

Bayesian Cluster Tool

Generated by Doxygen 1.9.1

Mon May 15 2023 12:48:45

1 Todo List	1
2 Class Index	3
2.1 Class List	3
3 File Index	5
3.1 File List	5
4 Class Documentation	7
4.1 Cluster Class Reference	7
4.1.1 Detailed Description	8
4.1.2 Constructor & Destructor Documentation	8
4.1.2.1 Cluster()	8
4.1.3 Member Function Documentation	8
4.1.3.1 GetParent()	9
4.1.3.2 operator+=()	9
4.1.4 Member Data Documentation	9
4.1.4.1 mParams	9
4.2 Configuration Class Reference	10
4.2.1 Detailed Description	14
4.2.2 Member Function Documentation	14
4.2.2.1 alpha()	14
4.2.2.2 ClusterR()	15
4.2.2.3 ClusterT()	15
4.2.2.4 dR()	15
4.2.2.5 dT()	16
4.2.2.6 FromCommandline()	16
4.2.2.7 FromVector()	16
4.2.2.8 getCentreX()	17
4.2.2.9 getCentreY()	17
4.2.2.10 getInstance()	17
4.2.2.11 getWidthX()	18
4.2.2.12 getWidthY()	18
4.2.2.13 inputFile()	18
4.2.2.14 log_probability_sigma() [1/2]	19
4.2.2.15 log_probability_sigma() [2/2]	19
4.2.2.16 logAlpha()	19
4.2.2.17 logGammaAlpha()	20
4.2.2.18 logPb()	20
4.2.2.19 logPbDagger()	20
4.2.2.20 max2R()	21
4.2.2.21 max2R2()	21
4.2.2.22 maxR()	21

4.2.2.23 maxR2()	22
4.2.2.24 maxScanR()	22
4.2.2.25 maxScanT()	22
4.2.2.26 minScanR()	23
4.2.2.27 minScanT()	23
4.2.2.28 outputFile()	23
4.2.2.29 probability_sigma() [1/2]	24
4.2.2.30 probability_sigma() [2/2]	24
4.2.2.31 Rbins()	24
4.2.2.32 SetAlpha()	25
4.2.2.33 SetCentre()	25
4.2.2.34 SetInputFile()	25
4.2.2.35 SetOutputFile()	26
4.2.2.36 SetPb()	26
4.2.2.37 SetRBins()	26
4.2.2.38 SetSigmaParameters()	28
4.2.2.39 SetTBins()	28
4.2.2.40 SetValidate()	29
4.2.2.41 SetWidth()	29
4.2.2.42 sigmabins() [1/2]	30
4.2.2.43 sigmabins() [2/2]	30
4.2.2.44 sigmabins2() [1/2]	30
4.2.2.45 sigmabins2() [2/2]	31
4.2.2.46 sigmacount()	31
4.2.2.47 sigmaspacing()	31
4.2.2.48 Tbins()	32
4.2.2.49 validate()	32
4.2.3 Member Data Documentation	32
4.2.3.1 mPhysicalCentreX	32
4.3 Data Class Reference	33
4.3.1 Detailed Description	34
4.3.2 Constructor & Destructor Documentation	34
4.3.2.1 Data() [1/3]	34
4.3.2.2 Data() [2/3]	35
4.3.2.3 Data() [3/3]	35
4.3.3 Member Function Documentation	35
4.3.3.1 CalculateLocalizationScore()	35
4.3.3.2 dPhi()	36
4.3.3.3 dR()	36
4.3.3.4 dR2()	37
4.3.3.5 operator<()	37
4.3.3.6 operator=() [1/2]	38

