned int *cellThreshold* = `5`)

Perform and report SNP diagnostic test information.

Parameters

`in`	*logFilename*	log filename
`in`	*global-Genotype-Threshold*	genotype count threshold
`in`	*cell-Threshold*	x^2 cell count threshold

Definition at line 1280 of file Dataset.cpp.

6.3.3.82 double **Dataset::SNPHWE** (int *obs_hets*, int *obs_hom1*, int *obs_hom2*)

This code implements an exact SNP test of Hardy-Weinberg Equilibrium.

As described in Wigginton, JE, Cutler, DJ, and Abecasis, GR (2005) A Note on Exact Tests of Hardy-Weinberg Equilibrium. American Journal of Human Genetics: 76. - Written by Jan Wigginton.

Parameters

`in`	*obs_hets*	observed heterozygotes
`in`	*obs_hom1*	observed homozygotes type 1
`in`	*obs_hom2*	homozygotes type 2

Returns

HWE value

Definition at line 1466 of file Dataset.cpp.

6.3.3.83 bool Dataset::SwapAttributes (unsigned int *a1*, unsigned int *a2*)

Swap two attributes/columns in the dataset.

Parameters

in	*a1*	attribue index 1
in	*a2*	attribue index 2

Returns

success

Definition at line 499 of file Dataset.cpp.

6.3.3.84 void Dataset::UpdateAllLevelCounts ( ) [protected]

Update level counts for all instances by calling UpdateLevelCounts(inst)

Definition at line 2001 of file Dataset.cpp.

6.3.3.85 void Dataset::UpdateLevelCounts (DatasetInstance ∗ *dsi*) [protected]

Update all attribute level counts from one data set instance.

Updates levelCountsByClass.

Parameters

in	*dsi*	pointer to a data set instance

Definition at line 2025 of file Dataset.cpp.

6.3.3.86 void Dataset::WriteLevelCounts (std::string *levelsFilename*)

Write attribute level counts to a text file.

Parameters

in	*levels-Filename*	filename to write levels to

Definition at line 966 of file Dataset.cpp.

6.3.3.87 bool **Dataset::WriteNewDataset** (std::string *newDatasetFilename,* **OutputDatasetType** *outputDatasetType*)

Write the dataset to a new filename, respecting masked attributes and numerics and class/phenotype data type.

Parameters

`in`	*newDataset-Filename*	new dataset filename

Returns

success

write the attribute names header

write the data, respecting the masked attributes, numerics and masked instances - 10/28/11 write the attribute names header

write continuous attribute values

Definition at line 250 of file Dataset.cpp.

6.3.4 Member Data Documentation

6.3.4.1 std::string **Dataset::alternatePhenotypesFilename** `[protected]`

file from which the alternate phenotypes (class labels) were read

Definition at line 633 of file Dataset.h.

6.3.4.2 std::vector<std::map<char, unsigned int> > **Dataset::attributeAlleleCounts** `[protected]`

allele->count

Definition at line 611 of file Dataset.h.

6.3.4.3 std::vector<std::pair<char, char> > **Dataset::attributeAlleles** `[protected]`

allele1, allele2

Definition at line 609 of file Dataset.h.

6.3.4.4 std::vector<std::set<std::string> > **Dataset::attributeLevelsSeen** `[protected]`

unique attribute values/levels read from file

Definition at line 607 of file Dataset.h.

6.3.4.5 std::vector<std::pair<char, double> > Dataset::attributeMinorAllele [protected]

minor allele, minor allele frequency

Definition at line 613 of file Dataset.h.

6.3.4.6 std::map<std::pair<char, char>, AttributeMutationType> Dataset::attributeMutationMap [protected]

Lookup table for mutation type.

Definition at line 619 of file Dataset.h.

6.3.4.7 std::vector<AttributeMutationType> Dataset::attributeMutationTypes [protected]

Keep mutation type for all attributes.

Definition at line 617 of file Dataset.h.

6.3.4.8 std::vector<std::string> Dataset::attributeNames [protected]

discrete attribute names read from file

Definition at line 601 of file Dataset.h.

6.3.4.9 std::map<std::string, unsigned int> Dataset::attributesMask [protected]

Definition at line 664 of file Dataset.h.

6.3.4.10 std::map<std::string, unsigned int> Dataset::attributesMaskPushed [protected]

masks can be temporarily pushed and popped

Definition at line 668 of file Dataset.h.

6.3.4.11 unsigned int Dataset::classColumn [protected]

class column from the original data set

Definition at line 655 of file Dataset.h.

6.3.4.12 std::map<ClassLevel, std::vector<unsigned int> > Dataset::classIndexes [protected]

class values mapped to instance indices

Definition at line 657 of file Dataset.h.

6.3.4.13 std::pair<NumericLevel, NumericLevel> Dataset::continuousPhenotypeMinMax [protected]

the minimum and maximum value for each continuous phenotype

Definition at line 641 of file Dataset.h.

6.3.4.14 std::vector<std::map<std::string, unsigned int> > Dataset::genotypeCounts [protected]

genotype->count

Definition at line 615 of file Dataset.h.

6.3.4.15 bool Dataset::hasAlternatePhenotypes [protected]

does the data set contain alternate phenotypes?

Definition at line 635 of file Dataset.h.

6.3.4.16 bool Dataset::hasContinuousPhenotypes [protected]

does the data set contain continuous phenotypes?

Definition at line 639 of file Dataset.h.

6.3.4.17 bool Dataset::hasGenotypes [protected]

does the data set contain any genotypes?

Definition at line 599 of file Dataset.h.

6.3.4.18 bool Dataset::hasNumerics [protected]

does the data set contain any continuous attributes?

Definition at line 624 of file Dataset.h.

6.3.4.19 std::vector<std::string> Dataset::instanceIds [protected]

IDs associated with the instances read from file.

Definition at line 646 of file Dataset.h.

6.3.4.20 std::vector<std::string> **Dataset::instanceIdsToLoad** `[protected]`

IDs of instances to load from numeric and/or phenotype files.

Definition at line 648 of file Dataset.h.

6.3.4.21 std::vector<**DatasetInstance**∗> **Dataset::instances** `[protected]`

vector of pointers to all instances in the data set

Definition at line 644 of file Dataset.h.

6.3.4.22 std::map<std::string, unsigned int> **Dataset::instancesMask** `[protected]`

Definition at line 666 of file Dataset.h.

6.3.4.23 std::map<std::string, unsigned int> **Dataset::instancesMaskPushed** `[protected]`

Definition at line 670 of file Dataset.h.

6.3.4.24 std::vector<std::map<**AttributeLevel**, unsigned int> > **Dataset::levelCounts** `[protected]`

attribute values/levels counts

Definition at line 603 of file Dataset.h.

6.3.4.25 std::vector<std::map<std::pair<**AttributeLevel**, **ClassLevel**>, unsigned int> > **Dataset::levelCountsByClass** `[protected]`

attribute values/levels counts by discrete class

Definition at line 605 of file Dataset.h.

6.3.4.26 bool **Dataset::maskIsPushed** `[protected]`

Definition at line 671 of file Dataset.h.

6.3.4.27 std::map<std::string, std::vector<unsigned int> > **Dataset::missingNumericValues** `[protected]`

missing continuous values and their instance indices

Definition at line 652 of file Dataset.h.

6.3.4.28 std::map<std::string, std::vector<unsigned int> > **Dataset::missingValues** `[protected]`

missing discrete values and their instance indices

Definition at line 650 of file Dataset.h.

6.3.4.29 std::string **Dataset::numericsFilename** `[protected]`

file from which the continuous attributes were read

Definition at line 622 of file Dataset.h.

6.3.4.30 std::vector<std::string> **Dataset::numericsIds** `[protected]`

IDs associated with the numerics read from file.

Definition at line 626 of file Dataset.h.

6.3.4.31 std::map<std::string, unsigned int> **Dataset::numericsMask** `[protected]`

Definition at line 665 of file Dataset.h.

6.3.4.32 std::map<std::string, unsigned int> **Dataset::numericsMaskPushed** `[protected]`

Definition at line 669 of file Dataset.h.

6.3.4.33 std::vector< std::pair<**NumericLevel**, **NumericLevel**> > **Dataset::numericsMinMax** `[protected]`

the minimum and maximum value for each continuous attribute

Definition at line 628 of file Dataset.h.

6.3.4.34 std::vector<std::string> **Dataset::numericsNames** `[protected]`

continuous attribute names read from file

Definition at line 630 of file Dataset.h.

6.3.4.35 std::vector<std::string> **Dataset::phenotypesIds** `[protected]`

IDs associated with the phenotypes/classes read from file.

Definition at line 637 of file Dataset.h.

6.3.4.36 GSLRandomFlat∗ Dataset::rng `[protected]`

random number generator classes use GNU Scienitifc Library (GSL)

Definition at line 674 of file Dataset.h.

6.3.4.37 std::string Dataset::snpsFilename `[protected]`

file from which the discrete attributes (SNPSs) were read

Definition at line 597 of file Dataset.h.

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

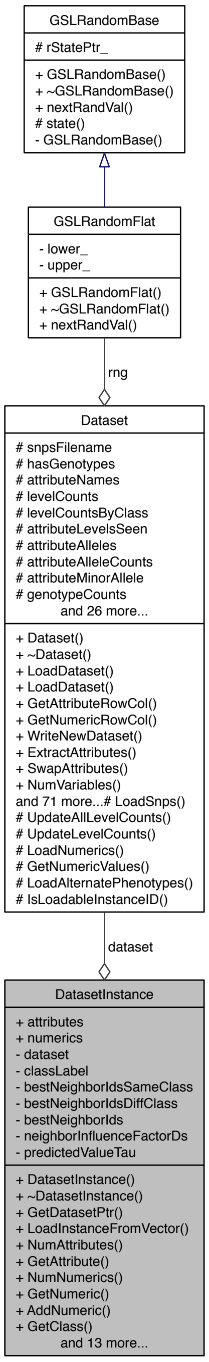
- src/library/Dataset.h
- src/library/Dataset.cpp

6.4 DatasetInstance Class Reference

Class to hold dataset instances (rows of attributes).

`#include <DatasetInstance.h>`

Collaboration diagram for DatasetInstance:



Public Member Functions

- DatasetInstance (Dataset ∗ds)

    Construct an data set instance object.
- ∼DatasetInstance ()
- Dataset ∗ GetDatasetPtr ()

    return the Dataset pointer associated with this instance
- bool LoadInstanceFromVector (std::vector< AttributeLevel > newAttributes)

    Load this instance with the attributes and class value from the newAttributes vector.
- unsigned int NumAttributes ()

    return the number of discrete attributes
- AttributeLevel GetAttribute (unsigned int index)

    Get and return an attribute value at index.
- unsigned int NumNumerics ()

    return the number of continuous attributes
- NumericLevel GetNumeric (unsigned int index)

    Get and return numeric value at index.
- bool AddNumeric (NumericLevel newNum)

    Add a numeric value to the instance's numerics vector.
- ClassLevel GetClass ()

    Get the discrete class value.
- void SetClass (ClassLevel classValue)

    Set the discrete class value.
- double GetPredictedValueTau ()

    Get the continuous class value.
- void SetPredictedValueTau (double newValue)

    Set the continuous class value.
- double GetInfluenceFactorD (unsigned int neighborIndex)

    Get the nearest neighbor value at neighborIndex.
- void ClearInfluenceFactors ()

    Clear all nearest neighbor values.
- bool AddInfluenceFactorD (double factor)

    Add the next nearest neighbor influence factor.
- void Print ()

    Print the attributes, numerics and class name of this instance to stdout.
- bool SwapAttributes (unsigned int a1, unsigned int a2)

    Swap attribute/column values in this instance.
- void SetDistanceSums (unsigned int kNearestNeighbors, DistancePairs &sameClassSums, std::map< ClassLevel, DistancePairs > &diffClassSums)

    Set the best kNearestNeighbors from the same and different classes SIDE_EFFECT: Sorts and loads class the vairables: sameSums snd diffSums from the neighbors.
- void SetDistanceSums (unsigned int kNearestNeighbors, DistancePairs instancesSums)

    Set the best kNearestNeighbors from all other instances/neighbors.

- void PrintDistancePairs (const DistancePairs &distPairs)

  Prints passed distance pairs.
- bool GetNNearestInstances (unsigned int n, std::vector< unsigned int > &sameClassInstances, std::vector< unsigned int > &diffClassInstances)

  Returns N closest instances using the sameSums and diffSums class variables.
- bool GetNNearestInstances (unsigned int n, std::vector< unsigned int > &sameClassInstances, std::map< ClassLevel, std::vector< unsigned int > > &diffClassInstances)

  Returns N closest instances using the sameSums and diffSums class variables.
- bool GetNNearestInstances (unsigned int n, std::vector< unsigned int > &closestInstances)

  Returns N closest instances to this instance.

Public Attributes

- std::vector< AttributeLevel > attributes

  discrete attributes
- std::vector< NumericLevel > numerics

  continuous attributes

Private Attributes

- Dataset $*$ dataset

  pointer to a Dataset object
- ClassLevel classLabel

  the class value for this instance
- std::vector< std::string > bestNeighborIdsSameClass

  vector of instance IDs for the best neighbors in this instance's class
- std::map< ClassLevel, std::vector< std::string > > bestNeighborIdsDiffClass

  vector of instance IDs for the best neighbors of different class(es)
- std::vector< std::string > bestNeighborIds

  best neighbor IDs for continuous class
- std::vector< double > neighborInfluenceFactorDs

  nearest neighbor weighting factors
- double predictedValueTau

  countinuous value for this class

6.4.1 Detailed Description

Class to hold dataset instances (rows of attributes).

Reworked entirely for McKinney Lab work - 2/28/11

Author

Bill White

Version

1.0

Contact: bill.c.white@gmail.com Created on: 6/14/05

Definition at line 40 of file DatasetInstance.h.

6.4.2 Constructor & Destructor Documentation

6.4.2.1 DatasetInstance::DatasetInstance (Dataset * *ds*)

Construct an data set instance object.

Parameters

in	*ds*	pointer to a Dataset object

Definition at line 34 of file DatasetInstance.cpp.

6.4.2.2 DatasetInstance::∼DatasetInstance ( )

Definition at line 40 of file DatasetInstance.cpp.

6.4.3 Member Function Documentation

6.4.3.1 bool DatasetInstance::AddInfluenceFactorD (double *factor*)

Add the next nearest neighbor influence factor.

Definition at line 129 of file DatasetInstance.cpp.

6.4.3.2 bool DatasetInstance::AddNumeric (NumericLevel *newNum*)

Add a numeric value to the instance's numerics vector.

Parameters

in	*newNum*	new numeric value

Returns

success

Definition at line 99 of file DatasetInstance.cpp.

6.4.3.3 void **DatasetInstance::ClearInfluenceFactors** ( )

Clear all nearest neighbor values.

Definition at line 125 of file DatasetInstance.cpp.

6.4.3.4 **AttributeLevel DatasetInstance::GetAttribute** (unsigned int *index*)

Get and return an attribute value at index.

Parameters

`in`	*index*	attribute index

Returns

attribute value at index

Definition at line 64 of file DatasetInstance.cpp.

6.4.3.5 **ClassLevel DatasetInstance::GetClass** ( )

Get the discrete class value.

Definition at line 105 of file DatasetInstance.cpp.

6.4.3.6 **Dataset** $*$ **DatasetInstance::GetDatasetPtr** ( )

return the Dataset pointer associated with this instance

Definition at line 43 of file DatasetInstance.cpp.

6.4.3.7 double **DatasetInstance::GetInfluenceFactorD** (unsigned int *neighborIndex*)

Get the nearest neighbor value at neighborIndex.

Definition at line 121 of file DatasetInstance.cpp.

6.4.3.8 bool **DatasetInstance::GetNNearestInstances** (unsigned int *n*, std::vector< unsigned int > & *sameClassInstances*, std::vector< unsigned int > & *diffClassInstances*)

Returns N closest instances using the sameSums and diffSums class variables.

Parameters

in	*n*	n nearest nerighbors
in	*sameCLass-Instances*	vector of same class instances indices
in	*diffClass-Instances*	vector of different class instance indices

Returns

success

6.4.3.9 bool **DatasetInstance::GetNNearestInstances** (unsigned int *n,* std::vector< unsigned int > & *sameClassInstances,* std::map< **ClassLevel**, std::vector< unsigned int > > & *diffClassInstances*)

Returns N closest instances using the sameSums and diffSums class variables.

Parameters

in	*n*	n nearest nerighbors
in	*sameCLass-Instances*	vector of same class instances indices
in	*diffClass-Instances*	vector of different classes instance indices

Returns

success

6.4.3.10 bool **DatasetInstance::GetNNearestInstances** (unsigned int *n,* std::vector< unsigned int > & *closestInstances*)

Returns N closest instances to this instance.

Parameters

in	*n*	n nearest neighbors
in	*closest-Instances*	reference to a vector of instance indices

Returns

success

6.4.3.11 double **DatasetInstance::GetNumeric** (unsigned int *index*)

Get and return numeric value at index.

Parameters

in	*index*	numeric index

Returns

numeric value at index

Definition at line 84 of file DatasetInstance.cpp.

6.4.3.12 double **DatasetInstance::GetPredictedValueTau** ( )

Get the continuous class value.

Definition at line 113 of file DatasetInstance.cpp.

6.4.3.13 bool **DatasetInstance::LoadInstanceFromVector** (std::vector< **AttributeLevel** > *newAttributes*)

Load this instance with the attributes and class value from the newAttributes vector.

Parameters

in	*new-Attributes*	vector of new attribute values

Returns

success

Definition at line 48 of file DatasetInstance.cpp.

6.4.3.14 unsigned int **DatasetInstance::NumAttributes** ( )

return the number of discrete attributes

Definition at line 60 of file DatasetInstance.cpp.

6.4.3.15 unsigned int **DatasetInstance::NumNumerics** ( )

return the number of continuous attributes

Definition at line 80 of file DatasetInstance.cpp.

6.4.3.16 void DatasetInstance::Print ( )

Print the attributes, numerics and class name of this instance to stdout.

Definition at line 134 of file DatasetInstance.cpp.

6.4.3.17 void DatasetInstance::PrintDistancePairs (const DistancePairs & *distPairs*)

Prints passed distance pairs.

Parameters

in	*distPairs*	distance pairs

Definition at line 240 of file DatasetInstance.cpp.

6.4.3.18 void DatasetInstance::SetClass (ClassLevel *classValue*)

Set the discrete class value.

Definition at line 109 of file DatasetInstance.cpp.

6.4.3.19 void DatasetInstance::SetDistanceSums (unsigned int *kNearestNeighbors,* DistancePairs & *sameClassSums,* std::map< ClassLevel, DistancePairs > & *diffClassSums*)

Set the best kNearestNeighbors from the same and different classes SIDE_EFFECT: Sorts and loads class the vairables: sameSums snd diffSums from the neighbors.

Parameters

in	*kNearest-Neighbors*	k nearest nerighbors,
in	*sameCLass-Sums*	vectors of pairs <instance, sum> of same class
in	*diffClass-Sums*	vectors of pairs <instance, sum> of other classes

Returns

nothing

6.4.3.20 void DatasetInstance::SetDistanceSums (unsigned int *kNearestNeighbors,* DistancePairs *instancesSums*)

Set the best kNearestNeighbors from all other instances/neighbors.

SIDE_EFFECT: Sorts and loads neighborSums from the instanceSums

Parameters

`in`	*kNearest-Neighbors*	k nearest neighbors
`in`	*instance-Sums*	vectors of k pairs <instance, sum> for neighbors

Returns

nothing

Definition at line 215 of file DatasetInstance.cpp.

6.4.3.21 void **DatasetInstance::SetPredictedValueTau** (double *newValue*)

Set the continuous class value.

Definition at line 117 of file DatasetInstance.cpp.

6.4.3.22 bool **DatasetInstance::SwapAttributes** (unsigned int *a1*, unsigned int *a2*)

Swap attribute/column values in this instance.

Parameters

`in`	*a1*	attribue index 1
`in`	*a2*	attribue index 2

Returns

bool success

Definition at line 154 of file DatasetInstance.cpp.

6.4.4 Member Data Documentation

6.4.4.1 std::vector<AttributeLevel> DatasetInstance::attributes

discrete attributes

Definition at line 158 of file DatasetInstance.h.

6.4.4.2 std::vector<std::string> DatasetInstance::bestNeighborIds `[private]`

best neighbor IDs for continuous class

Definition at line 171 of file DatasetInstance.h.

6.4.4.3 std::map<ClassLevel, std::vector<std::string> > DatasetInstance::bestNeighborIdsDiffClass [private]

vector of instance IDs for the best neighbors of different class(es)

Definition at line 169 of file DatasetInstance.h.

6.4.4.4 std::vector<std::string> DatasetInstance::bestNeighborIdsSameClass [private]

vector of instance IDs for the best neighbors in this instance's class

Definition at line 167 of file DatasetInstance.h.

6.4.4.5 ClassLevel DatasetInstance::classLabel [private]

the class value for this instance

Definition at line 165 of file DatasetInstance.h.

6.4.4.6 Dataset* DatasetInstance::dataset [private]

pointer to a Dataset object

Definition at line 163 of file DatasetInstance.h.

6.4.4.7 std::vector<double> DatasetInstance::neighborInfluenceFactorDs [private]

nearest neighbor weighting factors

Definition at line 173 of file DatasetInstance.h.

6.4.4.8 std::vector<NumericLevel> DatasetInstance::numerics

continuous attributes

Definition at line 160 of file DatasetInstance.h.

6.4.4.9 double DatasetInstance::predictedValueTau [private]

countinuous value for this class

Definition at line 175 of file DatasetInstance.h.

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

- src/library/DatasetInstance.h
- src/library/DatasetInstance.cpp

6.5 deref_less Class Reference

Public Member Functions

- bool operator() (const T a, const T b) const

6.5.1 Detailed Description

Definition at line 60 of file ReliefF.cpp.

6.5.2 Member Function Documentation

6.5.2.1 bool deref_less::operator() (const T *a*, const T *b*) const `[inline]`

Definition at line 64 of file ReliefF.cpp.

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

- src/library/ReliefF.cpp

6.6 deref_less_bcw Class Reference

Public Member Functions

- bool operator() (const T a, const T b) const

6.6.1 Detailed Description

Definition at line 25 of file DatasetInstance.cpp.

6.6.2 Member Function Documentation

6.6.2.1 bool deref_less_bcw::operator() (const T *a*, const T *b*) const `[inline]`

Definition at line 29 of file DatasetInstance.cpp.

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

- src/library/DatasetInstance.cpp

6.7 DgeData Class Reference

Digital gene expression data.

```
#include <DgeData.h>
```

Public Member Functions

- DgeData ()
- virtual ∼DgeData ()
- bool LoadData (std::string countsFile, std::string phenoFile, std::string normsFile="")

  Create a new set of DGE data with a counts file and a phenotype file.
- std::vector< std::string > GetSampleNames ()

  Get the sample names/IDs.
- std::vector< std::string > GetGeneNames ()

  Get the gene names/IDs.
- std::pair< double, double > GetGeneMinMax (int geneIndex)

  Get the min and max values for gene at index.
- int GetNumSamples ()

  Get the number of samples.
- int GetNumGenes ()

  Get the number of genes.
- std::vector< double > GetSampleCounts (int sampleIndex)

  Get sample counts for sample at index.
- int GetSamplePhenotype (int sampleIndex)

  Get the phenotype at sample index.
- std::vector< double > GetNormalizationFactors ()

  Get the normalization factors.
- void PrintSampleStats ()

  Print the Sample statistics to the console.

Private Attributes

- std::string countsFilename

  Filename containing DGE counts.
- std::string phenosFilename

  Filename containing DGE phenotypes.
- std::string normsFilename

  Filename containing DGE normalization factors.
- bool hasNormFactors

  Are we using normalization?
- std::vector< double > normFactors

  Vector of (optional) normalization factors for each sample.
- std::vector< std::string > geneNames

  Gene names.
- std::vector< std::vector < double > > counts

Digital gene expression counts.
- std::vector< std::string > sampleNames

  Sample names.
- std::vector< int > phenotypes

  Sample phenotypes.
- std::vector< std::pair< double, double > > minMaxGeneCounts

  Min and max count for genes.
- std::vector< std::pair< double, double > > minMaxSampleCounts

  Min and max values for samples.
- std::vector< std::vector< int > > sampleZeroes

  Zero count sample indices.

6.7.1 Detailed Description

Digital gene expression data.

Author

Bill White

Version

1.0

Contact: `bill.c.white@gmail.com` Created on: 1/18/12

Definition at line 16 of file DgeData.h.

6.7.2 Constructor & Destructor Documentation

6.7.2.1 DgeData::DgeData ( )

Definition at line 24 of file DgeData.cpp.

6.7.2.2 DgeData::∼DgeData ( ) `[virtual]`

Definition at line 28 of file DgeData.cpp.

6.7.3 Member Function Documentation

6.7.3.1 pair< double, double > DgeData::GetGeneMinMax (int *geneIndex*)

Get the min and max values for gene at index.

Definition at line 240 of file DgeData.cpp.

6.7.3.2 vector< string > DgeData::GetGeneNames ( )

Get the gene names/IDs.

Definition at line 236 of file DgeData.cpp.

6.7.3.3 vector< double > DgeData::GetNormalizationFactors ( )

Get the normalization factors.

Definition at line 284 of file DgeData.cpp.

6.7.3.4 int DgeData::GetNumGenes ( )

Get the number of genes.

Definition at line 255 of file DgeData.cpp.

6.7.3.5 int DgeData::GetNumSamples ( )

Get the number of samples.

Definition at line 251 of file DgeData.cpp.

6.7.3.6 vector< double > DgeData::GetSampleCounts (int *sampleIndex*)

Get sample counts for sample at index.

Definition at line 259 of file DgeData.cpp.

6.7.3.7 vector< string > DgeData::GetSampleNames ( )

Get the sample names/IDs.

Definition at line 232 of file DgeData.cpp.

6.7.3.8 int DgeData::GetSamplePhenotype (int *sampleIndex*)

Get the phenotype at sample index.

Definition at line 274 of file DgeData.cpp.

6.7.3.9 bool DgeData::LoadData (std::string *countsFile,* std::string *phenoFile,* std::string *normsFile =* " ")

Create a new set of DGE data with a counts file and a phenotype file.

read gene counts, create dummy name for each gene load all counts for this gene as doubles

save this gene's counts to the counts class member variable

get min and max sample counts, and sample zeroes

read phenotypes

Definition at line 31 of file DgeData.cpp.

6.7.3.10 void **DgeData::PrintSampleStats** ( )

Print the Sample statistics to the console.

Definition at line 288 of file DgeData.cpp.

6.7.4 Member Data Documentation

6.7.4.1 std::vector<std::vector<double> > **DgeData::counts** `[private]`

Digital gene expression counts.

Definition at line 54 of file DgeData.h.

6.7.4.2 std::string **DgeData::countsFilename** `[private]`

Filename containing DGE counts.

Definition at line 42 of file DgeData.h.

6.7.4.3 std::vector<std::string> **DgeData::geneNames** `[private]`

Gene names.

Definition at line 52 of file DgeData.h.

6.7.4.4 bool **DgeData::hasNormFactors** `[private]`

Are we using normalization?

Definition at line 48 of file DgeData.h.

6.7.4.5 std::vector<std::pair<double, double> > **DgeData::minMaxGeneCounts** `[private]`

Min and max count for genes.

Definition at line 60 of file DgeData.h.

6.7.4.6 std::vector<std::pair<double, double> > DgeData::minMaxSampleCounts [private]

Min and max values for samples.

Definition at line 62 of file DgeData.h.

6.7.4.7 std::vector<double> DgeData::normFactors [private]

Vector of (optional) normalization factors for each sample.

Definition at line 50 of file DgeData.h.

6.7.4.8 std::string DgeData::normsFilename [private]

Filename containing DGE normalization factors.

Definition at line 46 of file DgeData.h.

6.7.4.9 std::string DgeData::phenosFilename [private]

Filename containing DGE phenotypes.

Definition at line 44 of file DgeData.h.

6.7.4.10 std::vector<int> DgeData::phenotypes [private]

Sample phenotypes.

Definition at line 58 of file DgeData.h.

6.7.4.11 std::vector<std::string> DgeData::sampleNames [private]

Sample names.

Definition at line 56 of file DgeData.h.

6.7.4.12 std::vector<std::vector<int> > DgeData::sampleZeroes [private]

Zero count sample indices.

Definition at line 64 of file DgeData.h.

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

- src/library/DgeData.h
- src/library/DgeData.cpp

6.8 insilico::do_to_lower< charT > Class Template Reference

```
#include <StringUtils.h>
```

Public Member Functions

- do_to_lower (std::ctype< charT > &ct)
- do_to_lower (const std::locale &loc=std::locale())
- charT operator() (charT c) const

Private Attributes

- std::ctype< charT > const & m_ctype

6.8.1 Detailed Description

template<class charT = char>class insilico::do_to_lower< charT >

Definition at line 79 of file StringUtils.h.

6.8.2 Constructor & Destructor Documentation

6.8.2.1 template<class charT = char> insilico::do_to_lower< charT >::do_to_lower (std::ctype< charT > & *ct*) `[inline]`

Definition at line 83 of file StringUtils.h.

6.8.2.2 template<class charT = char> insilico::do_to_lower< charT >::do_to_lower (const std::locale & *loc* = `std::locale()`) `[inline]`

Definition at line 86 of file StringUtils.h.

6.8.3 Member Function Documentation

6.8.3.1 template<class charT = char> charT insilico::do_to_lower< charT >::operator() (charT *c*) const `[inline]`

Definition at line 89 of file StringUtils.h.

6.8.4 Member Data Documentation

6.8.4.1 template<class charT = char> std::ctype<charT> const& insilico::do_to_lower< charT >::m_ctype [private]

Definition at line 93 of file StringUtils.h.

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

- src/library/StringUtils.h

6.9 insilico::do_to_upper< charT > Class Template Reference

```
#include <StringUtils.h>
```

Public Member Functions

- do_to_upper (std::ctype< charT > &ct)
- do_to_upper (const std::locale &loc=std::locale())
- charT operator() (charT c) const

Private Attributes

- std::ctype< charT > const & m_ctype

6.9.1 Detailed Description

template<class charT = char>class insilico::do_to_upper< charT >

Definition at line 59 of file StringUtils.h.

6.9.2 Constructor & Destructor Documentation

6.9.2.1 template<class charT = char> insilico::do_to_upper< charT >::do_to_upper (std::ctype< charT > & *ct*) [inline]

Definition at line 63 of file StringUtils.h.

6.9.2.2 template<class charT = char> insilico::do_to_upper< charT >::do_to_upper (const std::locale & *loc* = std::locale()) [inline]

Definition at line 66 of file StringUtils.h.

6.9.3 Member Function Documentation

6.9.3.1 template<class charT = char> charT **insilico::do_to_upper**< charT >::operator() (charT *c*) const [inline]

Definition at line 69 of file StringUtils.h.

6.9.4 Member Data Documentation

6.9.4.1 template<class charT = char> std::ctype<charT> const& **insilico::do_to_upper**< charT >::**m_ctype** [private]

Definition at line 73 of file StringUtils.h.

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

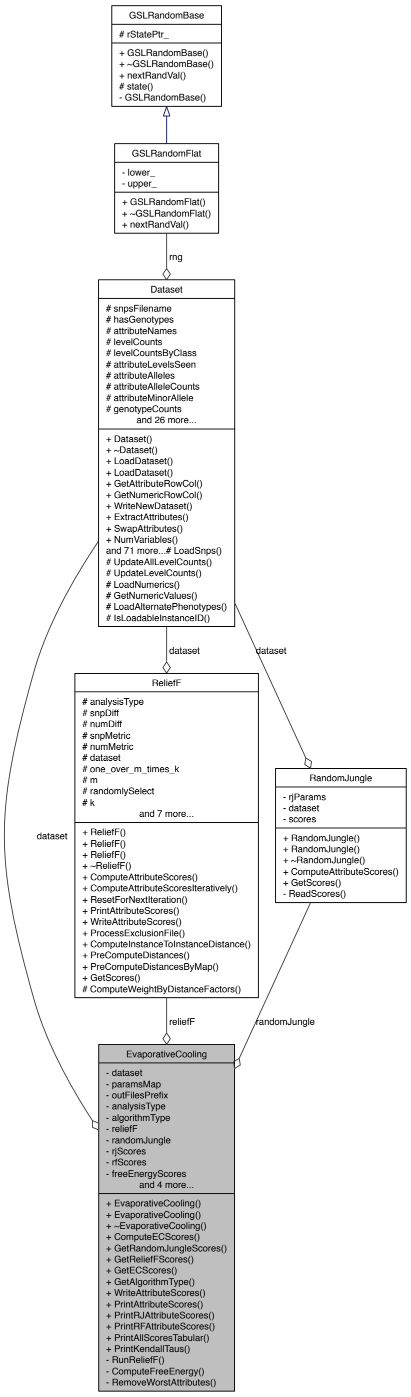
- src/library/StringUtils.h

6.10 EvaporativeCooling Class Reference

Evaporative Cooling attribute ranking algorithm.

`#include <EvaporativeCooling.h>`

Collaboration diagram for EvaporativeCooling:



Public Member Functions

- EvaporativeCooling (Dataset ∗ds, po::variables_map &vm, AnalysisType anaType=SNP_ONLY_ANALYSIS)

  Construct an EC algorithm object.
- EvaporativeCooling (Dataset ∗ds, ConfigMap &configMap, AnalysisType anaType=SNP_ONLY_ANALYSIS)

  Construct an EC algorithm object.
- virtual ∼EvaporativeCooling ()
- bool ComputeECScores ()

  Compute the EC scores based on the current set of attributes.
- EcScores & GetRandomJungleScores ()

  Get the last computed RandomJungle scores.
- EcScores & GetReliefFScores ()

  Get the last computed ReliefF scores.
- EcScores & GetECScores ()

  Get the last computed EC scores.
- EcAlgorithmType GetAlgorithmType ()

  Return the algorithm type: EC_ALL, EC_RJ or EC_RF.
- void WriteAttributeScores (std::string baseFilename)

  Write the scores and attribute names to file.
- void PrintAttributeScores (std::ofstream &outStream)

  Write the EC scores and attribute names to stream.
- void PrintRJAttributeScores (std::ofstream &outStream)

  Write the RJ scores and attribute names to stream.
- void PrintRFAttributeScores (std::ofstream &outStream)

  Write the RF scores and attribute names to stream.
- bool PrintAllScoresTabular ()

  Print the current attributes scores to stdout in tab-delimited format.
- bool PrintKendallTaus ()

  Print the kendall taus between the ReliefF and RandomJungle scores.

Private Member Functions

- bool RunReliefF ()

  Run the ReliefF algorithm.
- bool ComputeFreeEnergy (double temperature)

  Compute the attributes' free energy using the couple temperature.
- bool RemoveWorstAttributes (unsigned int numToRemove=1)

  Remove the worst attribute based on free energy scores.

Private Attributes

- Dataset ∗ dataset

  pointer to a Dataset object
- po::variables_map paramsMap

  command line parameters map
- std::string outFilesPrefix

  prefix for all output files
- AnalysisType analysisType

  type of analysis to perform
- EcAlgorithmType algorithmType

  algorithm steps to perform
- ReliefF ∗ reliefF

  pointer to a ReliefF or RReliefF algorithm object
- RandomJungle ∗ randomJungle

  pointer to a RandomJungle algorithm onject
- EcScores rjScores

  current random jungle scores
- EcScores rfScores

  current relieff scores
- EcScores freeEnergyScores

  current free energy scores
- unsigned int numRFThreads
- unsigned int numToRemovePerIteration

  number of attributes to remove per iteration
- unsigned int numTargetAttributes

  number of target attributes
- EcScores evaporatedAttributes

  attributes that have been evaporated so far
- EcScores ecScores

  current set of ec scores

6.10.1 Detailed Description

Evaporative Cooling attribute ranking algorithm.

Implements the Evaporative Cooling algorithm in: McKinney, et. al. "Capturing the Spectrum of Interaction Effects in Genetic Association Studies by Simulated Evaporative Cooling Network Analysis." PLoS Genetics, Vol 5, Issue 3, 2009.

See also

ReliefF
RReliefF
RandomJungle

Author

Bill White

Version

1.0

Contact: bill.c.white@gmail.com Created on: 7/14/11

Definition at line 53 of file EvaporativeCooling.h.

6.10.2 Constructor & Destructor Documentation

6.10.2.1 EvaporativeCooling::EvaporativeCooling (Dataset $*$ *ds*, po::variables_map & *vm*, AnalysisType *anaType* = SNP_ONLY_ANALYSIS)

Construct an EC algorithm object.

Parameters

in	*ds*	pointer to a Dataset object
in	*vm*	reference to a Boost map of command line options
in	*anaType*	analysis type

Definition at line 54 of file EvaporativeCooling.cpp.

6.10.2.2 EvaporativeCooling::EvaporativeCooling (Dataset $*$ *ds*, ConfigMap & *configMap*, AnalysisType *anaType* = SNP_ONLY_ANALYSIS)

Construct an EC algorithm object.

Parameters

in	*ds*	pointer to a Dataset object
in	*configMap*	reference to a ConfigMap (map<string, string>)
in	*anaType*	analysis type

Definition at line 150 of file EvaporativeCooling.cpp.

6.10.2.3 EvaporativeCooling::∼EvaporativeCooling ( ) [virtual]

Definition at line 255 of file EvaporativeCooling.cpp.

6.10.3 Member Function Documentation

6.10.3.1 bool EvaporativeCooling::ComputeECScores ( )

Compute the EC scores based on the current set of attributes.

Definition at line 264 of file EvaporativeCooling.cpp.

6.10.3.2 bool EvaporativeCooling::ComputeFreeEnergy (double *temperature*) [private]

Compute the attributes' free energy using the couple temperature.

Parameters

in	*tempreatire*	coupling temperature T

Returns

distance

Definition at line 626 of file EvaporativeCooling.cpp.

6.10.3.3 EcAlgorithmType EvaporativeCooling::GetAlgorithmType ( )

Return the algorithm type: EC_ALL, EC_RJ or EC_RF.

Definition at line 409 of file EvaporativeCooling.cpp.

6.10.3.4 EcScores & EvaporativeCooling::GetECScores ( )

Get the last computed EC scores.

Definition at line 405 of file EvaporativeCooling.cpp.

6.10.3.5 EcScores & EvaporativeCooling::GetRandomJungleScores ( )

Get the last computed RandomJungle scores.

Definition at line 397 of file EvaporativeCooling.cpp.

6.10.3.6 EcScores & EvaporativeCooling::GetReliefFScores ( )

Get the last computed ReliefF scores.

Definition at line 401 of file EvaporativeCooling.cpp.

6.10.3.7 bool EvaporativeCooling::PrintAllScoresTabular ( )

Print the current attributes scores to stdout in tab-delimited format.

Definition at line 508 of file EvaporativeCooling.cpp.

6.10.3.8 void **EvaporativeCooling::PrintAttributeScores** (std::ofstream & *outStream*)

Write the EC scores and attribute names to stream.

Parameters

`in`	*outStream*	stream to write score-attribute name pairs

Definition at line 413 of file EvaporativeCooling.cpp.

6.10.3.9 bool **EvaporativeCooling::PrintKendallTaus** ( )

Print the kendall taus between the ReliefF and RandomJungle scores.

Definition at line 542 of file EvaporativeCooling.cpp.

6.10.3.10 void **EvaporativeCooling::PrintRFAttributeScores** (std::ofstream & *outStream*)

Write the RF scores and attribute names to stream.

Parameters

`in`	*outStream*	stream to write score-attribute name pairs

Definition at line 430 of file EvaporativeCooling.cpp.

6.10.3.11 void **EvaporativeCooling::PrintRJAttributeScores** (std::ofstream & *outStream*)

Write the RJ scores and attribute names to stream.

Parameters

`in`	*outStream*	stream to write score-attribute name pairs

Definition at line 421 of file EvaporativeCooling.cpp.

6.10.3.12 bool **EvaporativeCooling::RemoveWorstAttributes** (unsigned int *numToRemove* = `1`) `[private]`

Remove the worst attribute based on free energy scores.

Parameters

in	*numTo-Remove*	number of attributes to remove/evaporate

Returns

distance

Definition at line 669 of file EvaporativeCooling.cpp.

6.10.3.13 bool EvaporativeCooling::RunReliefF ( ) [private]

Run the ReliefF algorithm.

Definition at line 584 of file EvaporativeCooling.cpp.

6.10.3.14 void EvaporativeCooling::WriteAttributeScores (std::string *baseFilename*)

Write the scores and attribute names to file.

Parameters

in	*base-Filename*	filename to write score-attribute name pairs

Definition at line 439 of file EvaporativeCooling.cpp.

6.10.4 Member Data Documentation

6.10.4.1 EcAlgorithmType EvaporativeCooling::algorithmType [private]

algorithm steps to perform

Definition at line 133 of file EvaporativeCooling.h.

6.10.4.2 AnalysisType EvaporativeCooling::analysisType [private]

type of analysis to perform

See also

ReliefF

Definition at line 131 of file EvaporativeCooling.h.

6.10.4.3 Dataset∗ EvaporativeCooling::dataset [private]

pointer to a Dataset object

Definition at line 123 of file EvaporativeCooling.h.

6.10.4.4 EcScores EvaporativeCooling::ecScores [private]

current set of ec scores

Definition at line 156 of file EvaporativeCooling.h.

6.10.4.5 EcScores EvaporativeCooling::evaporatedAttributes [private]

attributes that have been evaporated so far

Definition at line 154 of file EvaporativeCooling.h.

6.10.4.6 EcScores EvaporativeCooling::freeEnergyScores [private]

current free energy scores

Definition at line 145 of file EvaporativeCooling.h.

6.10.4.7 unsigned int EvaporativeCooling::numRFThreads [private]

Definition at line 148 of file EvaporativeCooling.h.

6.10.4.8 unsigned int EvaporativeCooling::numTargetAttributes [private]

number of target attributes

Definition at line 152 of file EvaporativeCooling.h.

6.10.4.9 unsigned int EvaporativeCooling::numToRemovePerIteration [private]

number of attributes to remove per iteration

Definition at line 150 of file EvaporativeCooling.h.

6.10.4.10 std::string EvaporativeCooling::outFilesPrefix [private]

prefix for all output files

Definition at line 127 of file EvaporativeCooling.h.

6.10.4.11 po::variables_map EvaporativeCooling::paramsMap `[private]`

command line parameters map

Definition at line 125 of file EvaporativeCooling.h.

6.10.4.12 RandomJungle∗ EvaporativeCooling::randomJungle `[private]`

pointer to a RandomJungle algorithm onject

Definition at line 138 of file EvaporativeCooling.h.

6.10.4.13 ReliefF∗ EvaporativeCooling::reliefF `[private]`

pointer to a ReliefF or RReliefF algorithm object

Definition at line 136 of file EvaporativeCooling.h.

6.10.4.14 EcScores EvaporativeCooling::rfScores `[private]`

current relieff scores

Definition at line 143 of file EvaporativeCooling.h.

6.10.4.15 EcScores EvaporativeCooling::rjScores `[private]`

current random jungle scores

Definition at line 141 of file EvaporativeCooling.h.

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

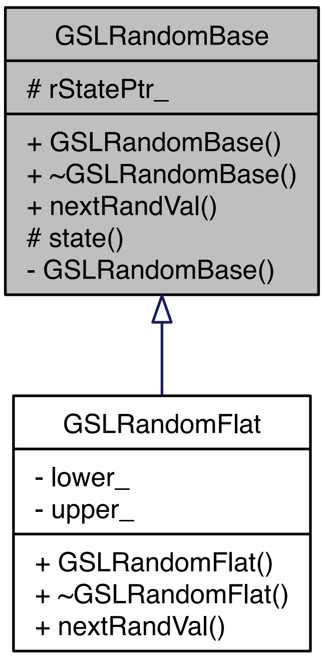
- src/library/EvaporativeCooling.h
- src/library/EvaporativeCooling.cpp

6.11 GSLRandomBase Class Reference

A base class for GNU Scientific Library (GSL) random number functions.

`#include <GSLRandomBase.h>`

Inheritance diagram for GSLRandomBase:



Public Member Functions

- GSLRandomBase (int seedVal)
- virtual ∼GSLRandomBase ()
- virtual double nextRandVal ()=0

Protected Member Functions

- gsl_rng ∗ state ()

Protected Attributes

- gsl_rng ∗ rStatePtr_

Private Member Functions

- GSLRandomBase (const GSLRandomBase &rhs)

6.11.1 Detailed Description

A base class for GNU Scientific Library (GSL) random number functions.

The setup, initialization and clean-up is the same for all GSL random number functions. This class abstracts away these details, placing the stup and initialization in the class constructor and the clean-up in the class destructor. The class constructor is passed a seed value for the random number generator.

A class that provides access to one or more GSL random number functions should be derived from this class. This class must provide an implementation for the nextRandVal() pure virtual function. The nextRandVal will call the specific random number function (for example gsl_ran_ugaussian() for Gaussian distribution or gsl_ran_flat() for a flat random number distribution).

This class uses the default random number generator. At least on Windows XP using the Visual C++ 6.0 compiler the type definitions for the random functions (for example gsl_rng_mt19937 or gsl_rng_knuthran) would not link properly. Perhaps they are not properly exported from the pre-built library.

I decided to use the GSL because is is supported on all major platforms (UNIX, Linux and Windows) and provides high quality pseudo-random number generation support. The standard POSIX rand() function is notorious for its poor quality. While the random() function on UNIX provides better pseudo-random number quality, but is still not as good as functions like MT19937.

Definition at line 39 of file GSLRandomBase.h.

6.11.2 Constructor & Destructor Documentation

6.11.2.1 GSLRandomBase::GSLRandomBase (const GSLRandomBase & *rhs*) `[private]`

6.11.2.2 GSLRandomBase::GSLRandomBase (int *seedVal*) `[inline]`

Definition at line 52 of file GSLRandomBase.h.

6.11.2.3 virtual GSLRandomBase::∼GSLRandomBase ( ) `[inline, virtual]`

Definition at line 67 of file GSLRandomBase.h.

6.11.3 Member Function Documentation

6.11.3.1 virtual double GSLRandomBase::nextRandVal ( ) `[pure virtual]`

Implemented in GSLRandomFlat.

6.11.3.2 gsl_rng∗ GSLRandomBase::state ( ) `[inline, protected]`

Definition at line 45 of file GSLRandomBase.h.

6.11.4 Member Data Documentation

6.11.4.1 gsl_rng∗ GSLRandomBase::rStatePtr_ `[protected]`

Definition at line 48 of file GSLRandomBase.h.

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

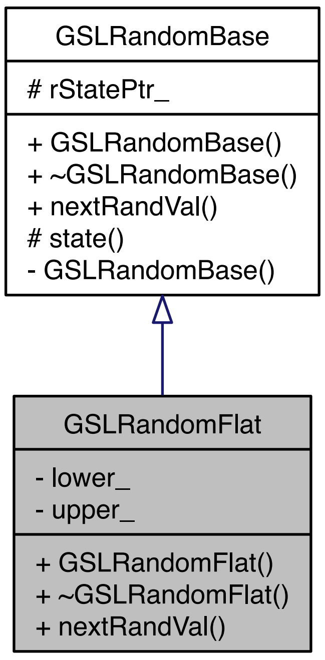
- src/library/GSLRandomBase.h

6.12 GSLRandomFlat Class Reference

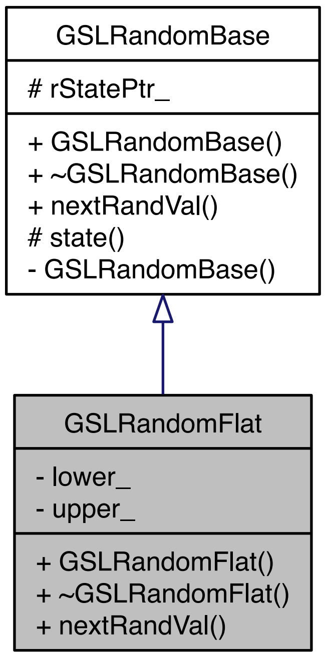
Random numbers in a flat, or uniform distribution.

```
#include <GSLRandomFlat.h>
```

Inheritance diagram for GSLRandomFlat:



Collaboration diagram for GSLRandomFlat:



Public Member Functions

- GSLRandomFlat (int seedVal, double lower, double upper)
- ∼GSLRandomFlat ()
- double nextRandVal ()

Private Attributes

- double lower_
- double upper_

6.12.1 Detailed Description

Random numbers in a flat, or uniform distribution.