4.3.3.7 operator=() [2/2]	38
4.3.3.8 Preprocess()	38
4.3.3.9 PreprocessLocalizationScores()	39
4.4 DataProxy Class Reference	39
4.4.1 Detailed Description	41
4.4.2 Constructor & Destructor Documentation	41
4.4.2.1 DataProxy() [1/3]	41
4.4.2.2 DataProxy() [2/3]	41
4.4.2.3 DataProxy() [3/3]	41
4.4.3 Member Function Documentation	42
4.4.3.1 Clusterize() [1/2]	42
4.4.3.2 Clusterize() [2/2]	42
4.4.3.3 GetCluster()	43
4.4.3.4 operator=() [1/2]	43
4.4.3.5 operator=() [2/2]	43
4.5 Event Class Reference	44
4.5.1 Detailed Description	45
4.5.2 Constructor & Destructor Documentation	45
4.5.2.1 Event() [1/2]	45
4.5.2.2 Event() [2/2]	45
4.5.3 Member Function Documentation	45
4.5.3.1 Clusterize()	45
4.5.3.2 LoadCSV()	46
4.5.3.3 operator=() [1/2]	46
4.5.3.4 operator=() [2/2]	47
4.5.3.5 ScanRT()	47
4.5.3.6 WriteCSV()	47
4.6 EventProxy Class Reference	48
4.6.1 Detailed Description	49
4.6.2 Constructor & Destructor Documentation	49
4.6.2.1 EventProxy() [1/3]	49
4.6.2.2 EventProxy() [2/3]	50
4.6.2.3 EventProxy() [3/3]	50
4.6.3 Member Function Documentation	50
4.6.3.1 CheckClusterization()	50
4.6.3.2 Clusterize()	51
4.6.3.3 GetData()	51
4.6.3.4 operator=() [1/2]	52
4.6.3.5 operator=() [2/2]	52
4.6.3.6 ScanRT()	52
4.7 GSLInterpolator Class Reference	53
4.7.1 Detailed Description	54

4.7.2 Constructor & Destructor Documentation	54
4.7.2.1 GSLInterpolator() [1/4]	54
4.7.2.2 GSLInterpolator() [2/4]	55
4.7.2.3 GSLInterpolator() [3/4]	55
4.7.2.4 GSLInterpolator() [4/4]	55
4.7.3 Member Function Documentation	56
4.7.3.1 Deriv()	56
4.7.3.2 Deriv2()	56
4.7.3.3 Eval()	56
4.7.3.4 Evaluate()	57
4.7.3.5 Integ()	57
4.7.3.6 operator=() [1/2]	58
4.7.3.7 operator=() [2/2]	58
4.7.3.8 SetData() [1/2]	59
4.7.3.9 SetData() [2/2]	59
4.8 Cluster::Parameter Struct Reference	60
4.8.1 Detailed Description	60
4.8.2 Member Function Documentation	60
4.8.2.1 alt_log_score()	61
4.8.2.2 log_score()	61
4.8.2.3 operator+=()	61
4.9 ProgressBar Struct Reference	62
4.9.1 Detailed Description	62
4.9.2 Constructor & Destructor Documentation	62
4.9.2.1 ProgressBar()	62
4.9.3 Member Function Documentation	63
4.9.3.1 operator++()	63
4.10 ProgressBar2 Struct Reference	63
4.10.1 Detailed Description	64
4.10.2 Constructor & Destructor Documentation	64
4.10.2.1 ProgressBar2()	64
4.10.3 Member Function Documentation	64
4.10.3.1 operator++()	64
4.11 PyIteator< U > Struct Template Reference	65
4.11.1 Detailed Description	65
4.11.2 Constructor & Destructor Documentation	66
4.11.2.1 PyIteator()	66
4.11.3 Member Function Documentation	66
4.11.3.1 next()	66
4.12 PyIteator< U * > Struct Template Reference	66
4.12.1 Detailed Description	67
4.12.2 Constructor & Destructor Documentation	67

4.12.2.1 PyIterator()	67
4.12.3 Member Function Documentation	68
4.12.3.1 next()	68
5 File Documentation	69
5.1 src/PythonBindings/PythonBindings.cpp File Reference	69
5.1.1 Detailed Description	70
5.1.2 Function Documentation	70
5.1.2.1 Cluster_GetIterator()	70
5.1.2.2 Cluster_GetSize()	71
5.1.2.3 ConfigFromVector()	71
5.1.2.4 Event_GetIterator()	72
5.1.2.5 Event_GetSize()	72
5.1.2.6 OneStopGetClusters()	72
5.1.2.7 py_list_to_std_vector()	73
5.1.2.8 std_vector_to_py_list()	73
Index	75