The class constructor is given a seed and a lower and upper bound value for the uniform distribution. The random numbers that result will be a uniform distribution in the range

```
    lower <= randVal < upper
```

Definition at line 21 of file GSLRandomFlat.h.

6.12.2 Constructor & Destructor Documentation

6.12.2.1 GSLRandomFlat::GSLRandomFlat (int *seedVal*, double *lower*, double *upper*) `[inline]`

Definition at line 27 of file GSLRandomFlat.h.

6.12.2.2 GSLRandomFlat::~GSLRandomFlat ( ) `[inline]`

Definition at line 36 of file GSLRandomFlat.h.

6.12.3 Member Function Documentation

6.12.3.1 double GSLRandomFlat::nextRandVal ( ) `[inline, virtual]`

Implements GSLRandomBase.

Definition at line 40 of file GSLRandomFlat.h.

6.12.4 Member Data Documentation

6.12.4.1 double GSLRandomFlat::lower_ `[private]`

Definition at line 23 of file GSLRandomFlat.h.

6.12.4.2 double GSLRandomFlat::upper_ `[private]`

Definition at line 23 of file GSLRandomFlat.h.

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

- src/library/GSLRandomFlat.h

6.13 insilico::is_classified< Type, charT > Class Template Reference

```
#include <StringUtils.h>
```

Public Member Functions

- is_classified (std::ctype< charT > &ct)
- is_classified (const std::locale &loc=std::locale())
- bool operator() (charT c) const

Private Attributes

- std::ctype< charT > const & m_ctype

6.13.1 Detailed Description

template<std::ctype_base::mask Type, class charT = char>class insilico::is_classified< Type, charT >

Definition at line 40 of file StringUtils.h.

6.13.2 Constructor & Destructor Documentation

6.13.2.1 template<std::ctype_base::mask Type, class charT = char> insilico::is_classified< Type, charT >::is_classified (std::ctype< charT > & *ct*) [inline]

Definition at line 44 of file StringUtils.h.

6.13.2.2 template<std::ctype_base::mask Type, class charT = char> insilico::is_classified< Type, charT >::is_classified (const std::locale & *loc* = std::locale()) [inline]

Definition at line 47 of file StringUtils.h.

6.13.3 Member Function Documentation

6.13.3.1 template<std::ctype_base::mask Type, class charT = char> bool insilico::is_classified< Type, charT >::operator() (charT *c*) const [inline]

Definition at line 50 of file StringUtils.h.

6.13.4 Member Data Documentation

6.13.4.1 template<std::ctype_base::mask Type, class charT = char> std::ctype<charT> const& insilico::is_classified< Type, charT >::m_ctype [private]

Definition at line 54 of file StringUtils.h.

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

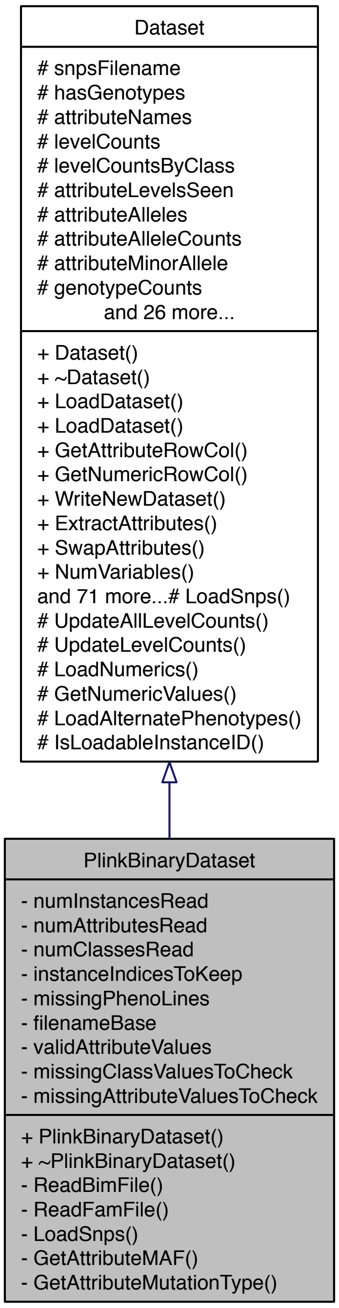
- src/library/StringUtils.h

6.14 PlinkBinaryDataset Class Reference

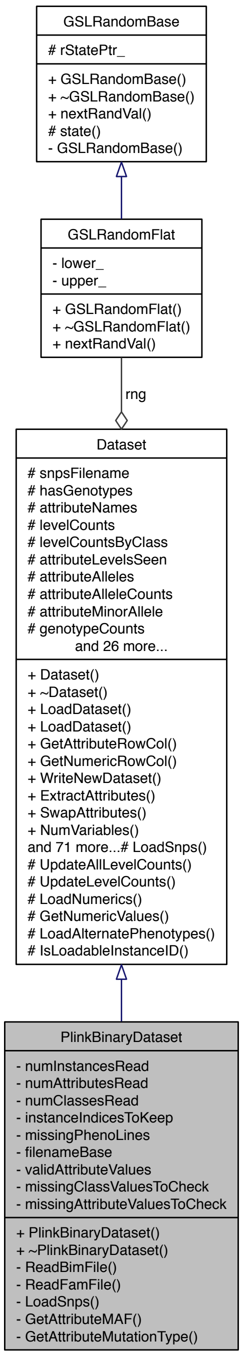
Plink binary PED/BED file format reader.

```
#include <PlinkBinaryDataset.h>
```

Inheritance diagram for PlinkBinaryDataset:



Collaboration diagram for PlinkBinaryDataset:



Public Member Functions

- PlinkBinaryDataset ()
- ∼PlinkBinaryDataset ()

Private Member Functions

- bool ReadBimFile (std::string bimFilename)

  Load attribute information.
- bool ReadFamFile (std::string famFilename)

  Load individual information.
- bool LoadSnps (std::string filename)

  Load SNPs from file using the data set filename.
- std::pair< char, double > GetAttributeMAF (unsigned int attributeIndex)

  Get attribute minor allele and frequency.
- AttributeMutationType GetAttributeMutationType (unsigned int attributeIndex)

  Get attribute mutation type.

Private Attributes

- unsigned int numInstancesRead
- unsigned int numAttributesRead
- unsigned int numClassesRead
- std::vector< int > instanceIndicesToKeep
- std::vector< int > missingPhenoLines
- std::string filenameBase
- std::vector< std::string > validAttributeValues

  for checking attribute values
- std::vector< std::string > missingClassValuesToCheck

  missing class values
- std::vector< std::string > missingAttributeValuesToCheck

  missing attribute values

6.14.1 Detailed Description

Plink binary PED/BED file format reader.

See also

Dataset

Author

Bill White

Version

1.0

Contact: bill.c.white@gmail.com Created on: 3/10/11

Definition at line 21 of file PlinkBinaryDataset.h.

6.14.2 Constructor & Destructor Documentation

6.14.2.1 PlinkBinaryDataset::PlinkBinaryDataset ( )

Definition at line 36 of file PlinkBinaryDataset.cpp.

6.14.2.2 PlinkBinaryDataset::~PlinkBinaryDataset ( ) [inline]

Definition at line 25 of file PlinkBinaryDataset.h.

6.14.3 Member Function Documentation

6.14.3.1 pair< char, double > PlinkBinaryDataset::GetAttributeMAF (unsigned int *attributeIndex*) [private, virtual]

Get attribute minor allele and frequency.

Parameters

in	*attribute*	index

Returns

pair (minor allele, minor allele frequency)

An Intriduction to Genetic Analysis by Griffiths, Miller, Suzuki, Lewontin and Gelbart, 2000, page 715.

Reimplemented from Dataset.

Definition at line 546 of file PlinkBinaryDataset.cpp.

6.14.3.2 AttributeMutationType PlinkBinaryDataset::GetAttributeMutationType (unsigned int *attributeIndex*) [private, virtual]

Get attribute mutation type.

Parameters

in	*attribute*	index

Returns

mutation type (transition, transversion, unknown)

Reimplemented from Dataset.

Definition at line 555 of file PlinkBinaryDataset.cpp.

6.14.3.3 bool PlinkBinaryDataset::LoadSnps (std::string *filename*) `[private, virtual]`

Load SNPs from file using the data set filename.

Parameters

`in`	*filename*	SNPs filename
`in`	*deRecodeA*	perform a recodeA operation after reading raw data?

Returns

success

------------ Beginning of private methods ------------------ Remove instances that are not in instanceIdsToLoad or marked as missing phenotype - 11/1/11 Only remove missing phenotypes if no alt pheno file - 1/23/12

Passed all tests, so add this instance to the data set

Release memory used by filtered out instances

Open the data file and read line-by-line

Detect the class type

Reimplemented from Dataset.

Definition at line 47 of file PlinkBinaryDataset.cpp.

6.14.3.4 bool PlinkBinaryDataset::ReadBimFile (std::string *bimFilename*) `[private]`

Load attribute information.

Parameters

`in`	*PLINK*	bim filename

Returns

success

set the mutation type

Definition at line 367 of file PlinkBinaryDataset.cpp.

6.14.3.5 bool PlinkBinaryDataset::ReadFamFile (std::string *famFilename*) `[private]`

Load individual information.

Parameters

`in`	*PLIN*	fam filename

Returns

success

Detect the class type

Read attribute information from the fam file

assign class level

Create a new instance for this individual

Definition at line 436 of file PlinkBinaryDataset.cpp.

6.14.4 Member Data Documentation

6.14.4.1 std::string PlinkBinaryDataset::filenameBase `[private]`

Definition at line 50 of file PlinkBinaryDataset.h.

6.14.4.2 std::vector<int> PlinkBinaryDataset::instanceIndicesToKeep `[private]`

Definition at line 47 of file PlinkBinaryDataset.h.

6.14.4.3 std::vector<std::string> PlinkBinaryDataset::missingAttributeValuesTo-Check `[private]`

missing attribute values

Definition at line 57 of file PlinkBinaryDataset.h.

6.14.4.4 std::vector<std::string> PlinkBinaryDataset::missingClassValuesToCheck `[private]`

missing class values

Definition at line 55 of file PlinkBinaryDataset.h.

6.14.4.5 std::vector<int> PlinkBinaryDataset::missingPhenoLines [private]

Definition at line 48 of file PlinkBinaryDataset.h.

6.14.4.6 unsigned int PlinkBinaryDataset::numAttributesRead [private]

Definition at line 44 of file PlinkBinaryDataset.h.

6.14.4.7 unsigned int PlinkBinaryDataset::numClassesRead [private]

Definition at line 45 of file PlinkBinaryDataset.h.

6.14.4.8 unsigned int PlinkBinaryDataset::numInstancesRead [private]

Definition at line 43 of file PlinkBinaryDataset.h.

6.14.4.9 std::vector<std::string> PlinkBinaryDataset::validAttributeValues [private]

for checking attribute values

Definition at line 53 of file PlinkBinaryDataset.h.

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

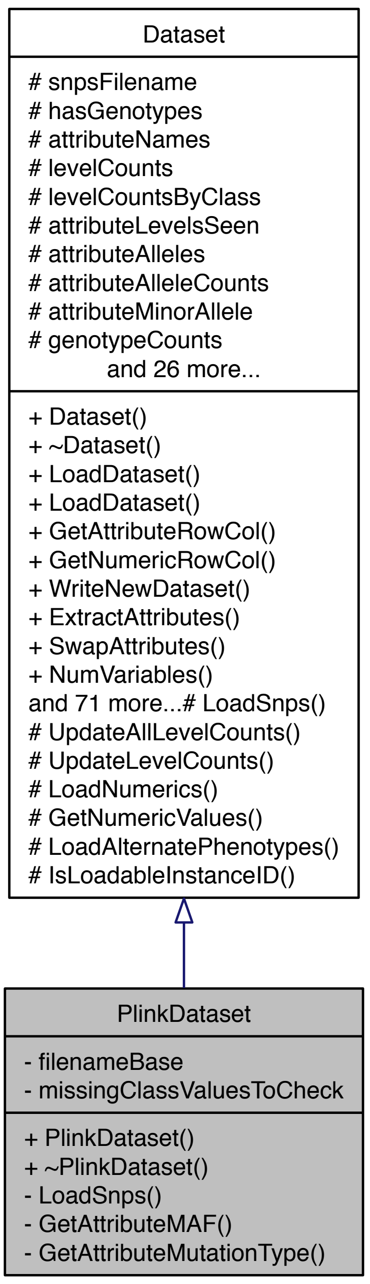
- src/library/PlinkBinaryDataset.h
- src/library/PlinkBinaryDataset.cpp

6.15 PlinkDataset Class Reference

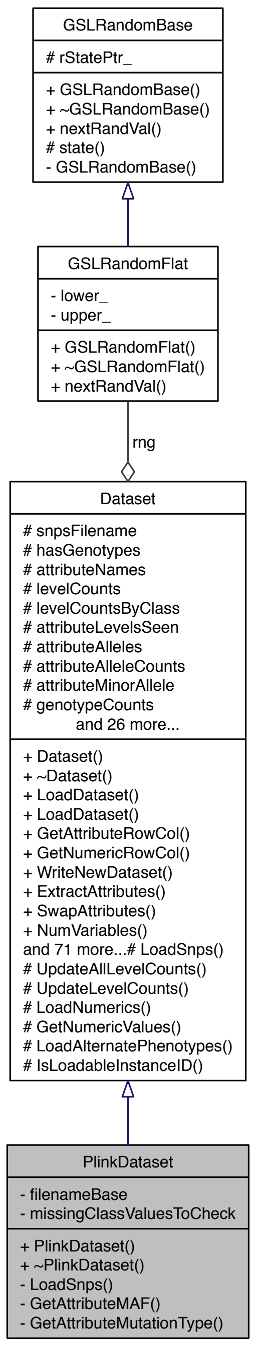
Plink MAP/PED file format reader.

`#include <PlinkDataset.h>`

Inheritance diagram for PlinkDataset:



Collaboration diagram for PlinkDataset:



Public Member Functions

- PlinkDataset ()

    Construct a PLINK data set reader. Calls Dataset base class constructor.
- ∼PlinkDataset ()

Private Member Functions

- bool LoadSnps (std::string filename)

    Load SNPs from file using the data set filename.
- std::pair< char, double > GetAttributeMAF (unsigned int attributeIndex)

    Get attribute minor allele and frequency.
- AttributeMutationType GetAttributeMutationType (unsigned int attributeIndex)

    Get attribute mutation type.

Private Attributes

- std::string filenameBase

    base filename for auxiliary files
- std::vector< std::string > missingClassValuesToCheck

    missing class values

6.15.1 Detailed Description

Plink MAP/PED file format reader.

See also

Dataset

Author

Bill White

Version

1.0

Contact: `bill.c.white@gmail.com` Created on: 2/1/11

Definition at line 35 of file PlinkDataset.h.

6.15.2 Constructor & Destructor Documentation

6.15.2.1 PlinkDataset::PlinkDataset ( )

Construct a PLINK data set reader. Calls Dataset base class constructor.

Definition at line 26 of file PlinkDataset.cpp.

6.15.2.2 PlinkDataset::∼PlinkDataset ( ) `[inline]`

Definition at line 40 of file PlinkDataset.h.

6.15.3 Member Function Documentation

6.15.3.1 pair< char, double > PlinkDataset::GetAttributeMAF (unsigned int *attributeIndex*) `[private, virtual]`

Get attribute minor allele and frequency.

Parameters

`in`	*attribute*	index

Returns

pair (minor allele, minor allele frequency)

An Intriduction to Genetic Analysis by Griffiths, Miller, Suzuki, Lewontin and Gelbart, 2000, page 715.

Reimplemented from Dataset.

Definition at line 383 of file PlinkDataset.cpp.

6.15.3.2 AttributeMutationType PlinkDataset::GetAttributeMutationType (unsigned int *attributeIndex*) `[private, virtual]`

Get attribute mutation type.

Parameters

`in`	*attribute*	index

Returns

mutation type (transition, transversion, unknown)

Reimplemented from Dataset.

Definition at line 392 of file PlinkDataset.cpp.

6.15.3.3 bool PlinkDataset::LoadSnps (std::string *filename*) [private, virtual]

Load SNPs from file using the data set filename.

Parameters

in	*filename*	SNPs filename
in	*deRecodeA*	perform a recodeA operation after reading raw data?

Returns

success

------------ Beginning of private methods ------------------ read attribute information from the map file

Detect the class type

read attribute values from the ped file

determine the MAP file type

get ID for matching between PLINK data, numeric and pheno files

assign class level

set the mutation type

Open the data file and read line-by-line

Detect the class type

Reimplemented from Dataset.

Definition at line 31 of file PlinkDataset.cpp.

6.15.4 Member Data Documentation

6.15.4.1 std::string PlinkDataset::filenameBase [private]

base filename for auxiliary files

Definition at line 47 of file PlinkDataset.h.

6.15.4.2 std::vector<std::string> PlinkDataset::missingClassValuesToCheck [private]

missing class values

Definition at line 49 of file PlinkDataset.h.

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

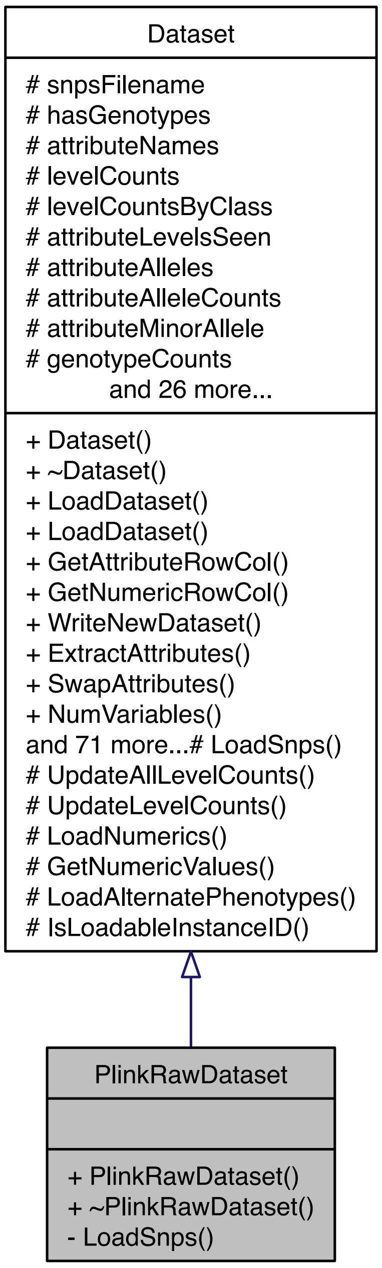
- src/library/PlinkDataset.h
- src/library/PlinkDataset.cpp

6.16 PlinkRawDataset Class Reference

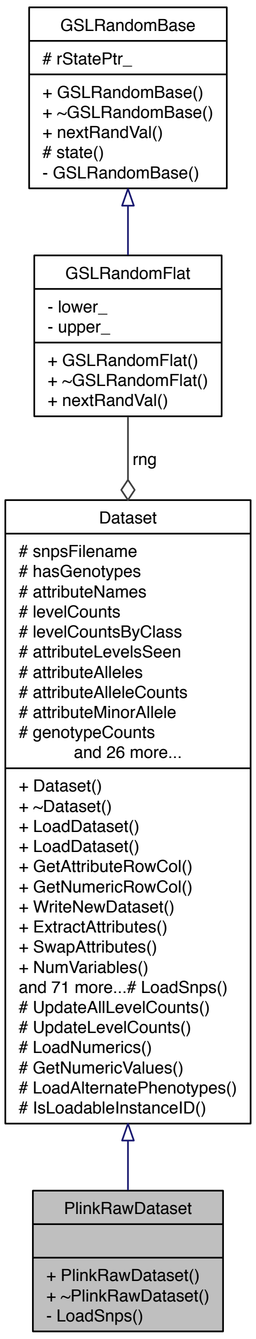
Plink recodeA/RAW file format reader.

```
#include <PlinkRawDataset.h>
```

Inheritance diagram for PlinkRawDataset:



Collaboration diagram for PlinkRawDataset:



Public Member Functions

- PlinkRawDataset ()
- ∼PlinkRawDataset ()

Private Member Functions

- bool LoadSnps (std::string filename)

  Load SNPs from file using the data set filename.

6.16.1 Detailed Description

Plink recodeA/RAW file format reader.

See also

Dataset

Author

Bill White

Version

1.0

Contact: `bill.c.white@gmail.com` Created on: 2/24/11

Definition at line 23 of file PlinkRawDataset.h.

6.16.2 Constructor & Destructor Documentation

6.16.2.1 PlinkRawDataset::PlinkRawDataset ( )

Definition at line 22 of file PlinkRawDataset.cpp.

6.16.2.2 PlinkRawDataset::∼PlinkRawDataset ( ) `[inline]`

Definition at line 27 of file PlinkRawDataset.h.

6.16.3 Member Function Documentation

6.16.3.1 bool PlinkRawDataset::LoadSnps (std::string *filename*) `[private, virtual]`

Load SNPs from file using the data set filename.

Parameters

in	*filename*	SNPs filename
in	*deRecodeA*	perform a recodeA operation after reading raw data?

Returns

success

------------ Beginning of private methods ------------------ Detect the class type

Open the data file and read line-by-line

Detect the class type

Reimplemented from Dataset.

Definition at line 25 of file PlinkRawDataset.cpp.

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

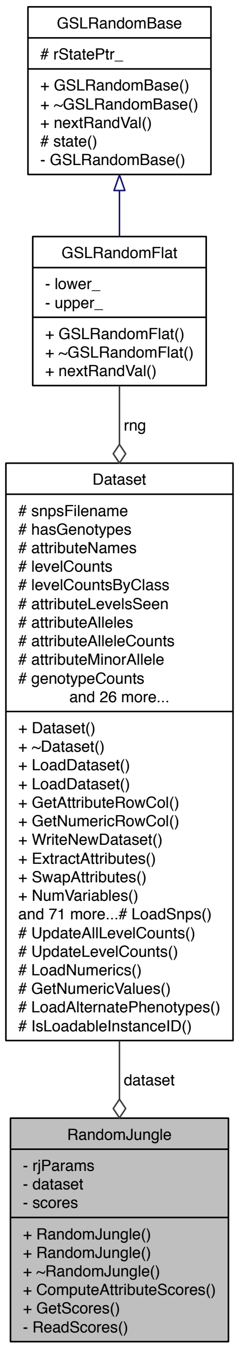
- src/library/PlinkRawDataset.h
- src/library/PlinkRawDataset.cpp

6.17 RandomJungle Class Reference

RandomJungle attribute ranking algorithm.

`#include <RandomJungle.h>`

Collaboration diagram for RandomJungle:



Public Member Functions

- RandomJungle (Dataset *ds, po::variables_map &vm)

  Construct an RandomJungle algorithm object.
- RandomJungle (Dataset *ds, ConfigMap &vm)

  Construct an RandomJungle algorithm object.
- virtual ~RandomJungle ()
- bool ComputeAttributeScores ()

  Score attributes by getting Random Jungle importance scores.
- std::vector< std::pair< double, std::string > > GetScores ()

  Get the (importance) scores as a vector of pairs: score, attribute name.