Chapter 1

Todo List

Member `Data::CalculateLocalizationScore` (`const std::vector< Data > &aData, const double &R`) `const`

Remind myself how this works and what the difference is with above

Member `Data::PreprocessLocalizationScores` (`std::vector< Data > &aData`)

Remind myself how this works and what the difference is with below

Class `PyIterator< U * >`

There must be an out-of-the box way, but I can't find it

Class `PyIterator< U >`

There must be an out-of-the box way, but I can't find it

Chapter 2

Class Index

2.1 Class List

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

Cluster	A class representing a cluster	7
Configuration	Class for storing the configuration parameters	10
Data	A class to store the raw data-points	33
DataProxy	A light-weight proxy for the raw data-points	39
Event	A class which holds the raw event data and global parameters	44
EventProxy	A lightweight wrapper for the event to store clusters for a given scan	48
GSLInterpolator	A utility wrapper around the GSL interpolator to give it a clean C++ interface	53
Cluster::Parameter	A struct representing the cluster parameters	60
ProgressBar	A utility progress-bar	62
ProgressBar2	A utility code timer	63
PyIterator< U >	A python iterator over a C++ container	65
PyIterator< U * >	A partial specialization creating a python iterator over a C++ container of pointers	66

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

include/BayesianClustering/ Cluster.hpp	??
include/BayesianClustering/ Configuration.hpp	??
include/BayesianClustering/ Data.hpp	??
include/BayesianClustering/ DataProxy.hpp	??
include/BayesianClustering/ Event.hpp	??
include/BayesianClustering/ EventProxy.hpp	??
include/BayesianClustering/ Precision.hpp	??
include/Utilities/ GSLInterpolator.hpp	??
include/Utilities/ ListComprehension.hpp	??
include/Utilities/ ProgressBar.hpp	??
include/Utilities/ Vectorize.hpp	??
src/ Cluster.cxx	??
src/ Scan.cxx	??
src/BayesianClustering/ Cluster.cpp	??
src/BayesianClustering/ Configuration.cpp	??
src/BayesianClustering/ Data.cpp	??
src/BayesianClustering/ DataProxy.cpp	??
src/BayesianClustering/ Event.cpp	??
src/BayesianClustering/ EventProxy.cpp	??
src/PythonBindings/ PythonBindings.cpp	
Self-contained sourcefile for producing python-bindings	69
src/Utilities/ GSLInterpolator.cpp	??
src/Utilities/ ProgressBar.cpp	??
src/Utilities/ Vectorize.cpp	??

Chapter 4

Class Documentation

4.1 Cluster Class Reference

A class representing a cluster.

```
#include <Cluster.hpp>
```

Collaboration diagram for Cluster:



Classes

- struct [Parameter](#)
A struct representing the cluster parameters.

Public Member Functions

- [Cluster](#) ()
Default constructor.
- [Cluster](#) (const [Data](#) &aData)
Construct a cluster from a single data-point.
- [Cluster](#) & [operator+=](#) (const [Cluster](#) &aOther)
Add another cluster to this one.
- [Cluster](#) * [GetParent](#) ()
Get a pointer to this cluster's ultimate parent.
- void [UpdateLogScore](#) ()
Update log-probability after a scan.

Public Attributes

- `std::vector< Parameter > mParams`
Get the points after clustering.
- `std::size_t mClusterSize`
The number of points in the current cluster.
- `std::size_t mLastClusterSize`
The number of points in the cluster on the previous scan iteration.
- `PRECISION mClusterScore`
The log-probability of the current cluster.
- `Cluster * mParent`
A pointer to the immediate parent of the current cluster.
- `std::vector< Data * > mData`
List of points in the cluster after clustering.

4.1.1 Detailed Description

A class representing a cluster.

Definition at line 14 of file Cluster.hpp.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 Cluster()

```
Cluster::Cluster (
    const Data & aData )
```

Construct a cluster from a single data-point.

Parameters

<i>aData</i>	A data-point with which to initialize the cluster
--------------	---

Definition at line 102 of file Cluster.cpp.

References `Configuration::Instance`, `mParams`, `Data::r2`, `Data::s`, `Data::x`, and `Data::y`.