Private Member Functions

- bool ReadScores (std::string importanceFilename)

  Read the importance scores as attribute rankings from file.

Private Attributes

- RJunglePar rjParams

  RandomJungle parameters object.
- Dataset * dataset

  pointer to a Dataset object
- std::vector< std::pair< double, std::string > > scores

  vector of pairs: scores, attribute names

6.17.1 Detailed Description

RandomJungle attribute ranking algorithm.

Adapter class to map EC call for Random Jungle importance scores to Random Jungle library functions.

Author

Bill White

Version

1.0

Contact: `bill.c.white@gmail.com` Created on: 10/16/11

Definition at line 32 of file RandomJungle.h.

6.17.2 Constructor & Destructor Documentation

6.17.2.1 RandomJungle::RandomJungle (Dataset ∗ *ds*, po::variables_map & *vm*)

Construct an RandomJungle algorithm object.

Parameters

in	*ds*	pointer to a Dataset object
in	*vm*	reference to a Boost map of command line options

Definition at line 33 of file RandomJungle.cpp.

6.17.2.2 RandomJungle::RandomJungle (Dataset ∗ *ds*, ConfigMap & *vm*)

Construct an RandomJungle algorithm object.

Parameters

in	*ds*	pointer to a Dataset object
in	*configMap*	reference ConfigMap (map<string, string>)

Definition at line 70 of file RandomJungle.cpp.

6.17.2.3 RandomJungle::∼RandomJungle ( ) [virtual]

Definition at line 114 of file RandomJungle.cpp.

6.17.3 Member Function Documentation

6.17.3.1 bool RandomJungle::ComputeAttributeScores ( )

Score attributes by getting Random Jungle importance scores.

Definition at line 120 of file RandomJungle.cpp.

6.17.3.2 vector< pair< double, string > > RandomJungle::GetScores ( )

Get the (importance) scores as a vector of pairs: score, attribute name.

Returns

vector of pairs

Definition at line 382 of file RandomJungle.cpp.

6.17.3.3 bool RandomJungle::ReadScores (std::string *importanceFilename*) `[private]`

Read the importance scores as attribute rankings from file.

Definition at line 386 of file RandomJungle.cpp.

6.17.4 Member Data Documentation

6.17.4.1 Dataset∗ RandomJungle::dataset `[private]`

pointer to a Dataset object

Definition at line 61 of file RandomJungle.h.

6.17.4.2 RJunglePar RandomJungle::rjParams `[private]`

RandomJungle parameters object.

Definition at line 59 of file RandomJungle.h.

6.17.4.3 std::vector<std::pair<double, std::string> > RandomJungle::scores `[private]`

vector of pairs: scores, attribute names

Definition at line 63 of file RandomJungle.h.

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

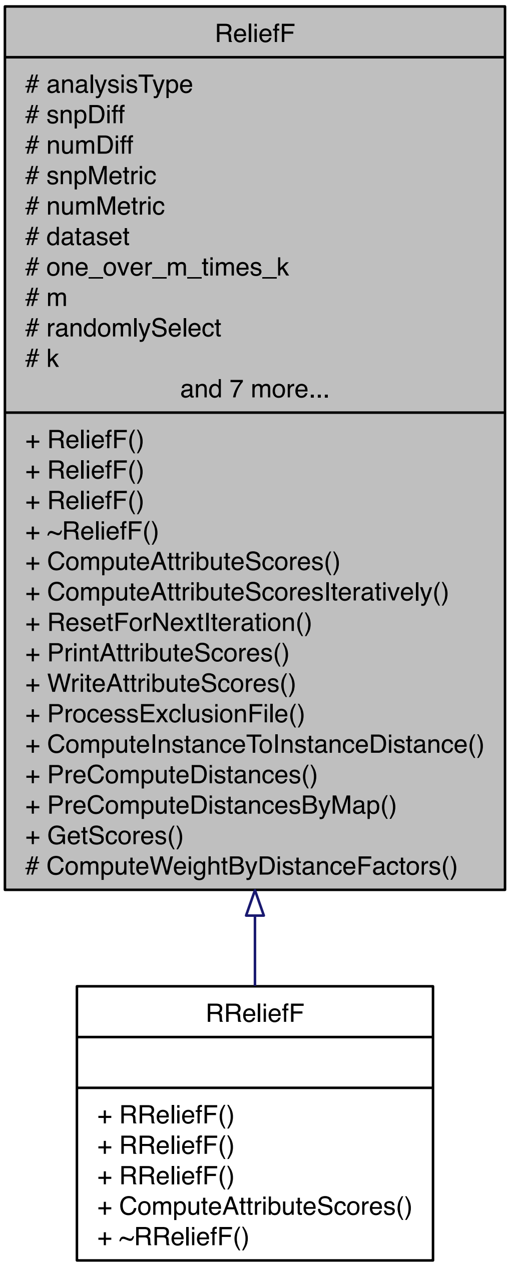
- src/library/RandomJungle.h
- src/library/RandomJungle.cpp

6.18 ReliefF Class Reference

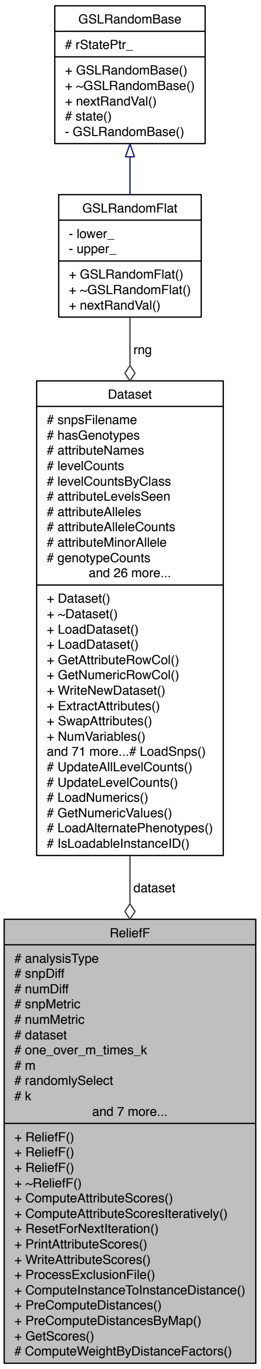
ReliefF attribute ranking algorithm.

`#include <ReliefF.h>`

Inheritance diagram for ReliefF:



Collaboration diagram for ReliefF:



Public Member Functions

- ReliefF (Dataset ∗ds, AnalysisType anaType)

  Construct an ReliefF algorithm object.
- ReliefF (Dataset ∗ds, po::variables_map &vm, AnalysisType anaType)

  Construct an ReliefF algorithm object.
- ReliefF (Dataset ∗ds, ConfigMap &vm, AnalysisType anaType)

  Construct an ReliefF algorithm object.
- virtual ∼ReliefF ()
- virtual bool ComputeAttributeScores ()

  Compute the ReliefF scores for the current set of attributes.
- bool ComputeAttributeScoresIteratively ()

  Compute the ReliefF scores by iteratively removing worst attributes.
- bool ResetForNextIteration ()

  Resets some data structures for the next iteration of ReliefF.
- void PrintAttributeScores (std::ofstream &outStream)

  Write the scores and attribute names to stream.
- void WriteAttributeScores (std::string baseFilename)

  Write the scores and attribute names to file.
- bool ProcessExclusionFile (std::string exclusionFilename)

  Remove file of attribute names from consideration in ReliefF.
- double ComputeInstanceToInstanceDistance (DatasetInstance ∗dsi1, Dataset-Instance ∗dsi2)

  Compute the distance between two DatasetInstances.
- bool PreComputeDistances ()

  Precompute all pairwise instance-to-instance distances.
- bool PreComputeDistancesByMap ()

  Precompute all pairwise distances homoring excluded instances.
- std::vector< std::pair< double, std::string > > GetScores ()

  Get the last computed ReliefF scores.

Protected Member Functions

- bool ComputeWeightByDistanceFactors ()

  Compute the weight by distance factors for nearest neighbors.

Protected Attributes

- AnalysisType analysisType

  type of analysis to perform
- double(∗ snpDiff)(unsigned int attributeIndex, DatasetInstance ∗dsi1, Dataset-Instance ∗dsi2)

  Compute the discrete difference in an attribute between two instances.

- double(∗ numDiff)(unsigned int attributeIndex, DatasetInstance ∗dsi1, DatasetInstance ∗dsi2)

  Compute the continuous difference in an attribute between two instances.
- std::string snpMetric

  the name of discrete diff(erence) function
- std::string numMetric

  the name of continuous diff(erence) function
- Dataset ∗ dataset

  the dataset on which the algorithm is working
- double one_over_m_times_k

  nomalizing factor for ReliefF $m * k$ loop
- unsigned int m

  number of instances to sample
- bool randomlySelect

  are instances being randomly selected?
- unsigned int k

  k nearest neighbors
- unsigned int removePerIteration

  number of attributes to remove each iteration if running iteratively
- bool doRemovePercent

  are we removing a percentage per iteration?
- double removePercentage

  percentage of attributes to remove per iteration if running iteratively
- std::string weightByDistanceMethod

  name of the weight-by-distance method
- double weightByDistanceSigma

  sigma value used in exponential decay weight-by-distance
- std::vector< double > W

  attribute scores/weights
- std::vector< std::string > scoreNames

  attribute names associated with scores
- std::map< std::string, double > finalScores

  final scores after all iterations

6.18.1 Detailed Description

ReliefF attribute ranking algorithm.

Totally redone for the McKinney insilico lab in 2011. Large refactoring to move all attribute elimination handling to the Dataset and its subclasses. 9/11/11

See also

RReliefF

Author

Bill White

Version

1.0

Contact: `bill.c.white@gmail.com` Created on: 7/16/05

Definition at line 32 of file ReliefF.h.

6.18.2 Constructor & Destructor Documentation

6.18.2.1 ReliefF::ReliefF (Dataset * *ds*, AnalysisType *anaType*)

Construct an ReliefF algorithm object.

Parameters

`in`	*ds*	pointer to a Dataset object
`in`	*anaType*	analysis type

Definition at line 69 of file ReliefF.cpp.

6.18.2.2 ReliefF::ReliefF (Dataset * *ds*, po::variables_map & *vm*, AnalysisType *anaType*)

Construct an ReliefF algorithm object.

Parameters

`in`	*ds*	pointer to a Dataset object
`in`	*vm*	reference to a Boost map of command line options
`in`	*anaType*	analysis type

Definition at line 135 of file ReliefF.cpp.

6.18.2.3 ReliefF::ReliefF (Dataset * *ds*, ConfigMap & *vm*, AnalysisType *anaType*)

Construct an ReliefF algorithm object.

Parameters

`in`	*ds*	pointer to a Dataset object
`in`	*configMap*	reference to a ConfigMap (map<string, string>)
`in`	*anaType*	analysis type

Definition at line 275 of file ReliefF.cpp.

6.18.2.4 ReliefF::∼ReliefF ( ) `[virtual]`

Definition at line 426 of file ReliefF.cpp.

6.18.3 Member Function Documentation

6.18.3.1 bool ReliefF::ComputeAttributeScores ( ) `[virtual]`

Compute the ReliefF scores for the current set of attributes.

Implements ReliefF algorithm: Marko Robnik-Sikonja, Igor Kononenko: Theoretical and Empirical Analysis of ReliefF and RReliefF. Machine Learning Journal, 53:23-69, 2003 `http://lkm.fri.uni-lj.si/rmarko/papers/robnik03-mlj.pdf` algorithm line 1

algorithm line 2

algorithm lines 4, 5 and 6

algorithm line 7

algorithm line 8

algorithm line 9

Reimplemented in RReliefF.

Definition at line 429 of file ReliefF.cpp.

6.18.3.2 bool ReliefF::ComputeAttributeScoresIteratively ( )

Compute the ReliefF scores by iteratively removing worst attributes.

Definition at line 600 of file ReliefF.cpp.

6.18.3.3 double ReliefF::ComputeInstanceToInstanceDistance (DatasetInstance ∗ *dsi1*, DatasetInstance ∗ *dsi2*)

Compute the distance between two DatasetInstances.

Parameters

`in`	*dsi1*	pointer to DatasetInstance 1
`in`	*dsi2*	pointer to DatasetInstance 2

Returns

distance

Definition at line 738 of file ReliefF.cpp.

6.18.3.4 bool ReliefF::ComputeWeightByDistanceFactors ( ) [protected]

Compute the weight by distance factors for nearest neighbors.

Definition at line 1029 of file ReliefF.cpp.

6.18.3.5 vector< pair< double, string > > ReliefF::GetScores ( )

Get the last computed ReliefF scores.

Definition at line 1015 of file ReliefF.cpp.

6.18.3.6 bool ReliefF::PreComputeDistances ( )

Precompute all pairwise instance-to-instance distances.

Definition at line 770 of file ReliefF.cpp.

6.18.3.7 bool ReliefF::PreComputeDistancesByMap ( )

Precompute all pairwise distances homoring excluded instances.

Definition at line 903 of file ReliefF.cpp.

6.18.3.8 void ReliefF::PrintAttributeScores (std::ofstream & *outStream*)

Write the scores and attribute names to stream.

Parameters

in	*outStream*	stream to write score-attribute name pairs

Definition at line 681 of file ReliefF.cpp.

6.18.3.9 bool ReliefF::ProcessExclusionFile (std::string *exclusionFilename*)

Remove file of attribute names from consideration in ReliefF.

Parameters

in	*excusion-Filename*	filename of attributes to exclude

Returns

success

Definition at line 712 of file ReliefF.cpp.

6.18.3.10 bool ReliefF::ResetForNextIteration ( )

Resets some data structures for the next iteration of ReliefF.

Definition at line 674 of file ReliefF.cpp.

6.18.3.11 void ReliefF::WriteAttributeScores (std::string *baseFilename*)

Write the scores and attribute names to file.

Parameters

in	*baseF-ilename*	filename to write score-attribute name pairs

Definition at line 692 of file ReliefF.cpp.

6.18.4 Member Data Documentation

6.18.4.1 AnalysisType ReliefF::analysisType [protected]

type of analysis to perform

Definition at line 102 of file ReliefF.h.

6.18.4.2 Dataset∗ ReliefF::dataset [protected]

the dataset on which the algorithm is working

Definition at line 128 of file ReliefF.h.

6.18.4.3 bool ReliefF::doRemovePercent [protected]

are we removing a percentage per iteration?

Definition at line 140 of file ReliefF.h.

6.18.4.4 std::map<std::string, double> ReliefF::finalScores [protected]

final scores after all iterations

Definition at line 153 of file ReliefF.h.

6.18.4.5 unsigned int ReliefF::k [protected]

k nearest neighbors

Definition at line 136 of file ReliefF.h.

6.18.4.6 unsigned int ReliefF::m [protected]

number of instances to sample

Definition at line 132 of file ReliefF.h.

6.18.4.7 double(∗ ReliefF::numDiff)(unsigned int attributeIndex, DatasetInstance ∗dsi1, DatasetInstance ∗dsi2) [protected]

Compute the continuous difference in an attribute between two instances.

Parameters

in	*attribute-Index*	index into vector of all attributes
in	*dsi1*	pointer to DatasetInstance 1
in	*dsi2*	pointer to DatasetInstance 2

Returns

diff(erence)

Definition at line 120 of file ReliefF.h.

6.18.4.8 std::string ReliefF::numMetric [protected]

the name of continuous diff(erence) function

Definition at line 126 of file ReliefF.h.

6.18.4.9 double ReliefF::one_over_m_times_k [protected]

nomalizing factor for ReliefF m ∗ k loop

Definition at line 130 of file ReliefF.h.

6.18.4.10 bool ReliefF::randomlySelect [protected]

are instances being randomly selected?

Definition at line 134 of file ReliefF.h.

6.18.4.11 double ReliefF::removePercentage [protected]

percentage of attributes to remove per iteration if running iteratively

Definition at line 142 of file ReliefF.h.

6.18.4.12 unsigned int ReliefF::removePerIteration [protected]

number of attributes to remove each iteration if running iteratively

Definition at line 138 of file ReliefF.h.

6.18.4.13 std::vector<std::string> ReliefF::scoreNames [protected]

attribute names associated with scores

Definition at line 151 of file ReliefF.h.

6.18.4.14 double(∗ ReliefF::snpDiff)(unsigned int attributeIndex, DatasetInstance ∗dsi1, DatasetInstance ∗dsi2) [protected]

Compute the discrete difference in an attribute between two instances.

Parameters

in	*attribute-Index*	index into vector of all attributes
in	*dsi1*	pointer to DatasetInstance 1
in	*dsi2*	pointer to DatasetInstance 2

Returns

diff(erence)

Definition at line 110 of file ReliefF.h.

6.18.4.15 std::string ReliefF::snpMetric [protected]

the name of discrete diff(erence) function

Definition at line 124 of file ReliefF.h.

6.18.4.16 std::vector<double> ReliefF::W [protected]

attribute scores/weights

Definition at line 149 of file ReliefF.h.

6.18.4.17 std::string ReliefF::weightByDistanceMethod [protected]

name of the weight-by-distance method

Definition at line 144 of file ReliefF.h.

6.18.4.18 double ReliefF::weightByDistanceSigma [protected]

sigma value used in exponential decay weight-by-distance

Definition at line 146 of file ReliefF.h.

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

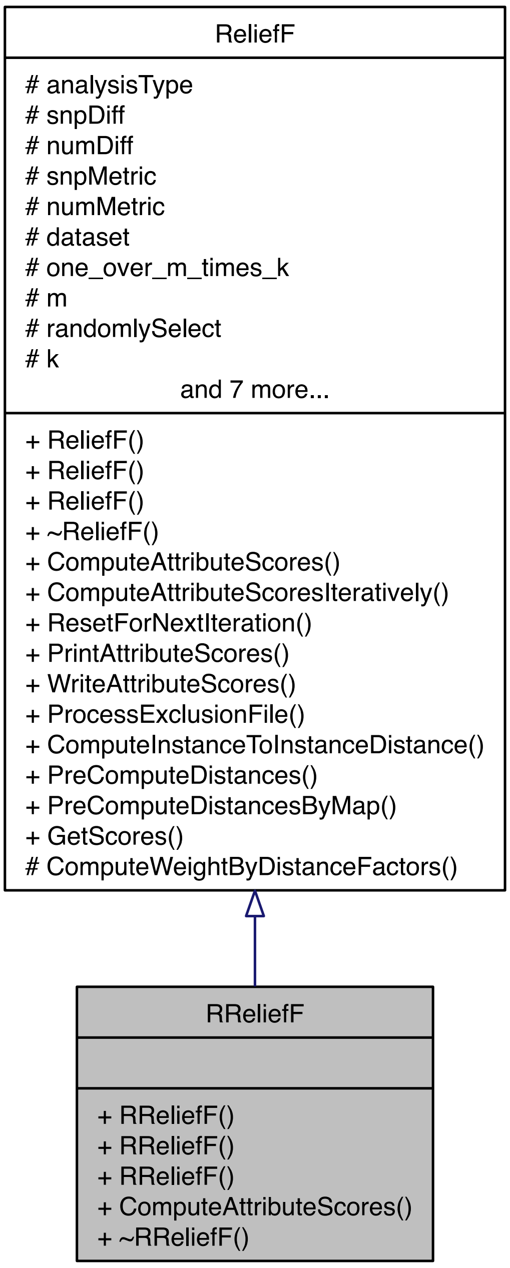
- src/library/ReliefF.h
- src/library/ReliefF.cpp

6.19 RReliefF Class Reference

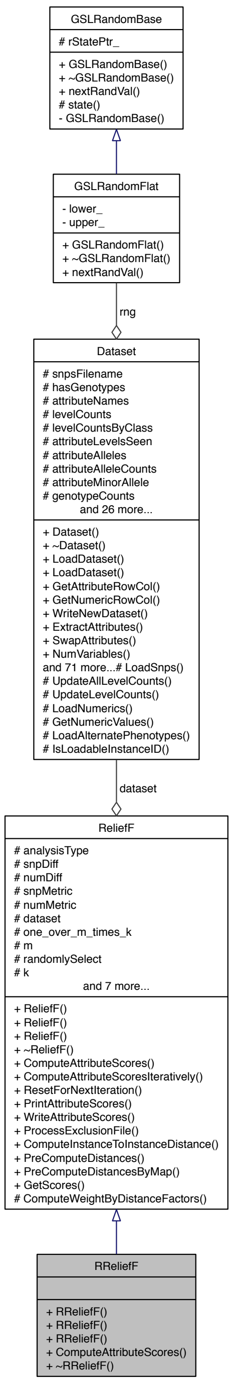
Regression ReliefF attribute ranking algorithm.

`#include <RReliefF.h>`

Inheritance diagram for RReliefF:



Collaboration diagram for RReliefF:



Public Member Functions

- RReliefF (Dataset ∗ds)

  Construct an ReliefF algorithm object.
- RReliefF (Dataset ∗ds, po::variables_map &vm)

  Construct an ReliefF algorithm object.
- RReliefF (Dataset ∗ds, ConfigMap &configMap)

  Construct an ReliefF algorithm object.
- bool ComputeAttributeScores ()

  Compute the ReliefF scores for the current set of attributes.
- virtual ∼RReliefF ()

6.19.1 Detailed Description

Regression ReliefF attribute ranking algorithm.

Totally redone for the McKinney insilico lab in 2011. Large refactoring to move all attribute elimination handling to the Dataset and its subclasses. 9/11/11

See also

ReliefF

Author

Bill White

Version

1.0

Contact: `bill.c.white@gmail.com` Created on: 9/27/11

Definition at line 33 of file RReliefF.h.

6.19.2 Constructor & Destructor Documentation

6.19.2.1 RReliefF::RReliefF (Dataset ∗ *ds*)

Construct an ReliefF algorithm object.

Parameters

`in`	*ds*	pointer to a Dataset object

Definition at line 19 of file RReliefF.cpp.

6.19.2.2 RReliefF::RReliefF (Dataset ∗ *ds*, po::variables_map & *vm*)

Construct an ReliefF algorithm object.

Parameters

in	*ds*	pointer to a Dataset object
in	*vm*	reference to a Boost map of command line options

Definition at line 29 of file RReliefF.cpp.

6.19.2.3 RReliefF::RReliefF (Dataset ∗ *ds*, ConfigMap & *configMap*)

Construct an ReliefF algorithm object.

Parameters

in	*ds*	pointer to a Dataset object
in	*configMap*	reference to a ConfigMap (map<string, string>)

Definition at line 39 of file RReliefF.cpp.

6.19.2.4 RReliefF::∼RReliefF ( ) `[virtual]`

Definition at line 49 of file RReliefF.cpp.

6.19.3 Member Function Documentation

6.19.3.1 bool RReliefF::ComputeAttributeScores ( ) `[virtual]`

Compute the ReliefF scores for the current set of attributes.

Implements ReliefF algorithm: Marko Robnik-Sikonja, Igor Kononenko: Theoretical and Empirical Analysis of ReliefF and RReliefF. Machine Learning Journal, 53:23-69, 2003 `http://lkm.fri.uni-lj.si/rmarko/papers/robnik03-mlj.pdf` - Used to hold the probability of a different class val given nearest instances (numeric class)

Used to hold the prob of different value of an attribute given nearest instances (numeric class case)

Used to hold the prob of a different class val and different att val given nearest instances (numeric class case)

algorithm line 1

algorithm line 2

algorithm lines 4, 5 and 6

algorithm line 7

algorithm line 8

algorithm line 9

Reimplemented from ReliefF.

Definition at line 52 of file RReliefF.cpp.

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

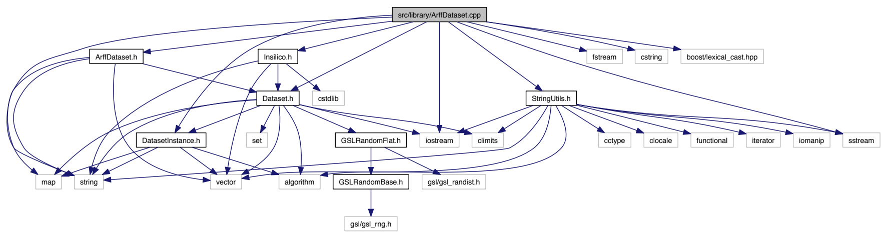
- src/library/RReliefF.h
- src/library/RReliefF.cpp

Chapter 7

File Documentation

7.1 src/library/ArffDataset.cpp File Reference

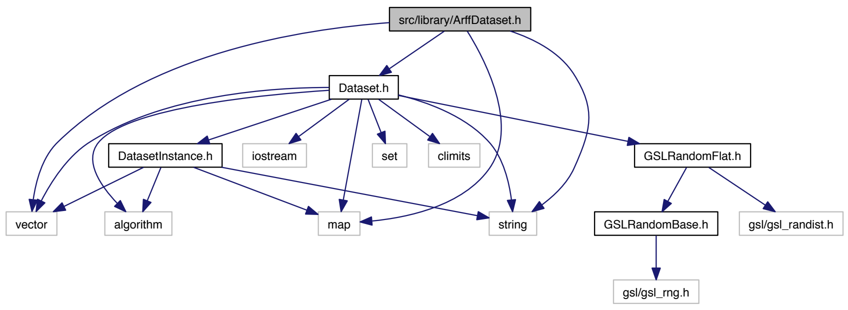
`#include <string>` `#include <iostream>` `#include <fstream>` ×
`#include <cstring>` `#include <sstream>` `#include <boost/lexical-`
`_cast.hpp>` `#include "Dataset.h"` `#include "DatasetInstance.-`
`h"` `#include "StringUtils.h"` `#include "ArffDataset.h"` ×
`#include "Insilico.h"` Include dependency graph for ArffDataset.cpp:



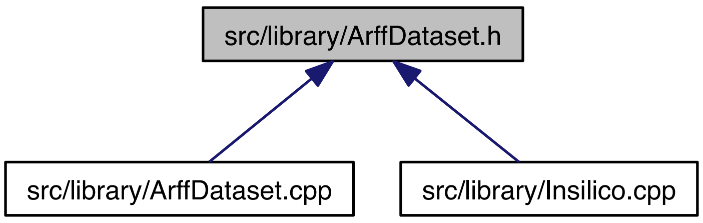
7.2 src/library/ArffDataset.h File Reference

`#include <vector>` `#include <map>` `#include <string>` ×

`#include "Dataset.h"` Include dependency graph for ArffDataset.h:



This graph shows which files directly or indirectly include this file:



Classes

- class ArffDataset

    ARFF file format reader.

Enumerations

- enum ArffAttributeType { ARFF_NUMERIC_TYPE, ARFF_NOMINAL_TYPE, A-RFF_STRING_TYPE, ARFF_DATE_TYPE, ARFF_ERROR_TYPE }

7.2.1 Enumeration Type Documentation

7.2.1.1 enum ArffAttributeType

ARFF attribute types.

Enumerator:

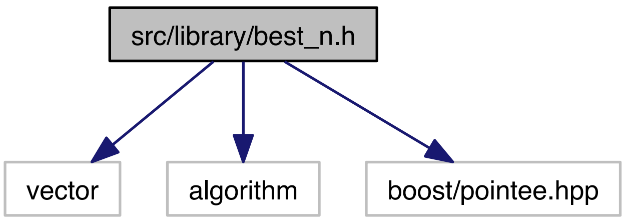
ARFF_NUMERIC_TYPE	continuous levels
ARFF_NOMINAL_TYPE	discrete levels
ARFF_STRING_TYPE	string levels
ARFF_DATE_TYPE	date levels
ARFF_ERROR_TYPE	unknown type

Definition at line 29 of file ArffDataset.h.

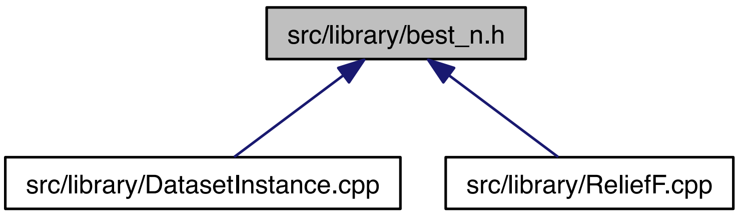
7.3 src/library/best_n.h File Reference

Find the best n keeping original order for ties - stable sort.

`#include <vector>` `#include <algorithm>` `#include <boost/pointee.-hpp>` Include dependency graph for best_n.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace insilico

Functions

- template<typename InputIt , typename OutputIt , typename Comp >
  void insilico::best_n (InputIt begin, InputIt end, OutputIt out, size_t n, Comp comp)

  Get the best n values with ties keeping same original order.

7.3.1 Detailed Description

Find the best n keeping original order for ties - stable sort.

Author

Nate Barney

Version

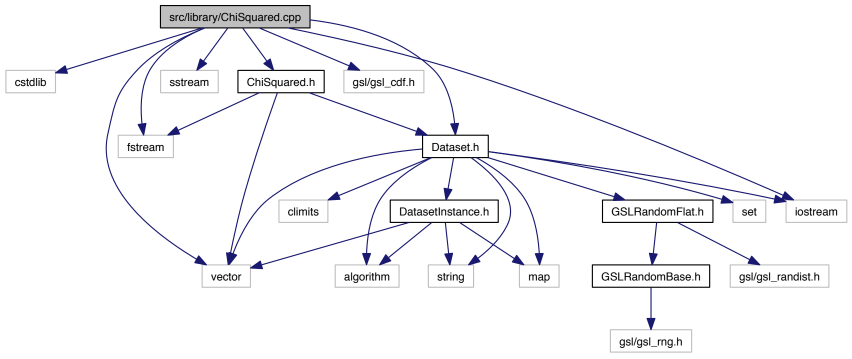
1.0

Contact: `bill.c.white@gmail.com` Created on: 4/7/04

Definition in file best_n.h.

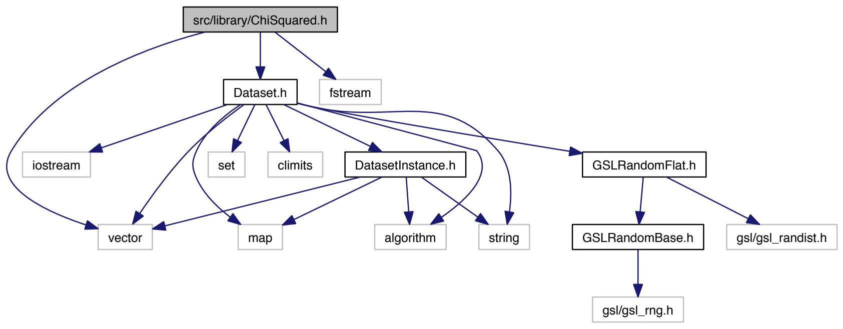
7.4 src/library/ChiSquared.cpp File Reference

`#include <cstdlib>` `#include <iostream>` `#include <fstream>` ×
`#include <sstream>` `#include <vector>` `#include "gsl/gsl_-cdf.h"` `#include "ChiSquared.h"` `#include "Dataset.h"` Include dependency graph for ChiSquared.cpp:

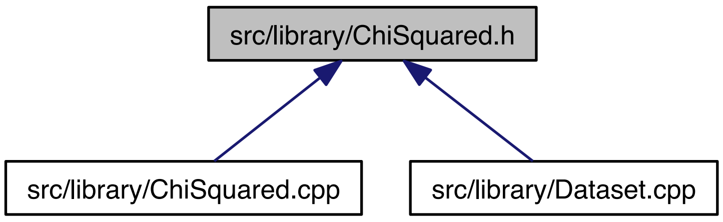


7.5 src/library/ChiSquared.h File Reference

`#include <vector>` `#include <fstream>` `#include "Dataset.-h"` Include dependency graph for ChiSquared.h:



This graph shows which files directly or indirectly include this file:



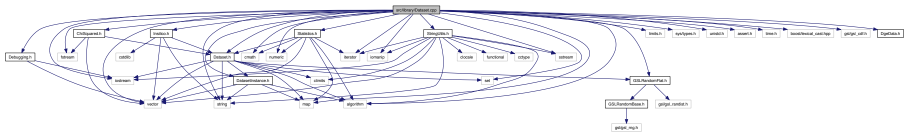
Classes

- class ChiSquared

    Chi-squared attribute ranking algorithm.

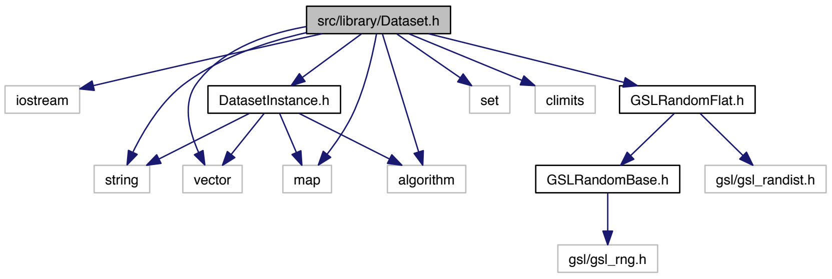
7.6 src/library/Dataset.cpp File Reference

`#include <iostream>` `#include <iomanip>` `#include <fstream>` ×
`#include <string>` `#include <vector>` `#include <set>` ×
`#include <map>` `#include <iterator>` `#include <cmath>` ×
`#include <algorithm>` `#include <numeric>` `#include <sstream>` ×
`#include <limits.h>` `#include <sys/types.h>` `#include <unistd.-`
`h>` `#include <assert.h>` `#include <time.h>` `#include <boost/lexical-`
`_cast.hpp>` `#include "gsl/gsl_cdf.h"` `#include "GSLRandom-`
`Flat.h"` `#include "ChiSquared.h"` `#include "Dataset.h"` ×
`#include "DatasetInstance.h"` `#include "StringUtils.h"` ×
`#include "Statistics.h"` `#include "Debugging.h"` `#include`
`"Insilico.h"` `#include "DgeData.h"` Include dependency graph for -
Dataset.cpp:

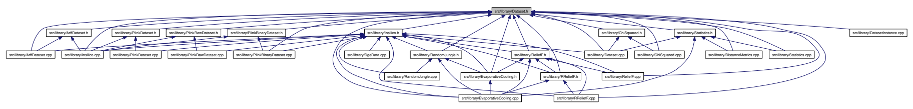


7.7 src/library/Dataset.h File Reference

`#include <iostream>` `#include <string>` `#include <vector>` `#include <map>` `#include <set>` `#include <algorithm>`× `#include <climits>` `#include "DatasetInstance.h"` `#include "GSLRandomFlat.h"` Include dependency graph for Dataset.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Dataset

  Base class for collections of instances containing attributea and class.

Enumerations

- enum ValueType { NUMERIC_VALUE, DISCRETE_VALUE, MISSING_VALUE, NO_VALUE }
- enum AttributeType { NUMERIC_TYPE, DISCRETE_TYPE, NO_TYPE }
- enum ClassType { CONTINUOUS_CLASS_TYPE, CASE_CONTROL_CLASS_-TYPE, MULTI_CLASS_TYPE, NO_CLASS_TYPE }
- enum AttributeMutationType { TRANSITION_MUTATION, TRANSVERSION_M-UTATION, UNKNOWN_MUTATION }

- enum OutputDatasetType { TAB_DELIMITED_DATASET, CSV_DELIMITED_DATASET, ARFF_DATASET, NO_OUTPUT_DATASET }

Variables

- static const int INVALID_DISTANCE = INT_MAX

  return value for invalid distance
- static const int INVALID_INDEX = INT_MAX

  return value for invalid index into attributes
- static const AttributeLevel INVALID_ATTRIBUTE_VALUE = INT_MIN

  invalid attribute value
- static const NumericLevel INVALID_NUMERIC_VALUE = INT_MIN

  invalid attribute value
- static const ClassLevel INVALID_DISCRETE_CLASS_VALUE = INT_MIN

  stored value for missing discrete class
- static const NumericLevel INVALID_NUMERIC_CLASS_VALUE = INT_MIN

  stored value for missing numeric class
- static const AttributeLevel MISSING_ATTRIBUTE_VALUE = -9

  stored value for missing discrete attribute
- static const NumericLevel MISSING_NUMERIC_VALUE = -9

  stored value for missing numeric attribute
- static const ClassLevel MISSING_DISCRETE_CLASS_VALUE = -9

  stored value for missing discrete class
- static const NumericLevel MISSING_NUMERIC_CLASS_VALUE = -9

  stored value for missing numeric class

7.7.1 Enumeration Type Documentation

7.7.1.1 enum AttributeMutationType

Type of attribute mutation.

Enumerator:

TRANSITION_MUTATION	transition within family
TRANSVERSION_MUTATION	transversion between families
UNKNOWN_MUTATION	unknown - no allele information

Definition at line 96 of file Dataset.h.

7.7.1.2 enum AttributeType

Type of attributes that are stored in data set instances.

Enumerator:

NUMERIC_TYPE continuous numeric type
DISCRETE_TYPE discrete genotype type
NO_TYPE default no type

Definition at line 73 of file Dataset.h.

7.7.1.3 enum ClassType

Type of classes that are stored in data set instances.

Enumerator:

CONTINUOUS_CLASS_TYPE continuous numeric type
CASE_CONTROL_CLASS_TYPE discrete case-control type
MULTI_CLASS_TYPE multiclass type
NO_CLASS_TYPE default no type

Definition at line 84 of file Dataset.h.

7.7.1.4 enum OutputDatasetType

Type of data set to write filtered output.

Enumerator:

TAB_DELIMITED_DATASET tab-delimited .txt file
CSV_DELIMITED_DATASET comma separated values .csv file
ARFF_DATASET WEKA ARFF format .arff file.
NO_OUTPUT_DATASET no output data set specified

Definition at line 107 of file Dataset.h.

7.7.1.5 enum ValueType

Return types for determing a value's type.

Enumerator:

NUMERIC_VALUE continuous numeric value
DISCRETE_VALUE discrete genotype value
MISSING_VALUE missing value
NO_VALUE default no value type

Definition at line 61 of file Dataset.h.

7.7.2 Variable Documentation

7.7.2.1 const AttributeLevel INVALID_ATTRIBUTE_VALUE = INT_MIN `[static]`

invalid attribute value

Definition at line 40 of file Dataset.h.

7.7.2.2 const ClassLevel INVALID_DISCRETE_CLASS_VALUE = INT_MIN `[static]`

stored value for missing discrete class

Definition at line 44 of file Dataset.h.

7.7.2.3 const int INVALID_DISTANCE = INT_MAX `[static]`

return value for invalid distance

Definition at line 35 of file Dataset.h.

7.7.2.4 const int INVALID_INDEX = INT_MAX `[static]`

return value for invalid index into attributes

Definition at line 37 of file Dataset.h.

7.7.2.5 const NumericLevel INVALID_NUMERIC_CLASS_VALUE = INT_MIN `[static]`

stored value for missing numeric class

Definition at line 46 of file Dataset.h.

7.7.2.6 const NumericLevel INVALID_NUMERIC_VALUE = INT_MIN `[static]`

invalid attribute value

Definition at line 42 of file Dataset.h.

7.7.2.7 const AttributeLevel MISSING_ATTRIBUTE_VALUE = -9 `[static]`

stored value for missing discrete attribute

Definition at line 49 of file Dataset.h.

7.7.2.8 const ClassLevel MISSING_DISCRETE_CLASS_VALUE = -9 `[static]`

stored value for missing discrete class

Definition at line 53 of file Dataset.h.

7.7.2.9 const NumericLevel MISSING_NUMERIC_CLASS_VALUE = -9 `[static]`

stored value for missing numeric class

Definition at line 55 of file Dataset.h.

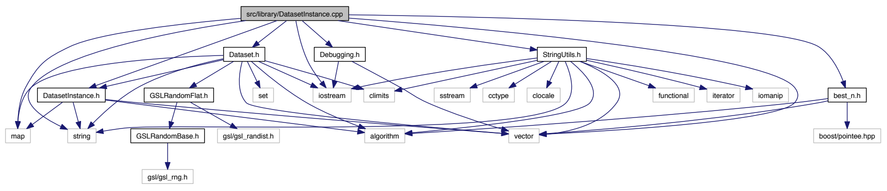
7.7.2.10 const NumericLevel MISSING_NUMERIC_VALUE = -9 `[static]`

stored value for missing numeric attribute

Definition at line 51 of file Dataset.h.

7.8 src/library/DatasetInstance.cpp File Reference

`#include <iostream>` `#include <string>` `#include <vector>` `#include <map>` `#include "Dataset.h"` `#include "DatasetInstance.h"` `#include "StringUtils.h"` `#include "best_n.h"` `#include "Debugging.h"` Include dependency graph for DatasetInstance.cpp:



Classes

- class deref_less_bcw

Typedefs

- typedef DistancePair T

  functor for T comparison

7.8.1 Typedef Documentation

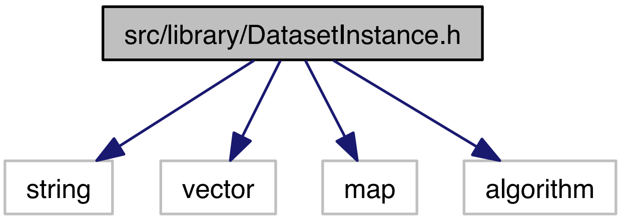
7.8.1.1 typedef DistancePair T

functor for T comparison

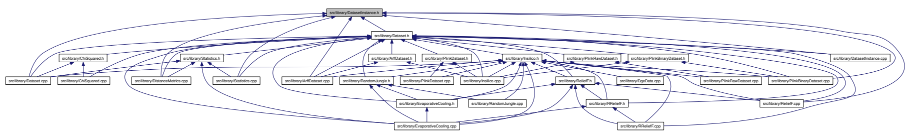
Definition at line 23 of file DatasetInstance.cpp.

7.9 src/library/DatasetInstance.h File Reference

`#include <string>` `#include <vector>` `#include <map>` × `#include <algorithm>` Include dependency graph for DatasetInstance.h:



This graph shows which files directly or indirectly include this file:



Classes

- class DatasetInstance

  Class to hold dataset instances (rows of attributes).

Typedefs

- typedef int AttributeLevel

  type of discrete attribute values

- typedef double NumericLevel

  type of continuous attributes
- typedef int ClassLevel

  type of instance class labels
- typedef std::pair< double, std::string > DistancePair

  distance pair type: distance, instance ID
- typedef std::vector< DistancePair > DistancePairs

  vector of distance pairs represents distances to nearest neighbors
- typedef DistancePairs::const_iterator DistancePairsIt

  distance pairs iterator

7.9.1 Typedef Documentation

7.9.1.1 typedef int **AttributeLevel**

type of discrete attribute values

Definition at line 24 of file DatasetInstance.h.

7.9.1.2 typedef int **ClassLevel**

type of instance class labels

Definition at line 28 of file DatasetInstance.h.

7.9.1.3 typedef std::pair<double, std::string> **DistancePair**

distance pair type: distance, instance ID

Definition at line 31 of file DatasetInstance.h.

7.9.1.4 typedef std::vector<**DistancePair**> **DistancePairs**

vector of distance pairs represents distances to nearest neighbors

Definition at line 33 of file DatasetInstance.h.

7.9.1.5 typedef DistancePairs::const_iterator **DistancePairsIt**

distance pairs iterator

Definition at line 35 of file DatasetInstance.h.

7.9.1.6 typedef double NumericLevel

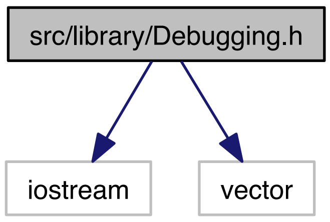
type of continuous attributes

Definition at line 26 of file DatasetInstance.h.

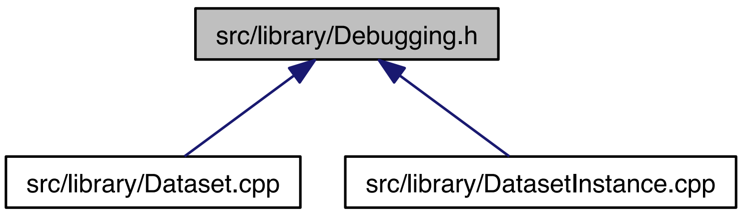
7.10 src/library/Debugging.h File Reference

Debugging utilities.

`#include <iostream>` `#include <vector>` Include dependency graph for Debugging.h:



This graph shows which files directly or indirectly include this file:



Functions

- template<class T >

void PrintVector (std::vector< T > vec, std::string title="")

Print a vector of T values with optional title.

- template<class T >
  void PrintVector (vector< T > vec, string title)

7.10.1 Detailed Description

Debugging utilities.

Author

Bill White

Version

1.0

Contact: `bill.c.white@gmail.com` Created on: 8/q/11

Definition in file Debugging.h.

7.10.2 Function Documentation

7.10.2.1 template<class T > void PrintVector (std::vector< T > *vec,* std::string *title* = "")

Print a vector of T values with optional title.

Parameters

`in`	*vec*	vector of T type values
`in`	*title*	optional title to print before the vector

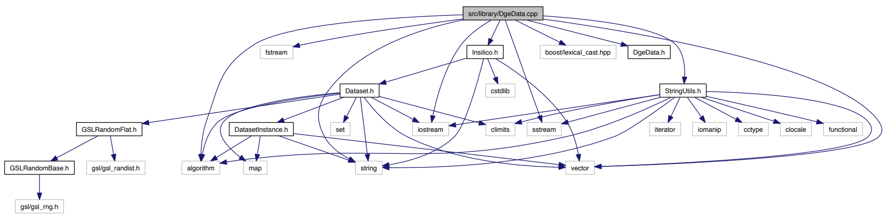
7.10.2.2 template<class T > void PrintVector (vector< T > *vec,* string *title*)

Definition at line 28 of file Debugging.h.

7.11 src/library/DgeData.cpp File Reference

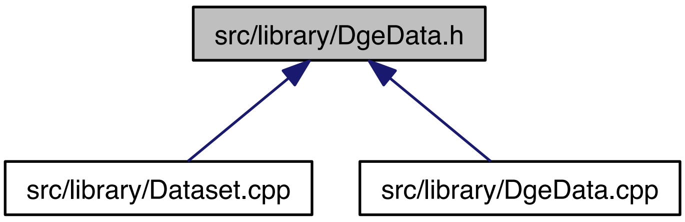
`#include <iostream>` `#include <fstream>` `#include <string>` ×
`#include <sstream>` `#include <vector>` `#include <algorithm>` ×
`#include <boost/lexical_cast.hpp>` `#include "DgeData.h"` ×
`#include "Insilico.h"` `#include "StringUtils.h"` Include depen-

dency graph for DgeData.cpp:



7.12 src/library/DgeData.h File Reference

This graph shows which files directly or indirectly include this file:



Classes

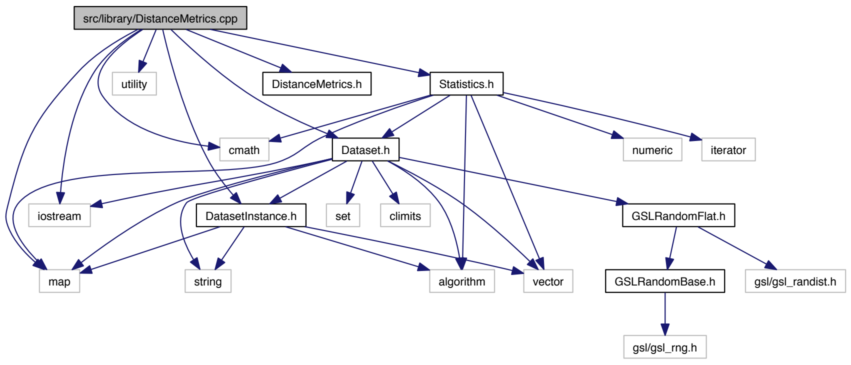
- class DgeData

  Digital gene expression data.