4.1.3 Member Function Documentation

4.1.3.1 GetParent()

```
Cluster * Cluster::GetParent ( )
```

Get a pointer to this cluster's ultimate parent.

Returns

A pointer to this cluster's ultimate parent

Definition at line 161 of file Cluster.cpp.

References GetParent(), and mParent.

Referenced by DataProxy::GetCluster(), and GetParent().

4.1.3.2 operator+=()

```
Cluster & Cluster::operator+= (
    const Cluster & aOther )
```

Add another cluster to this one.

Parameters

<i>aOther</i>	Another cluster of parameters to add to this one
---------------	--

Returns

Reference to this, for chaining calls

Definition at line 151 of file Cluster.cpp.

References mClusterSize, and mParams.

4.1.4 Member Data Documentation

4.1.4.1 mParams

```
std::vector< Parameter > Cluster::mParams
```

Get the points after clustering.

Returns

Reference to a list of points in the cluster after clustering The collection of parameters, each corresponding to a different sigma hypothesis

Definition at line 82 of file Cluster.hpp.

Referenced by Cluster(), operator+=(), UpdateLogScore(), and EventProxy::ValidateLogScore().

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

- include/BayesianClustering/Cluster.hpp
- src/BayesianClustering/Cluster.cpp

4.2 Configuration Class Reference

Class for storing the configuration parameters.

```
#include <Configuration.hpp>
```

Collaboration diagram for Configuration:



Public Member Functions

- [Configuration](#) ()
Default constructor.
- void [SetCentre](#) (const double &aPhysicalCentreX, const double &aPhysicalCentreY)
Setter for the centre of the scan window.
- void [SetWidth](#) (const double &aWidthX, const double &aWidthY)
Setter for the size of the RoI window.
- void [SetSigmaParameters](#) (const std::size_t &aSigmaCount, const double &aSigmaMin, const double &aSigmaMax, const std::function< double(const double &) > &aInterpolator)
Setter for the sigma-bins to be integrated over.
- void [SetRBins](#) (const std::size_t &aRBins, const double &aMinScanR=0.0, const double &aMaxScanR=-1)
Setter for the R bins for the RT scan.
- void [SetTBins](#) (const std::size_t &aTBins, const double &aMinScanT=0.0, const double &aMaxScanT=-1)
- void [SetPb](#) (const double &aPB)
Setter for the P_b parameter.
- void [SetAlpha](#) (const double &aAlpha)
Setter for the alpha parameter.
- void [SetValidate](#) (const bool &aValidate)

- Set whether to validate clusterization.*

 - void [SetInputFile](#) (const std::string &aFileName)

Setter for the input file.
- void [SetOutputFile](#) (const std::string &aFileName)

Setter for the output file.
- void [FromCommandline](#) (int argc, char **argv)

Parse the parameters when passed in as commandline arguments.
- void [FromVector](#) (const std::vector< std::string > &Args)

Parse the parameters when passed in as commandline arguments.
- const std::size_t & [sigmacount](#) () const

Getter for the scale-parameter squared.
- const double & [sigmaspacing](#) () const

Getter for the sigma spacing.
- const std::vector< double > & [sigmabins](#) () const

Getter for the values of sigma.
- const std::vector< double > & [sigmabins2](#) () const

Getter for the values of sigma squared.
- const std::vector< double > & [probability_sigma](#) () const

Getter for the probabilities of a given sigma.
- const std::vector< double > & [log_probability_sigma](#) () const

Getter for the log of the probabilities of a given sigma.
- const double & [sigmabins](#) (const std::size_t &i) const

Getter for the i'th value of sigma.
- const double & [sigmabins2](#) (const std::size_t &i) const

Getter for the i'th value of sigma squared.
- const double & [probability_sigma](#) (const std::size_t &i) const

Getter for the probability of the i'th value of sigma.
- const double & [log_probability_sigma](#) (const std::size_t &i) const

Getter for the log-probability of the i'th value of sigma.
- const double & [maxR](#) () const

Getter for the maximum value of R.
- const double & [maxR2](#) () const

Getter for the maximum value of R squared.
- const double & [max2R](#) () const

Getter for the maximum value of 2R.
- const double & [max2R2](#) () const

Getter for the maximum value of 2R squared.
- const double & [minScanR](#) () const

Getter for the lowest value of R to scan.
- const double & [maxScanR](#) () const

Getter for the highest value of R to scan.
- const double & [minScanT](#) () const

Getter for the lowest value of T to scan.
- const double & [maxScanT](#) () const

Getter for the highest value of T to scan.
- const double & [dR](#) () const

Getter for the spacing of value of R to scan.
- const std::size_t & [Rbins](#) () const

Getter for the number of R values to scan.
- const double & [dT](#) () const

Getter for the spacing of value of T to scan.