7.13 src/library/DistanceMetrics.cpp File Reference

```
#include <cmath>   #include <iostream>   #include <map> ×
#include <utility> #include "Dataset.h" #include "Distance-
Metrics.h" #include "DatasetInstance.h" #include "Statistics.-
```

h" Include dependency graph for DistanceMetrics.cpp:



Functions

- pair< bool, double > CheckMissing (unsigned int attributeIndex, DatasetInstance ∗dsi1, DatasetInstance ∗dsi2)

  Check for a missing discrete value and return value.
- pair< bool, double > CheckMissingNumeric (unsigned int numericIndex, - DatasetInstance ∗dsi1, DatasetInstance ∗dsi2)

  Check for a missing continuous value and return value.
- double norm (double x, double minX, double maxX)

  Normalizes a given value of a numeric attribute.
- double diffAMM (unsigned int attributeIndex, DatasetInstance ∗dsi1, Dataset-Instance ∗dsi2)

  Allele mismatch metric.
- double diffGMM (unsigned int attributeIndex, DatasetInstance ∗dsi1, Dataset-Instance ∗dsi2)

  Genotype mismatch metric.
- double diffManhattan (unsigned int attributeIndex, DatasetInstance ∗dsi1, - DatasetInstance ∗dsi2)

  "Manhattan" distance between continuous attributes.
- double diffPredictedValueTau (DatasetInstance ∗dsi1, DatasetInstance ∗dsi2)

  Same as "Manhattan" distance but uses method calls versus public variables.

7.13.1 Function Documentation

7.13.1.1 pair<bool, double> CheckMissing (unsigned int *attributeIndex*, DatasetInstance ∗ *dsi1*, DatasetInstance ∗ *dsi2*)

Check for a missing discrete value and return value.

Parameters

in	*attribute-Index*	index into the vector of attributes
in	*dsi1*	data set instance 1
in	*dsi2*	data set instance 2

Returns

pair: has missing value?, value for missing (true) or 0.0 (false)

Definition at line 21 of file DistanceMetrics.cpp.

7.13.1.2 pair<bool, double> **CheckMissingNumeric** (unsigned int *numericIndex*, **DatasetInstance** ∗ *dsi1*, **DatasetInstance** ∗ *dsi2*)

Check for a missing continuous value and return value.

Parameters

in	*attribute-Index*	index into the vector of attributes
in	*dsi1*	data set instance 1
in	*dsi2*	data set instance 2

Returns

pair: has missing value?, value for missing (true) or 0.0 (false)

Definition at line 88 of file DistanceMetrics.cpp.

7.13.1.3 double **diffAMM** (unsigned int *attributeIndex*, **DatasetInstance** ∗ *dsi1*, **DatasetInstance** ∗ *dsi2*)

Allele mismatch metric.

Parameters

in	*attribute-Index*	index into the vector of attributes
in	*dsi1*	data set instance 1
in	*dsi2*	data set instance 2

Returns

diff(erence) between attribute values: 0.0, 0.5, 1.0

Definition at line 135 of file DistanceMetrics.cpp.

7.13.1.4 double **diffGMM** (unsigned int *attributeIndex,* **DatasetInstance** ∗ *dsi1,* **DatasetInstance** ∗ *dsi2*)

Genotype mismatch metric.

Parameters

`in`	*attribute-Index*	index into the vector of attributes
`in`	*dsi1*	data set instance 1
`in`	*dsi2*	data set instance 2

Returns

diff(erence) between attribute values: 0.0 (same) or 1.0 (not same)

Definition at line 150 of file DistanceMetrics.cpp.

7.13.1.5 double **diffManhattan** (unsigned int *attributeIndex,* **DatasetInstance** ∗ *dsi1,* **DatasetInstance** ∗ *dsi2*)

"Manhattan" distance between continuous attributes.

Parameters

`in`	*attribute-Index*	index into the vector of attributes
`in`	*dsi1*	data set instance 1
`in`	*dsi2*	data set instance 2

Returns

absolute value of difference divided by attribute's range

Definition at line 164 of file DistanceMetrics.cpp.

7.13.1.6 double **diffPredictedValueTau** (**DatasetInstance** ∗ *dsi1,* **DatasetInstance** ∗ *dsi2*)

Same as "Manhattan" distance but uses method calls versus public variables.

Parameters

`in`	*attribute-Index*	index into the vector of attributes
`in`	*dsi1*	data set instance 1
`in`	*dsi2*	data set instance 2

Returns

absolute value of difference divided by attribute's range

Definition at line 189 of file DistanceMetrics.cpp.

7.13.1.7 double norm (double *x*, double *minX*, double *maxX*)

Normalizes a given value of a numeric attribute.

Borrowed from Weka 8/18/11

Parameters

in	*x*	value
in	*minX*	minimum value for x
in	*maxX*	maximum value for x

Returns

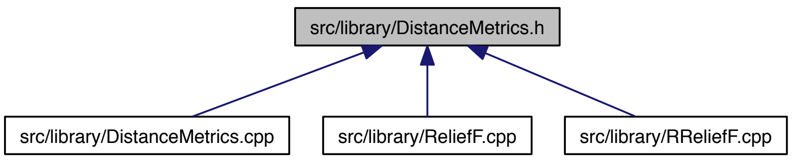
normalized value

Definition at line 127 of file DistanceMetrics.cpp.

7.14 src/library/DistanceMetrics.h File Reference

Distance metrics for ReliefF.

This graph shows which files directly or indirectly include this file:


src/library/DistanceMetrics.h
src/library/DistanceMetrics.cpp
src/library/ReliefF.cpp
src/library/RReliefF.cpp

Functions

- std::pair< bool, double > CheckMissing (unsigned int attributeIndex, DatasetInstance *dsi1, DatasetInstance *dsi2)

  Check for a missing discrete value and return value.
- std::pair< bool, double > CheckMissingNumeric (unsigned int numericIndex, DatasetInstance *dsi1, DatasetInstance *dsi2)

Check for a missing continuous value and return value.

- double norm (double x, double minX, double maxX)

  Normalizes a given value of a numeric attribute.
- double diffAMM (unsigned int attributeIndex, DatasetInstance ∗dsi1, DatasetInstance ∗dsi2)

  Allele mismatch metric.
- double diffGMM (unsigned int attributeIndex, DatasetInstance ∗dsi1, DatasetInstance ∗dsi2)

  Genotype mismatch metric.
- double diffManhattan (unsigned int attributeIndex, DatasetInstance ∗dsi1, -DatasetInstance ∗dsi2)

  "Manhattan" distance between continuous attributes.
- double diffPredictedValueTau (DatasetInstance ∗dsi1, DatasetInstance ∗dsi2)

  Same as "Manhattan" distance but uses method calls versus public variables.

7.14.1 Detailed Description

Distance metrics for ReliefF.

Author

: Bill White

Version

1.0

Contact: `bill.c.white@gmail.com` Created on 3/29/11

Definition in file DistanceMetrics.h.

7.14.2 Function Documentation

7.14.2.1 std::pair<bool, double> CheckMissing (unsigned int *attributeIndex*, DatasetInstance ∗ *dsi1*, DatasetInstance ∗ *dsi2*)

Check for a missing discrete value and return value.

Parameters

`in`	*attribute-Index*	index into the vector of attributes
`in`	*dsi1*	data set instance 1
`in`	*dsi2*	data set instance 2

Returns

pair: has missing value?, value for missing (true) or 0.0 (false)

Definition at line 21 of file DistanceMetrics.cpp.

7.14.2.2 std::pair<bool, double> **CheckMissingNumeric** (unsigned int *numericIndex*, **DatasetInstance** ∗ *dsi1*, **DatasetInstance** ∗ *dsi2*)

Check for a missing continuous value and return value.

Parameters

`in`	*attribute-Index*	index into the vector of attributes
`in`	*dsi1*	data set instance 1
`in`	*dsi2*	data set instance 2

Returns

pair: has missing value?, value for missing (true) or 0.0 (false)

Definition at line 88 of file DistanceMetrics.cpp.

7.14.2.3 double **diffAMM** (unsigned int *attributeIndex*, **DatasetInstance** ∗ *dsi1*, **DatasetInstance** ∗ *dsi2*)

Allele mismatch metric.

Parameters

`in`	*attribute-Index*	index into the vector of attributes
`in`	*dsi1*	data set instance 1
`in`	*dsi2*	data set instance 2

Returns

diff(erence) between attribute values: 0.0, 0.5, 1.0

Definition at line 135 of file DistanceMetrics.cpp.

7.14.2.4 double **diffGMM** (unsigned int *attributeIndex*, **DatasetInstance** ∗ *dsi1*, **DatasetInstance** ∗ *dsi2*)

Genotype mismatch metric.

Parameters

`in`	*attribute-Index*	index into the vector of attributes
`in`	*dsi1*	data set instance 1
`in`	*dsi2*	data set instance 2

Returns

diff(erence) between attribute values: 0.0 (same) or 1.0 (not same)

Definition at line 150 of file DistanceMetrics.cpp.

7.14.2.5 double **diffManhattan** (unsigned int *attributeIndex,* **DatasetInstance** ∗ *dsi1,* **DatasetInstance** ∗ *dsi2*)

"Manhattan" distance between continuous attributes.

Parameters

`in`	*attribute-Index*	index into the vector of attributes
`in`	*dsi1*	data set instance 1
`in`	*dsi2*	data set instance 2

Returns

absolute value of difference divided by attribute's range

Definition at line 164 of file DistanceMetrics.cpp.

7.14.2.6 double **diffPredictedValueTau** (**DatasetInstance** ∗ *dsi1,* **DatasetInstance** ∗ *dsi2*)

Same as "Manhattan" distance but uses method calls versus public variables.

Parameters

`in`	*attribute-Index*	index into the vector of attributes
`in`	*dsi1*	data set instance 1
`in`	*dsi2*	data set instance 2

Returns

absolute value of difference divided by attribute's range

Definition at line 189 of file DistanceMetrics.cpp.

7.14.2.7 double **norm** (double *x*, double *minX*, double *maxX*)

Normalizes a given value of a numeric attribute.

Borrowed from Weka 8/18/11

Parameters

in	*x*	value
in	*minX*	minimum value for x
in	*maxX*	maximum value for x

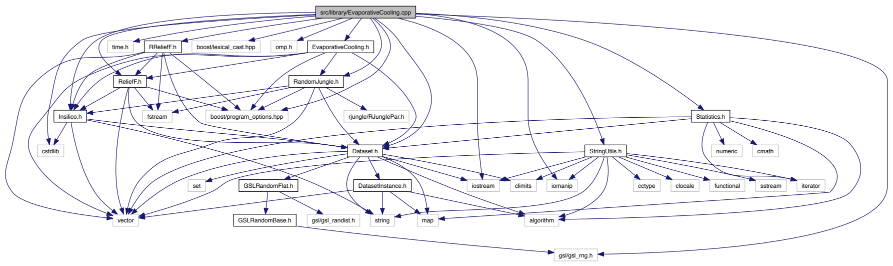
Returns

normalized value

Definition at line 127 of file DistanceMetrics.cpp.

7.15 src/library/EvaporativeCooling.cpp File Reference

`#include <cstdlib>` `#include <iostream>` `#include <iomanip>` × `#include <time.h>` `#include <boost/program_options.hpp>` `#include <boost/lexical_cast.hpp>` `#include <omp.h>` × `#include <gsl/gsl_rng.h>` `#include "EvaporativeCooling.h"` `#include "Dataset.h"` `#include "Statistics.h"` `#include "-StringUtils.h"` `#include "RandomJungle.h"` `#include "Relief-F.h"` `#include "RReliefF.h"` `#include "Insilico.h"` Include dependency graph for EvaporativeCooling.cpp:



Functions

- bool scoresSortAsc (const pair< double, string > &p1, const pair< double, string > &p2)
- bool scoresSortAscByName (const pair< double, string > &p1, const pair< double, string > &p2)
- bool scoresSortDesc (const pair< double, string > &p1, const pair< double, string > &p2)

7.15.1 Function Documentation

7.15.1.1 bool **scoresSortAsc** (const pair< double, string > & *p1*, const pair< double, string > & *p2*)

Definition at line 39 of file EvaporativeCooling.cpp.

7.15.1.2 bool **scoresSortAscByName** (const pair< double, string > & *p1*, const pair< double, string > & *p2*)

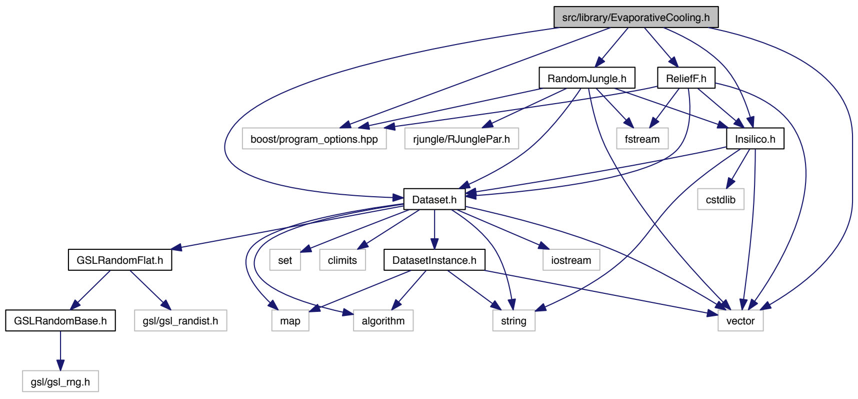
Definition at line 44 of file EvaporativeCooling.cpp.

7.15.1.3 bool **scoresSortDesc** (const pair< double, string > & *p1*, const pair< double, string > & *p2*)

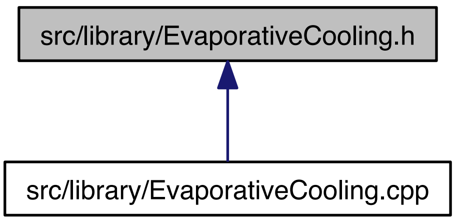
Definition at line 49 of file EvaporativeCooling.cpp.

7.16 src/library/EvaporativeCooling.h File Reference

`#include <vector>` `#include <boost/program_options.hpp>` × `#include "Dataset.h"` `#include "RandomJungle.h"` `#include "ReliefF.h"` `#include "Insilico.h"` Include dependency graph for - EvaporativeCooling.h:



This graph shows which files directly or indirectly include this file:


src/library/EvaporativeCooling.h
src/library/EvaporativeCooling.cpp

Classes

- class EvaporativeCooling

  Evaporative Cooling attribute ranking algorithm.

Typedefs

- typedef std::vector< std::pair < double, std::string > > EcScores

  evaporative cooling scores - sorted by score key
- typedef std::vector< std::pair < double, std::string > >::iterator EcScoresIt

  evaporative cooling scores iterator - sorted by score key
- typedef std::vector< std::pair < double, std::string > >::const_iterator EcScores-CIt

  evaporative cooling scores constant iterator - sorted by score key

Enumerations

- enum EcAlgorithmType { EC_ALL, EC_RJ, EC_RF }

Functions

- void libec_is_present (void)

  HACK FOR AUTOTOOLS LIBRARY DETECTION.

7.16.1 Typedef Documentation

7.16.1.1 typedef std::vector<std::pair<double, std::string> > EcScores

evaporative cooling scores - sorted by score key

Definition at line 36 of file EvaporativeCooling.h.

7.16.1.2 typedef std::vector<std::pair<double, std::string> >::const_iterator EcScoresCIt

evaporative cooling scores constant iterator - sorted by score key

Definition at line 40 of file EvaporativeCooling.h.

7.16.1.3 typedef std::vector<std::pair<double, std::string> >::iterator EcScoresIt

evaporative cooling scores iterator - sorted by score key

Definition at line 38 of file EvaporativeCooling.h.

7.16.2 Enumeration Type Documentation

7.16.2.1 enum EcAlgorithmType

Type of algorithm steps to perform.

Enumerator:

EC_ALL Run RandomJungle and ReliefF.

EC_RJ Run only RandomJungle.

EC_RF Run only ReliefF.

Definition at line 46 of file EvaporativeCooling.h.

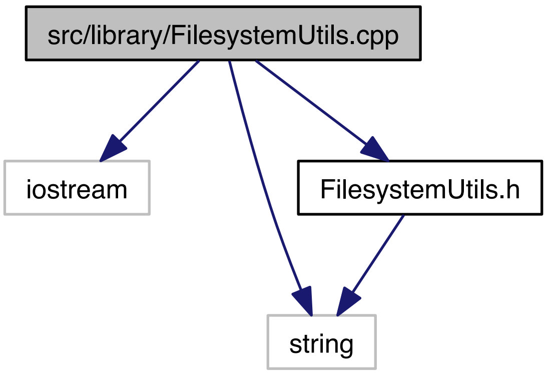
7.16.3 Function Documentation

7.16.3.1 void libec_is_present (void)

HACK FOR AUTOTOOLS LIBRARY DETECTION.

7.17 src/library/FilesystemUtils.cpp File Reference

`#include <iostream>` `#include <string>` `#include "FilesystemUtils.h"` Include dependency graph for FilesystemUtils.cpp:



Functions

- string GetFileBasename (string fileName)
- string GetFileExtension (string fileName)

7.17.1 Function Documentation

7.17.1.1 string GetFileBasename (string *fileName*)

Definition at line 8 of file FilesystemUtils.cpp.

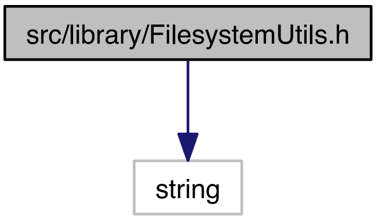
7.17.1.2 string GetFileExtension (string *fileName*)

Definition at line 13 of file FilesystemUtils.cpp.

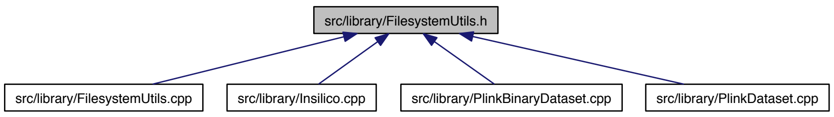
7.18 src/library/FilesystemUtils.h File Reference

Filesystem utilities.

`#include <string>` Include dependency graph for FilesystemUtils.h:



This graph shows which files directly or indirectly include this file:


src/library/FilesystemUtils.h
src/library/FilesystemUtils.cpp
src/library/Insilico.cpp
src/library/PlinkBinaryDataset.cpp
src/library/PlinkDataset.cpp

Functions

- std::string GetFileBasename (std::string fullFilename)

  Get the full filename without the extension.
- std::string GetFileExtension (std::string fullFilename)

  Get the filename extension.

7.18.1 Detailed Description

Filesystem utilities.

Author

Bill White

Version

1.0

Contact: bill.c.white@gmail.com Created on: 4/7/11

Definition in file FilesystemUtils.h.

7.18.2 Function Documentation

7.18.2.1 std::string GetFileBasename (std::string *fullFilename*)

Get the full filename without the extension.

Parameters

in	*fullFilename*	complete filename

Returns

path/filename without extension

7.18.2.2 std::string GetFileExtension (std::string *fullFilename*)

Get the filename extension.

Parameters

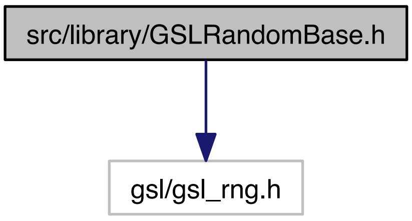
in	*fullFilename*	complete filename

Returns

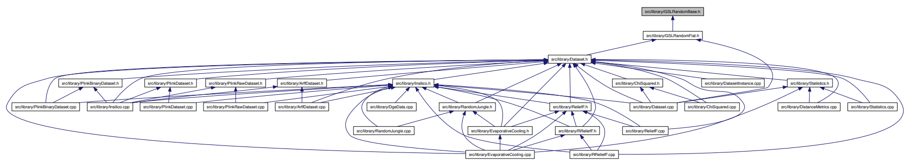
filename extension

7.19 src/library/GSLRandomBase.h File Reference

`#include "gsl/gsl_rng.h"` Include dependency graph for GSLRandomBase.-h:



This graph shows which files directly or indirectly include this file:



Classes

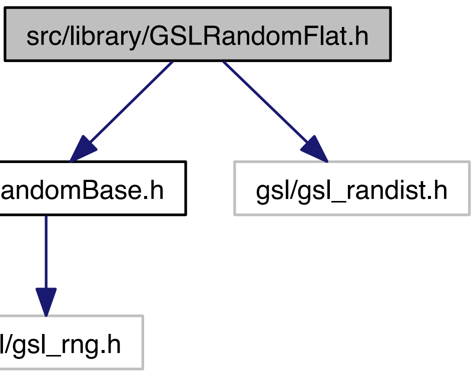
- class GSLRandomBase

  A base class for GNU Scientific Library (GSL) random number functions.

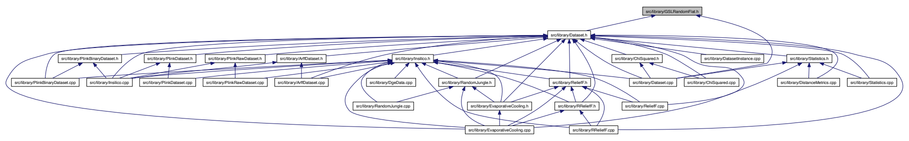
7.20 src/library/GSLRandomFlat.h File Reference

`#include "GSLRandomBase.h"  #include "gsl/gsl_randist.h"` ×

Include dependency graph for GSLRandomFlat.h:



This graph shows which files directly or indirectly include this file:



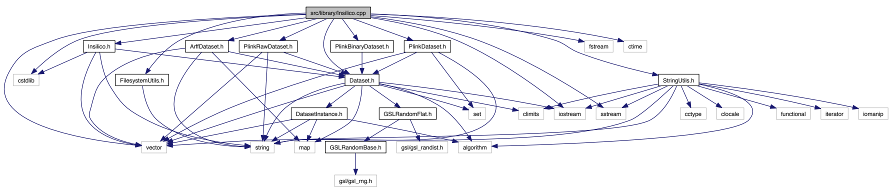
Classes

- class GSLRandomFlat

  Random numbers in a flat, or uniform distribution.

7.21 src/library/Insilico.cpp File Reference

`#include <cstdlib>` `#include <iostream>` `#include <fstream>` × `#include <vector>` `#include <sstream>` `#include <ctime>` × `#include "Dataset.h"` `#include "ArffDataset.h"` `#include "-PlinkDataset.h"` `#include "PlinkRawDataset.h"` `#include "-PlinkBinaryDataset.h"` `#include "StringUtils.h"` `#include "-FilesystemUtils.h"` `#include "Insilico.h"` Include dependency graph for Insilico.cpp:



Functions

- string Timestamp ()

  Return a timestamp string for logging purposes.
- Dataset ∗ ChooseSnpsDatasetByExtension (string snpsFilename)
- bool LoadNumericIds (string filename, vector< string > &retIds)
- bool LoadPhenoIds (string filename, vector< string > &retIds)
- bool GetMatchingIds (string numericsFilename, string altPhenotypeFilename, vector< string > numericsIds, vector< string > phenoIds, vector< string > &matchingIds)
- ClassType DetectClassType (std::string filename, int classColumn, bool hasHeader)

  Detect the class type by reading the specified column from a whitespace- delimited text file.
- bool GetConfigValue (ConfigMap &configMap, std::string key, std::string &value)

  Get the parameter value from the configuration map key.

7.21.1 Function Documentation

7.21.1.1 Dataset∗ ChooseSnpsDatasetByExtension (string *snpsFilename*)

Definition at line 42 of file Insilico.cpp.

7.21.1.2 ClassType DetectClassType (std::string *filename,* int *classColumn,* bool *hasHeader*)

Detect the class type by reading the specified column from a whitespace- delimited text file.

Parameters

in	*filename*	whitespace-delimited text file name
in	*classColumn*	the column containing the class values
in	*heasHeader*	does the file have a header line?

Returns

ClassType defined in Dataset.h

Open the file for reading

Skip the header if it has one

Determine the phenotype type

Definition at line 237 of file Insilico.cpp.