- `const std::size_t & Tbins () const`
Getter for the number of T values to scan.
- `const double & logPb () const`
Logarithm of the P_b parameter
- `const double & logPbDagger () const`
Logarithm of the (1 - P_b) parameter
- `const double & alpha () const`
Getter for the alpha parameter
- `const double & logAlpha () const`
Getter for the logarithm of the alpha parameter
- `const double & logGammaAlpha () const`
Getter for the logarithm of the gamma function of alpha parameter
- `const bool & validate () const`
Getter for whether or not to run the validation on the clustering.
- `const std::string & inputFile () const`
Getter for the input file.
- `const std::string & outputFile () const`
Getter for the output file.
- `const double & ClusterR () const`
Getter for the R value for a clusterization pass.
- `const double & ClusterT () const`
Getter for the T value for a clusterization pass.
- `double getCentreX () const`
Getter for the x-coordinate of the physical centre.
- `double getCentreY () const`
Getter for the y-coordinate of the physical centre.
- `double getWidthX () const`
Getter for the scaling factor applied to the dataset.
- `double getWidthY () const`
Getter for the height of the ROI window.

Static Public Member Functions

- `static Configuration & getInstance ()`
Getter for the singleton instance.

Static Public Attributes

- `static Configuration Instance`
A single global copy of the global variables.

Private Attributes

- double [mPhysicalCentreX](#)
The scale parameter.
- double [mPhysicalCentreY](#)
The y-coordinate of the centre of the window in physical units.
- double [mWidthX](#)
The width of the window in the x-direction in physical units.
- double [mWidthY](#)
The width of the window in the y-direction in physical units.
- std::size_t [mSigmacount](#)
The number of sigma bins.
- double [mSigmaspacing](#)
The spacing of sigma bins.
- std::vector< double > [mSigmabins](#)
The values of sigma.
- std::vector< double > [mSigmabins2](#)
The values of sigma squared.
- std::vector< double > [mProbabilitySigma](#)
The probability of a given sigma.
- std::vector< double > [mLogProbabilitySigma](#)
The log-probability of a given sigma.
- double [mMaxR](#)
The maximum value of R.
- double [mMaxR2](#)
The maximum value of R squared.
- double [mMax2R](#)
The maximum value of 2R
- double [mMax2R2](#)
The maximum value of 2R squared.
- double [mMinScanR](#)
The lowest value of R to scan.
- double [mMaxScanR](#)
The largest value of R to scan.
- double [mMinScanT](#)
The lowest value of T to scan.
- double [mMaxScanT](#)
The largest value of T to scan.
- double [mDR](#)
The spacing of value of R to scan.
- double [mDT](#)
The spacing of value of T to scan.
- std::size_t [mRbins](#)
The number of R values to scan.
- std::size_t [mTbins](#)
The number of T values to scan.
- double [mAlpha](#)
The alpha parameter.
- double [mLogAlpha](#)
Logarithm of the alpha parameter.

- double [mLogGammaAlpha](#)
Logarithm of the gamma function of alpha parameter
- double [mLogPb](#)
Logarithm of the P_b parameter
- double [mLogPbDagger](#)
Logarithm of the $(1 - P_b)$ parameter
- bool [mValidate](#)
Whether or not to run the validation on the clustering.
- std::string [mInputFile](#)
The input event file.
- std::string [mOutputFile](#)
The output file.
- double [mClusterR](#)
The value of R for clustering.
- double [mClusterT](#)
The value of T for clustering.

4.2.1 Detailed Description

Class for storing the configuration parameters.

Definition at line 71 of file Configuration.hpp.