7.21.1.3 bool GetConfigValue (ConfigMap & *configMap,* std::string *key,* std::string & *value*)

Get the parameter value from the configuration map key.

Parameters

`in`	*configMap*	reference to a configuration map
`in`	*key*	parameter name
`out`	*parameter*	value

Returns

true if key found, false if not found

Definition at line 304 of file Insilico.cpp.

7.21.1.4 bool GetMatchingIds (string *numericsFilename,* string *altPhenotypeFilename,* vector< string > *numericsIds,* vector< string > *phenoIds,* vector< string > & *matchingIds*)

Definition at line 183 of file Insilico.cpp.

7.21.1.5 bool LoadNumericIds (string *filename,* vector< string > & *retIds*)

Definition at line 82 of file Insilico.cpp.

7.21.1.6 bool LoadPhenoIds (string *filename,* vector< string > & *retIds*)

Definition at line 134 of file Insilico.cpp.

7.21.1.7 string Timestamp ( )

Return a timestamp string for logging purposes.

Returns

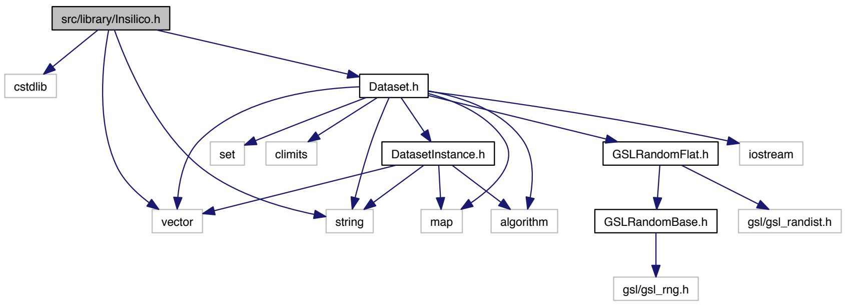
fixed-length, formatted timestamp as a string

Definition at line 30 of file Insilico.cpp.

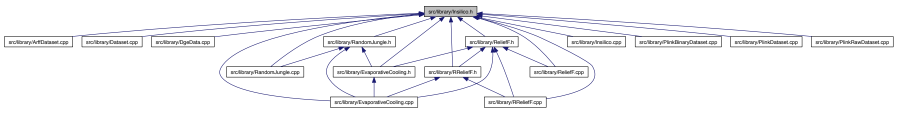
7.22 src/library/Insilico.h File Reference

Common functions for Insilico Lab projects.

`#include <cstdlib>` `#include <string>` `#include <vector>` ×
`#include "Dataset.h"` Include dependency graph for Insilico.h:



This graph shows which files directly or indirectly include this file:


src/library/Insilico.h
src/library/ArffDataset.cpp
src/library/Dataset.cpp
src/library/DgeData.cpp
src/library/RandomJungle.h
src/library/ReliefF.h
src/library/Insilico.cpp
src/library/PlinkBinaryDataset.cpp
src/library/PlinkDataset.cpp
src/library/PlinkRawDataset.cpp
src/library/RandomJungle.cpp
src/library/EvaporativeCooling.h
src/library/RReliefF.h
src/library/ReliefF.cpp
src/library/EvaporativeCooling.cpp
src/library/RReliefF.cpp

Typedefs

- typedef std::map< std::string, std::string > ConfigMap

  Forward reference to Dataset class.

Enumerations

- enum AnalysisType { SNP_ONLY_ANALYSIS, SNP_CLEAN_ANALYSIS, NUMERIC_ONLY_ANALYSIS, INTEGRATED_ANALYSIS, DIAGNOSTIC_ANALYSIS, REGRESSION_ANALYSIS, DGE_ANALYSIS, NO_ANALYSIS }

Functions

- std::string Timestamp ()

  Return a timestamp string for logging purposes.
- Dataset ∗ ChooseSnpsDatasetByExtension (std::string snpsFilename)

  Determines the data set type to instantiate based on the data set filenames's extension.
- bool LoadNumericIds (std::string filename, std::vector< std::string > &retIds)

  Loads the individual (instance) IDs from the numerics file.
- bool LoadPhenoIds (std::string filename, std::vector< std::string > &retIds)

  Loads the individual (instance) IDs from the numerics file.
- bool GetMatchingIds (std::string numericsFilename, std::string altPhenotypeFilename, std::vector< std::string > numericsIds, std::vector< std::string > phenoIds, std::vector< std::string > &matchingIds)

  Return matching IDs from numeric and/or phenotype file IDs.
- ClassType DetectClassType (std::string filename, int classColumn, bool hasHeader)

  Detect the class type by reading the specified column from a whitespace- delimited text file.
- bool GetConfigValue (ConfigMap &configMap, std::string key, std::string &value)

  Get the parameter value from the configuration map key.

Variables

- static const int COMMAND_LINE_ERROR = EXIT_FAILURE

  Error codes.

7.22.1 Detailed Description

Common functions for Insilico Lab projects.

Author

: Bill White

Version

1.0

Contact: `bill.c.white@gmail.com` Created on 10/13/11

Definition in file Insilico.h.

7.22.2 Typedef Documentation

7.22.2.1 typedef std::map<std::string, std::string> ConfigMap

Forward reference to Dataset class.

Definition at line 23 of file Insilico.h.

7.22.3 Enumeration Type Documentation

7.22.3.1 enum AnalysisType

Type of analysis to perform.

Enumerator:

SNP_ONLY_ANALYSIS	discrete analysis
SNP_CLEAN_ANALYSIS	discrete analysis - no filtering
NUMERIC_ONLY_ANALYSIS	continuous attributes
INTEGRATED_ANALYSIS	discrete and continuous analysis
DIAGNOSTIC_ANALYSIS	diagnostic mode - no ReliefF analysis
REGRESSION_ANALYSIS	regression ReliefF analysis
DGE_ANALYSIS	digital gene expression (DGE) analysis
NO_ANALYSIS	no analysis specified

Definition at line 29 of file Insilico.h.

7.22.4 Function Documentation

7.22.4.1 Dataset* ChooseSnpsDatasetByExtension (std::string *snpsFilename*)

Determines the data set type to instantiate based on the data set filenames's extension.

Parameters

`in`	*snps-Filename*	SNP data set filename

Returns

pointer to new dataset or NULL if could not match filename extension

7.22.4.2 ClassType DetectClassType (std::string *filename,* int *classColumn,* bool *hasHeader*)

Detect the class type by reading the specified column from a whitespace- delimited text file.

Parameters

in	*filename*	whitespace-delimited text file name
in	*classColumn*	the column containing the class values
in	*heasHeader*	does the file have a header line?

Returns

ClassType defined in Dataset.h

Open the file for reading

Skip the header if it has one

Determine the phenotype type

Definition at line 237 of file Insilico.cpp.

7.22.4.3 bool GetConfigValue (ConfigMap & *configMap,* std::string *key,* std::string & *value*)

Get the parameter value from the configuration map key.

Parameters

in	*configMap*	reference to a configuration map
in	*key*	parameter name
out	*parameter*	value

Returns

true if key found, false if not found

Definition at line 304 of file Insilico.cpp.

7.22.4.4 bool GetMatchingIds (std::string *numericsFilename,* std::string *altPhenotypeFilename,* std::vector< std::string > *numericsIds,* std::vector< std::string > *phenoIds,* std::vector< std::string > & *matchingIds*)

Return matching IDs from numeric and/or phenotype file IDs.

Parameters

`in`	*numerics-Filename*	name of the PLINK covar format file
`in`	*alt-Phenotype-Filename*	name of the alternate pheno file PLINK
`in`	*numericsIds*	covar format file ids
`in`	*phenoIds*	alternate phenotype file ids
`out`	*matchingIds*	ids that match between numerics and phenotypes

Returns

success

7.22.4.5 bool **LoadNumericIds** (std::string *filename,* std::vector< std::string > & *retIds*)

Loads the individual (instance) IDs from the numerics file.

Returns the IDs through reference parameter retIds.

Parameters

`in`	*filename*	filename that contains numerics IDs
`out`	*vector*	of individual (instance) IDs (strings)

Returns

success

7.22.4.6 bool **LoadPhenoIds** (std::string *filename,* std::vector< std::string > & *retIds*)

Loads the individual (instance) IDs from the numerics file.

Returns the IDs through reference parameter retIds.

Parameters

`in`	*filename*	filename that contains numerics IDs
`out`	*vector*	of individual (instance) IDs (strings)

Returns

success

7.22.4.7 std::string **Timestamp** ( )

Return a timestamp string for logging purposes.

Returns

fixed-length, formatted timestamp as a string

Definition at line 30 of file Insilico.cpp.

7.22.5 Variable Documentation

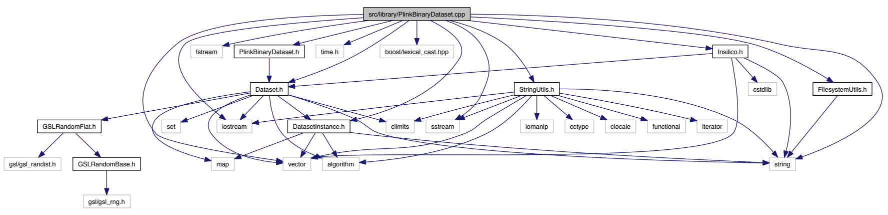
7.22.5.1 const int **COMMAND_LINE_ERROR** = EXIT_FAILURE `[static]`

Error codes.

Definition at line 42 of file Insilico.h.

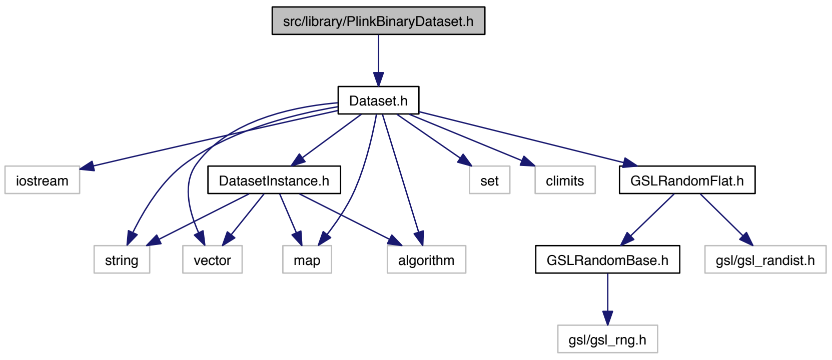
7.23 src/library/PlinkBinaryDataset.cpp File Reference

`#include <string>` `#include <iostream>` `#include <fstream>`× `#include <vector>` `#include <time.h>` `#include <sstream>` `#include <boost/lexical_cast.hpp>` `#include "Dataset.h"` `#include "DatasetInstance.h"` `#include "StringUtils.h"` `#include "FilesystemUtils.h"` `#include "PlinkBinaryDataset.h"` `#include "Insilico.h"` Include dependency graph for PlinkBinaryDataset.cpp:

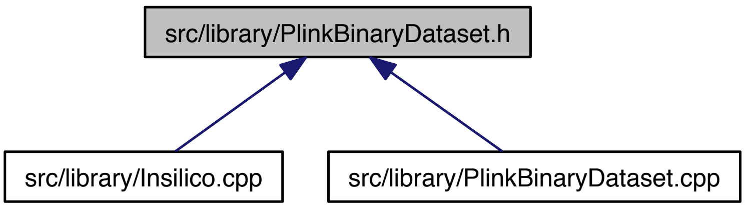


7.24 src/library/PlinkBinaryDataset.h File Reference

`#include "Dataset.h"` Include dependency graph for PlinkBinaryDataset.h:



This graph shows which files directly or indirectly include this file:


src/library/PlinkBinaryDataset.h
src/library/Insilico.cpp
src/library/PlinkBinaryDataset.cpp

Classes

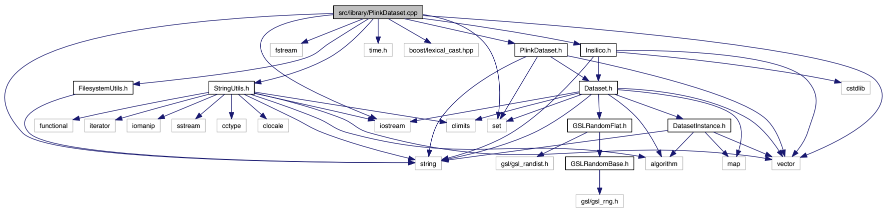
- class PlinkBinaryDataset

  Plink binary PED/BED file format reader.

7.25 src/library/PlinkDataset.cpp File Reference

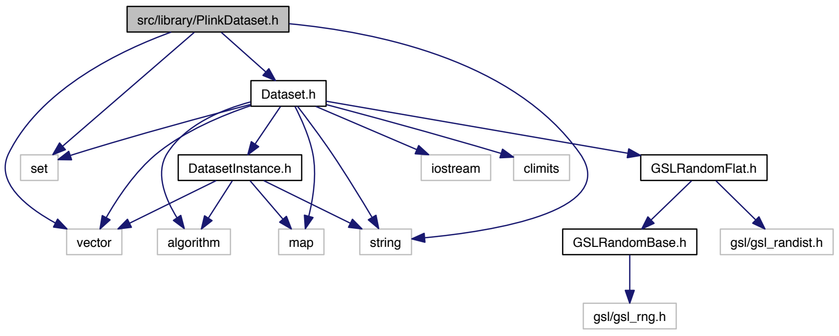
`#include <string>` `#include <iostream>` `#include <fstream>` ×
`#include <vector>` `#include <set>` `#include <time.h>` ×

`#include <boost/lexical_cast.hpp>` `#include "StringUtils.-h"` `#include "FilesystemUtils.h"` `#include "PlinkDataset.h"` `#include "Insilico.h"` Include dependency graph for PlinkDataset.cpp:

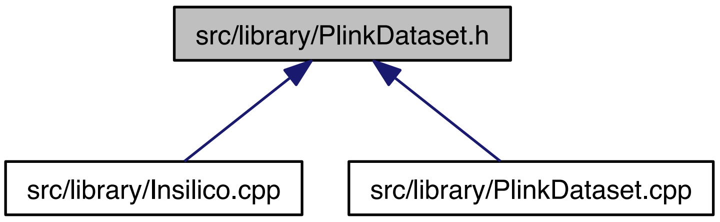


7.26 src/library/PlinkDataset.h File Reference

`#include <set>` `#include <vector>` `#include <string>` × `#include "Dataset.h"` Include dependency graph for PlinkDataset.h:



This graph shows which files directly or indirectly include this file:



Classes

- class PlinkDataset

    Plink MAP/PED file format reader.

Enumerations

- enum MapFileType { MAP3_FILE, MAP4_FILE, ERROR_FILE }

7.26.1 Enumeration Type Documentation

7.26.1.1 enum MapFileType

PLINK map file types.

Enumerator:

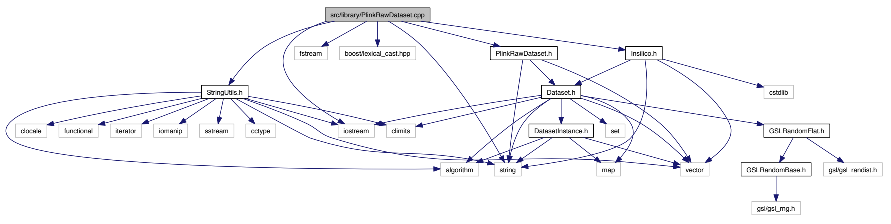
MAP3_FILE	map 3 simplified format
MAP4_FILE	map 4 standard format
ERROR_FILE	default

Definition at line 28 of file PlinkDataset.h.

7.27 src/library/PlinkRawDataset.cpp File Reference

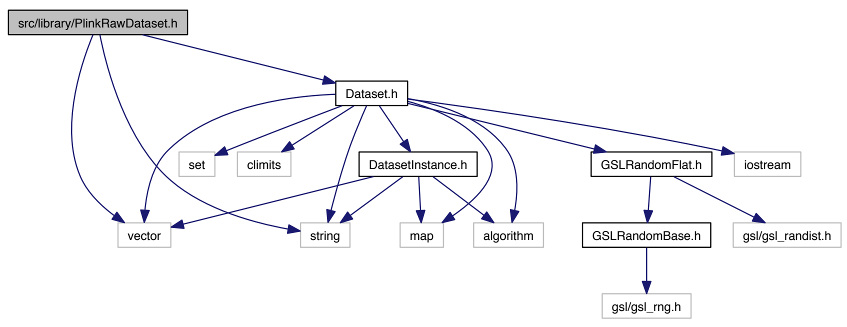
```
#include <string> #include <iostream> #include <fstream> ×
#include <boost/lexical_cast.hpp> #include "StringUtils.-
```

`h" #include "PlinkRawDataset.h" #include "Insilico.h"` Include dependency graph for PlinkRawDataset.cpp:

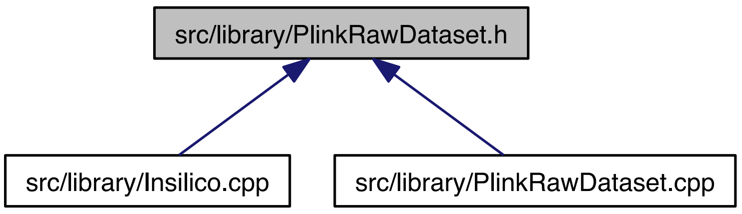


7.28 src/library/PlinkRawDataset.h File Reference

`#include <string> #include <vector> #include "Dataset.h"` ×
Include dependency graph for PlinkRawDataset.h:



This graph shows which files directly or indirectly include this file:


src/library/PlinkRawDataset.h
src/library/Insilico.cpp
src/library/PlinkRawDataset.cpp

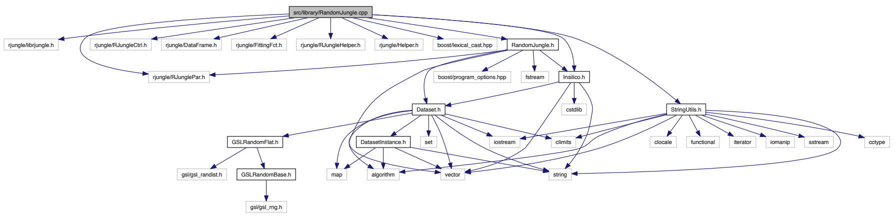
Classes

- class PlinkRawDataset

  Plink recodeA/RAW file format reader.

7.29 src/library/RandomJungle.cpp File Reference

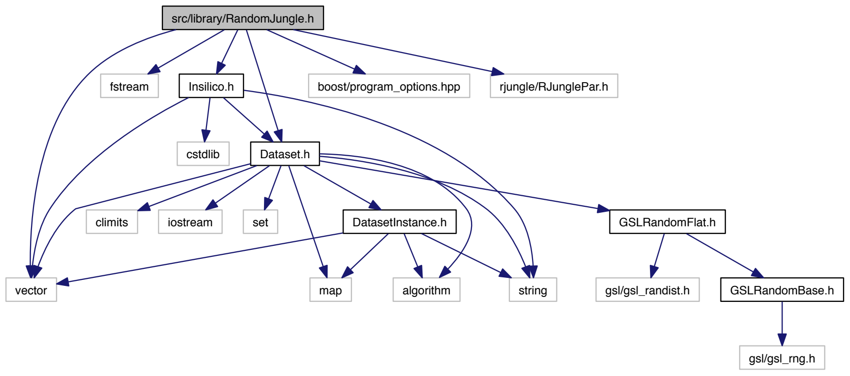
`#include "rjungle/librjungle.h"` `#include "rjungle/RJungle-Par.h"` `#include "rjungle/RJungleCtrl.h"` `#include "rjungle/-DataFrame.h"` `#include "rjungle/FittingFct.h"` `#include "rjungle/RJungleHelper.h"` `#include "rjungle/Helper.h"` × `#include "boost/lexical_cast.hpp"` `#include "RandomJungle.-h"` `#include "StringUtils.h"` `#include "Insilico.h"` Include dependency graph for RandomJungle.cpp:



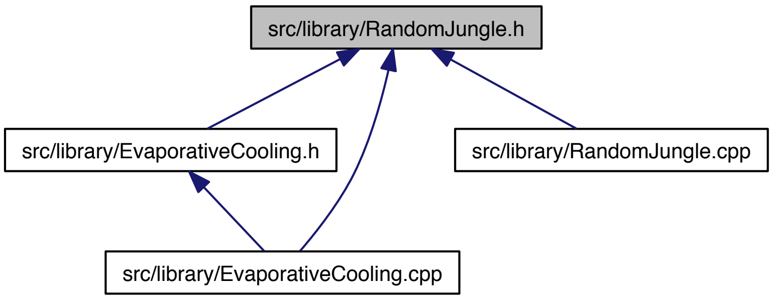
7.30 src/library/RandomJungle.h File Reference

`#include <vector>` `#include <fstream>` `#include "Insilico.-h"` `#include "Dataset.h"` `#include <boost/program_options.-`

`hpp>` `#include "rjungle/RJunglePar.h"` Include dependency graph for RandomJungle.h:



This graph shows which files directly or indirectly include this file:


src/library/RandomJungle.h
src/library/EvaporativeCooling.h
src/library/RandomJungle.cpp
src/library/EvaporativeCooling.cpp

Classes

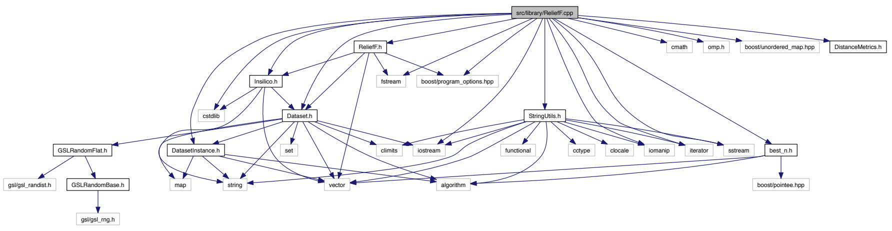
- class RandomJungle

  RandomJungle attribute ranking algorithm.

7.31 src/library/ReliefF.cpp File Reference

`#include <cstdlib>` `#include <iostream>` `#include <fstream>` ×
`#include <iomanip>` `#include <iterator>` `#include <cmath>`

`#include <sstream>` `#include <omp.h>` `#include <boost/program-_options.hpp>` `#include <boost/unordered_map.hpp>` `#include "ReliefF.h"` `#include "Dataset.h"` `#include "DatasetInstance.-h"` `#include "StringUtils.h"` `#include "DistanceMetrics.h"` ×
`#include "best_n.h"` `#include "Insilico.h"` Include dependency graph for ReliefF.cpp:



Classes

- class deref_less

Typedefs

- typedef vector< pair< double, unsigned int > > ScoresMap

    scores map: score->attribute index
- typedef vector< pair< double, unsigned int > >::iterator ScoresMapIt

    scores map iterator
- typedef vector< pair< unsigned int, double > > AttributeIndex

    attribute index map: attribute index->score
- typedef vector< pair< unsigned int, double > >::const_iterator AttributeIndex-It

    attribute index map iterator
- typedef pair< unsigned int, DatasetInstance ∗ > T

    functor for T comparison

Functions

- bool scoreSort (const pair< double, string > &p1, const pair< double, string > &p2)

    attribute score sorting functor
- bool attributeSort (const pair< unsigned int, double > &p1, const pair< unsigned int, double > &p2)

    attribute index sorting functor
- void librelieff_is_present (void)

7.31.1 Typedef Documentation

7.31.1.1 typedef vector<pair<unsigned int, double> > **AttributeIndex**

attribute index map: attribute index->score

Definition at line 41 of file ReliefF.cpp.

7.31.1.2 typedef vector<pair<unsigned int, double> >::const_iterator **AttributeIndexIt**

attribute index map iterator

Definition at line 43 of file ReliefF.cpp.

7.31.1.3 typedef vector<pair<double, unsigned int> > **ScoresMap**

scores map: score->attribute index

Definition at line 37 of file ReliefF.cpp.

7.31.1.4 typedef vector<pair<double, unsigned int> >::iterator **ScoresMapIt**

scores map iterator

Definition at line 39 of file ReliefF.cpp.

7.31.1.5 typedef pair<unsigned int, **DatasetInstance**∗> **T**

functor for T comparison

Definition at line 58 of file ReliefF.cpp.

7.31.2 Function Documentation

7.31.2.1 bool **attributeSort** (const pair< unsigned int, double > & *p1*, const pair< unsigned int, double > & *p2*)

attribute index sorting functor

Definition at line 52 of file ReliefF.cpp.

7.31.2.2 void **librelieff_is_present** (void)

Definition at line 1073 of file ReliefF.cpp.

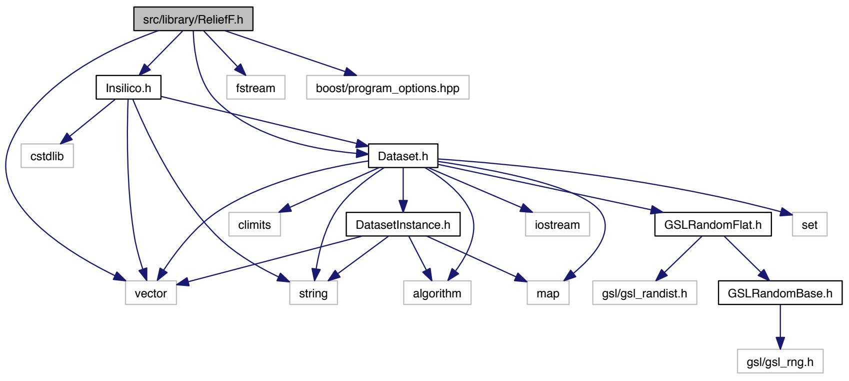
7.31.2.3 bool scoreSort (const pair< double, string > & *p1*, const pair< double, string > & *p2*)

attribute score sorting functor

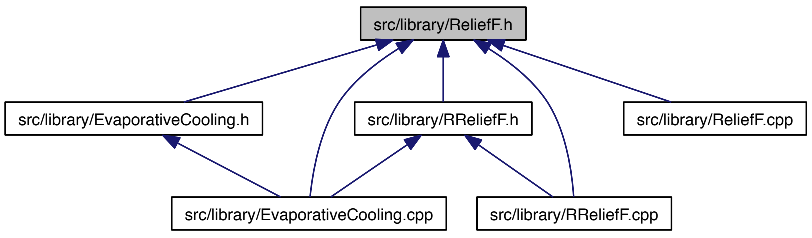
Definition at line 46 of file ReliefF.cpp.

7.32 src/library/ReliefF.h File Reference

`#include <vector>` `#include <fstream>` `#include <boost/program-_options.hpp>` `#include "Dataset.h"` `#include "Insilico.h"`
Include dependency graph for ReliefF.h:



This graph shows which files directly or indirectly include this file:



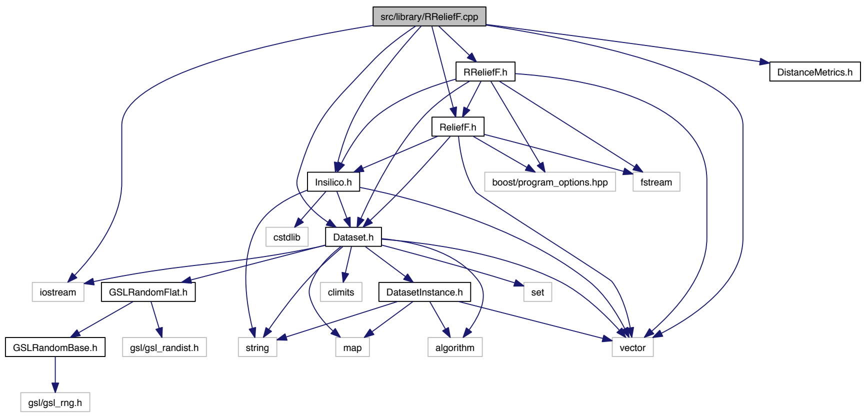
Classes

- class ReliefF

ReliefF attribute ranking algorithm.

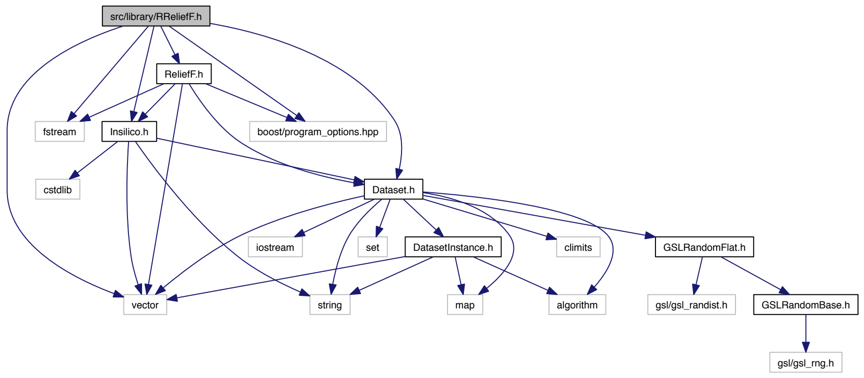
7.33 src/library/RReliefF.cpp File Reference

`#include <iostream>` `#include <vector>` `#include "ReliefF.-h"` `#include "RReliefF.h"` `#include "Dataset.h"` `#include "-DistanceMetrics.h"` `#include "Insilico.h"` Include dependency graph for RReliefF.cpp:

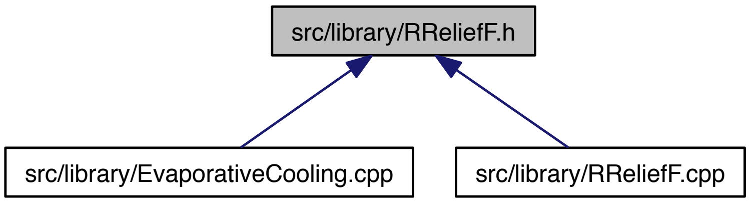


7.34 src/library/RReliefF.h File Reference

`#include <vector>` `#include <fstream>` `#include "Relief-F.h"` `#include "Dataset.h"` `#include "Insilico.h"` `#include <boost/program_options.hpp>` Include dependency graph for RReliefF.h:



This graph shows which files directly or indirectly include this file:



Classes

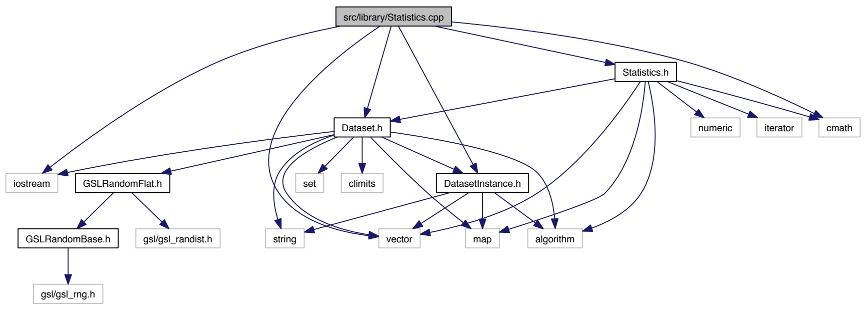
- class RReliefF

  Regression ReliefF attribute ranking algorithm.

7.35 src/library/Statistics.cpp File Reference

```
#include <iostream> #include <vector> #include <cmath>×
#include "Dataset.h" #include "DatasetInstance.h" #include
```

"`Statistics.h`" Include dependency graph for Statistics.cpp:



Defines

- #define DEBUG_Z 0
- #define DEBUG_E 1

Functions

- void PrintHistogram (Histogram histogram)

  Print a Histogram to cout.
- bool ZTransform (const VectorDouble &inputValues, VectorDouble &output-Values)

  ZTransform input values.
- double SelfEntropy (const vector< AttributeLevel > &a, const vector< Attribute-Level > &c)
- double Entropy (const vector< AttributeLevel > &sequenceValues)
- double condentropy (const vector< AttributeLevel > &X, const vector< Attribute-Level > &Y)
- double ConditionalEntropy (const vector< AttributeLevel > &sequenceValues, const vector< AttributeLevel > &givenValues)
- bool ConstructAttributeCart (const vector< AttributeLevel > &a, const vector< AttributeLevel > &b, vector< AttributeLevel > &ab)
- double KendallTau (vector< string > X, vector< string > Y)
- double KendallTau (vector< double > X, vector< double > Y)
- double KendallTau (vector< int > X, vector< int > Y)

7.35.1 Define Documentation

7.35.1.1 #define DEBUG_E 1

Definition at line 18 of file Statistics.cpp.

7.35.1.2 #define DEBUG_Z 0

Definition at line 17 of file Statistics.cpp.

7.35.2 Function Documentation

7.35.2.1 double condentropy (const vector< AttributeLevel > & *X*, const vector< AttributeLevel > & *Y*)

Definition at line 118 of file Statistics.cpp.

7.35.2.2 double ConditionalEntropy (const vector< AttributeLevel > & *sequenceValues*, const vector< AttributeLevel > & *givenValues*)

convert from base e to base 2

Definition at line 127 of file Statistics.cpp.

7.35.2.3 bool ConstructAttributeCart (const vector< AttributeLevel > & *a*, const vector< AttributeLevel > & *b*, vector< AttributeLevel > & *ab*)

Get the number of levels in a for a multiplier

Definition at line 208 of file Statistics.cpp.

7.35.2.4 double Entropy (const vector< AttributeLevel > & *sequenceValues*)

Definition at line 93 of file Statistics.cpp.

7.35.2.5 double KendallTau (vector< string > *X*, vector< string > *Y*)

Definition at line 248 of file Statistics.cpp.

7.35.2.6 double KendallTau (vector< double > *X*, vector< double > *Y*)

Definition at line 284 of file Statistics.cpp.

7.35.2.7 double KendallTau (vector< int > *X*, vector< int > *Y*)

Definition at line 318 of file Statistics.cpp.

7.35.2.8 void PrintHistogram (Histogram *histogram*)

Print a Histogram to cout.

Parameters

in	*histogram*	Histogram to print

Definition at line 20 of file Statistics.cpp.

7.35.2.9 double SelfEntropy (const vector< AttributeLevel > & *a*, const vector< AttributeLevel > & *c*)

Definition at line 87 of file Statistics.cpp.

7.35.2.10 bool ZTransform (const VectorDouble & *inputValues*, VectorDouble & *outputValues*)

ZTransform input values.

Parameters

in	*inputValues*	const vector of double input values
out	*output-Values*	transformed input values to z-scores with mean=0, stddev=1

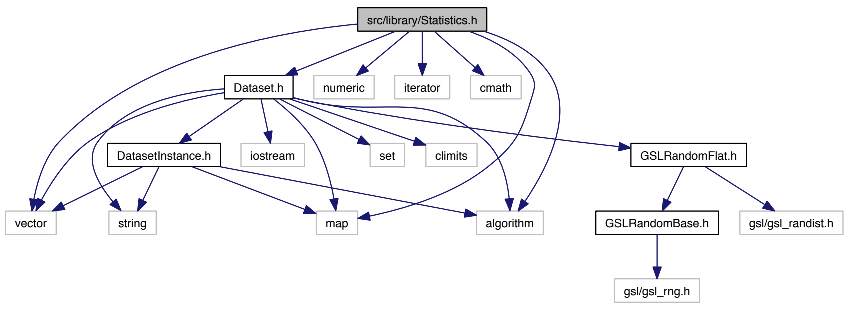
Returns

success

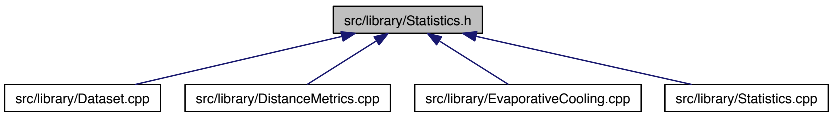
Definition at line 27 of file Statistics.cpp.

7.36 src/library/Statistics.h File Reference

`#include <vector>` `#include <map>` `#include <numeric>`×
`#include <iterator>` `#include <cmath>` `#include <algorithm>`×
`#include "Dataset.h"` Include dependency graph for Statistics.h:



This graph shows which files directly or indirectly include this file:


src/library/Statistics.h
src/library/Dataset.cpp
src/library/DistanceMetrics.cpp
src/library/EvaporativeCooling.cpp
src/library/Statistics.cpp

Typedefs

- typedef std::vector< double > VectorDouble

  vector of doubles type
- typedef std::vector< double > ::const_iterator VectorDoubleIt

  vector of doubles iterator
- typedef std::map < AttributeLevel, unsigned int > Histogram

  histogram type is a map: value->count
- typedef std::map < AttributeLevel, unsigned int > ::const_iterator HistogramIt

  historgram iterator

Functions

- void PrintHistogram (Histogram histogram)

  Print a Histogram to cout.
- bool ZTransform (const VectorDouble &inputValues, VectorDouble &output-Values)

  ZTransform input values.
- double SelfEntropy (const std::vector< AttributeLevel > &a, const std::vector< AttributeLevel > &c)

  Calculates the entropy of a sequence with itself and the class.
- double Entropy (const std::vector< AttributeLevel > &attributeValues)

  Calculates the entropy of a sequence of unsigned integers.
- double ConditionalEntropy (const std::vector< AttributeLevel > &attributeValues, const std::vector< AttributeLevel > &givenValues)

  Calculates the conditional entropy of a sequence of unsigned integers based (conditioned) on another sequence of unsigned integers (the givens).
- double condentropy (const std::vector< AttributeLevel > &X, const std::vector< AttributeLevel > &Y)
- bool ConstructAttributeCart (const std::vector< AttributeLevel > &a, const std::vector< AttributeLevel > &b, std::vector< AttributeLevel > &ab)

  Create a new attribute that is the cartesian product of a and b.
- double KendallTau (std::vector< std::string > X, std::vector< std::string > Y)

  Compute KendallTau for two ranked vectors of strings.

- double KendallTau (std::vector< double > X, std::vector< double > Y)

  Compute KendallTau for two ranked vectors of doubles.
- double KendallTau (std::vector< int > X, std::vector< int > Y)

  Compute KendallTau for two ranked vectors of integers.
- template<class T >
  std::pair< double, double > VarStd (std::vector< T > &values)

  Calculate variance and standard deviation of a vector of values.

7.36.1 Typedef Documentation

7.36.1.1 typedef std::map<**AttributeLevel**, unsigned int> **Histogram**

histogram type is a map: value->count

Definition at line 30 of file Statistics.h.

7.36.1.2 typedef std::map<**AttributeLevel**, unsigned int>::const_iterator **HistogramIt**

historgram iterator

Definition at line 32 of file Statistics.h.

7.36.1.3 typedef std::vector<double> **VectorDouble**

vector of doubles type

Definition at line 26 of file Statistics.h.

7.36.1.4 typedef std::vector<double>::const_iterator **VectorDoubleIt**

vector of doubles iterator

Definition at line 28 of file Statistics.h.

7.36.2 Function Documentation

7.36.2.1 double **condentropy** (const std::vector< **AttributeLevel** > & *X*, const std::vector< **AttributeLevel** > & *Y*)

7.36.2.2 double **ConditionalEntropy** (const std::vector< **AttributeLevel** > & *attributeValues*, const std::vector< **AttributeLevel** > & *givenValues*)

Calculates the conditional entropy of a sequence of unsigned integers based (conditioned) on another sequence of unsigned integers (the givens).

P(sequenceValues | givenValues)

Parameters

`in`	*attribute-Values*	vector of values
`in`	*givenValues*	vector of givens

Returns

conditional entropy as a double-precision float

7.36.2.3 bool **ConstructAttributeCart** (const std::vector< **AttributeLevel** > & *a*, const std::vector< **AttributeLevel** > & *b*, std::vector< **AttributeLevel** > & *ab*)

Create a new attribute that is the cartesian product of a and b.

NOTE: works for genotypes; need to verify for missign data levels, etc.

Parameters

`in`	*a*	attributes vector a
`in`	*b*	attributes vector b
`out`	*vector*	ab, the cartesian product of a and b

Returns

success

7.36.2.4 double **Entropy** (const std::vector< **AttributeLevel** > & *attributeValues*)

Calculates the entropy of a sequence of unsigned integers.

Parameters

`in`	*attribute-Values*	vector of sequence values - unsigned ints - positive categor-ical

Returns

entropy as a double-precision float

7.36.2.5 double **KendallTau** (std::vector< std::string > *X*, std::vector< std::string > *Y*)

Compute KendallTau for two ranked vectors of strings.

Why Kenall Tau - G. E. NOETHER `http://www.rsscse-edu.org.-uk/tsj/bts/noether/text.html`

Parameters

in	*X*	ranked attribute vector X
in	*Y*	ranked attribute vector Y

Returns

Kendall Tau value (-1, 1)

7.36.2.6 double **KendallTau** (std::vector< double > *X*, std::vector< double > *Y*)

Compute KendallTau for two ranked vectors of doubles.

Why Kenall Tau - G. E. NOETHER http://www.rsscse-edu.org.-uk/tsj/bts/noether/text.html

Parameters

in	*X*	ranked attribute vector X
in	*Y*	ranked attribute vector Y

Returns

Kendall Tau value (-1, 1)

7.36.2.7 double **KendallTau** (std::vector< int > *X*, std::vector< int > *Y*)

Compute KendallTau for two ranked vectors of integers.

Why Kenall Tau - G. E. NOETHER http://www.rsscse-edu.org.-uk/tsj/bts/noether/text.html

Parameters

in	*X*	ranked attribute vector X
in	*Y*	ranked attribute vector Y

Returns

Kendall Tau value (-1, 1)

7.36.2.8 void **PrintHistogram** (**Histogram** *histogram*)

Print a Histogram to cout.

Parameters

in	*histogram*	Histogram to print

Definition at line 20 of file Statistics.cpp.

7.36.2.9 double SelfEntropy (const std::vector< AttributeLevel > & *a*, const std::vector< AttributeLevel > & *c*)

Calculates the entropy of a sequence with itself and the class.

Parameters

in	*a*	vector of values
in	*c*	vector of class levels

Returns

entropy as a double-precision float

7.36.2.10 template<class T > std::pair<double, double> VarStd (std::vector< T > & *values*)

Calculate variance and standard deviation of a vector of values.

Parameters

in	*ranked*	attribute lists X and Y

Returns

Kendall Tau value (-1, 1)

Definition at line 116 of file Statistics.h.

7.36.2.11 bool ZTransform (const VectorDouble & *inputValues*, VectorDouble & *outputValues*)

ZTransform input values.

Parameters

in	*inputValues*	const vector of double input values
out	*output-Values*	transformed input values to z-scores with mean=0, stddev=1

Returns

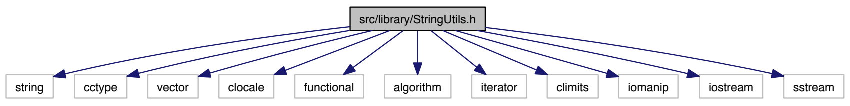
success

Definition at line 27 of file Statistics.cpp.

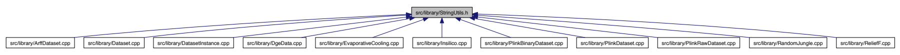
7.37 src/library/StringUtils.h File Reference

Various string-related utilities.

`#include <string>` `#include <cctype>` `#include <vector>` `#include <clocale>` `#include <functional>` `#include <algorithm>` × `#include <iterator>` `#include <climits>` `#include <iomanip>` × `#include <iostream>` `#include <sstream>` Include dependency graph for StringUtils.h:



This graph shows which files directly or indirectly include this file:


src/library/StringUtils.h
src/library/ArffDataset.cpp
src/library/Dataset.cpp
src/library/DatasetInstance.cpp
src/library/DgeData.cpp
src/library/EvaporativeCooling.cpp
src/library/Insilico.cpp
src/library/PlinkBinaryDataset.cpp
src/library/PlinkDataset.cpp
src/library/PlinkRawDataset.cpp
src/library/RandomJungle.cpp
src/library/ReliefF.cpp

Classes

- class insilico::is_classified< Type, charT >
- class insilico::do_to_upper< charT >
- class insilico::do_to_lower< charT >

Namespaces

- namespace insilico

Functions

- template<typename stringT >
  stringT insilico::trim_left (const stringT &s, const std::locale &loc=std::locale())

- template<typename stringT >
  stringT insilico::trim_right (const stringT &s, const std::locale &loc=std::locale())
- template<typename stringT >
  stringT insilico::trim (const stringT &s, const std::locale &loc=std::locale())
- template<typename Container , typename stringT >
  void insilico::split (Container &cont, const stringT &s, const std::locale &loc=std-::locale())
- template<typename Container , typename stringT >
  void insilico::split (Container &cont, const stringT &s, const stringT &delim)
- template<typename Container , typename stringT , typename Pred >
  void insilico::split_if (Container &cont, const stringT &s, const Pred &pred)
- template<typename It , typename stringT >
  stringT insilico::join (const It &begin, const It &end, const stringT &delim)
- template<typename stringT >
  stringT insilico::to_upper (const stringT &str, const std::locale &loc=std::locale())
- template<typename stringT >
  stringT insilico::to_lower (const stringT &str, const std::locale &loc=std::locale())
- std::string insilico::trim_left (const char ∗s, const std::locale &loc=std::locale())
- std::wstring insilico::trim_left (const wchar_t ∗s, const std::locale &loc=std-::locale())
- std::string insilico::trim_right (const char ∗s, const std::locale &loc=std::locale())
- std::wstring insilico::trim_right (const wchar_t ∗s, const std::locale &loc=std-::locale())
- std::string insilico::trim (const char ∗s, const std::locale &loc=std::locale())
- std::wstring insilico::trim (const wchar_t ∗s, const std::locale &loc=std::locale())
- template<typename Container >
  void insilico::split (Container &cont, const char ∗s, const std::locale &loc=std-::locale())
- template<typename Container >
  void insilico::split (Container &cont, const wchar_t ∗s, const std::locale &loc=std-::locale())
- template<typename Container >
  void insilico::split (Container &cont, const std::string &s, const char ∗delim)
- template<typename Container >
  void insilico::split (Container &cont, const char ∗s, const std::string &delim)
- template<typename Container >
  void insilico::split (Container &cont, const char ∗s, const char ∗delim)
- template<typename Container >
  void insilico::split (Container &cont, const std::wstring &s, const wchar_t ∗delim)
- template<typename Container >
  void insilico::split (Container &cont, const wchar_t ∗s, const std::wstring &delim)
- template<typename Container >
  void insilico::split (Container &cont, const wchar_t ∗s, const wchar_t ∗delim)
- template<typename Container , typename Pred >
  void insilico::split_if (Container &cont, const char ∗s, const Pred &pred)
- template<typename Container , typename Pred >
  void insilico::split_if (Container &cont, const wchar_t ∗s, const Pred &pred)

- template<typename It >
  std::string insilico::join (const It &begin, const It &end, const char ∗delim)
- template<typename It >
  std::wstring insilico::join (const It &begin, const It &end, const wchar_t ∗delim)
- std::string insilico::to_upper (const char ∗s, const std::locale &loc=std::locale())
- std::wstring insilico::to_upper (const wchar_t ∗s, const std::locale &loc=std::locale())
- std::string insilico::to_lower (const char ∗s, const std::locale &loc=std::locale())
- std::wstring insilico::to_lower (const wchar_t ∗s, const std::locale &loc=std::locale())
- template<typename T >
  std::string insilico::get_bits (T value)
- template<typename T >
  std::string insilico::zeroPadNumber (T num, int padSize)

7.37.1 Detailed Description

Various string-related utilities. This is originally from Nate Barney circa Moore Lab days 2003-2007. His function naming follows lowercase with underscores style, while my additions are camelCase.

Author

Bill White, Nate Barney

Version

1.0

Contact: `bill.c.white@gmail.com` Created on: 10/7/04

Definition in file StringUtils.h.