4.2.2 Member Function Documentation

4.2.2.1 `alpha()`

```
const double& Configuration::alpha ( ) const [inline]
```

Getter for the alpha parameter

Returns

The alpha parameter

Definition at line 229 of file Configuration.hpp.

References `mAlpha`.

4.2.2.2 ClusterR()

```
const double& Configuration::ClusterR ( ) const [inline]
```

Getter for the R value for a clusterization pass.

Returns

The R value for a clusterization pass

Definition at line 252 of file Configuration.hpp.

References mClusterR.

4.2.2.3 ClusterT()

```
const double& Configuration::ClusterT ( ) const [inline]
```

Getter for the T value for a clusterization pass.

Returns

The T value for a clusterization pass

Definition at line 255 of file Configuration.hpp.

References mClusterT.

4.2.2.4 dR()

```
const double& Configuration::dR ( ) const [inline]
```

Getter for the spacing of value of R to scan.

Returns

The spacing of value of R to scan

Definition at line 209 of file Configuration.hpp.

References mDR.

Referenced by Data::PreprocessLocalizationScores().

4.2.2.5 dT()

```
const double& Configuration::dT ( ) const [inline]
```

Getter for the spacing of value of T to scan.

Returns

The spacing of value of T to scan

Definition at line 215 of file Configuration.hpp.

References mDT.

Referenced by EventProxy::ScanRT().

4.2.2.6 FromCommandline()

```
void Configuration::FromCommandline (
    int argc,
    char ** argv )
```

Parse the parameters when passed in as commandline arguments.

Parameters

<i>argc</i>	The number of commandline arguments
<i>argv</i>	The commandline arguments

Definition at line 151 of file Configuration.cpp.

References FromVector().

4.2.2.7 FromVector()

```
void Configuration::FromVector (
    const std::vector< std::string > & aArgs )
```

Parse the parameters when passed in as commandline arguments.

Parameters

<i>aArgs</i>	The commandline arguments
--------------	---------------------------

Definition at line 157 of file Configuration.cpp.

References `GSLInterpolator::Eval()`, `mClusterR`, `mClusterT`, `SetAlpha()`, `SetCentre()`, `SetInputFile()`, `SetOutputFile()`, `SetPb()`, `SetRBins()`, `SetSigmaParameters()`, `SetTBins()`, `SetValidate()`, and `SetWidth()`.

Referenced by `ConfigFromVector()`, and `FromCommandline()`.

4.2.2.8 `getCentreX()`

```
double Configuration::getCentreX ( ) const [inline]
```

Getter for the x-coordinate of the physical centre.

Returns

The x-coordinate of the physical centre

Definition at line 308 of file `Configuration.hpp`.

References `mPhysicalCentreX`.

Referenced by `Event::WriteCSV()`.

4.2.2.9 `getCentreY()`

```
double Configuration::getCentreY ( ) const [inline]
```

Getter for the y-coordinate of the physical centre.

Returns

The y-coordinate of the physical centre

Definition at line 311 of file `Configuration.hpp`.

References `mPhysicalCentreY`.

Referenced by `Event::WriteCSV()`.

4.2.2.10 `getInstance()`

```
static Configuration& Configuration::getInstance ( ) [inline], [static]
```

Getter for the singleton instance.

Returns

The singleton instance

Definition at line 331 of file `Configuration.hpp`.

References `Instance`.

4.2.2.11 getWidthX()

```
double Configuration::getWidthX ( ) const [inline]
```

Getter for the scaling factor applied to the dataset.

Returns

The scaling factor applied to the dataset
Getter for the width of the ROI window
The width of the ROI window

Definition at line 319 of file Configuration.hpp.

References mWidthX.

4.2.2.12 getWidthY()

```
double Configuration::getWidthY ( ) const [inline]
```

Getter for the height of the ROI window.

Returns

The height of the ROI window

Definition at line 323 of file Configuration.hpp.

References mWidthY.

4.2.2.13 inputFile()

```
const std::string& Configuration::inputFile ( ) const [inline]
```

Getter for the input file.

Returns

The name of the input event file

Definition at line 244 of file Configuration.hpp.

References mInputFile.

Referenced by Event::Event().

4.2.2.14 log_probability_sigma() [1/2]

```
const std::vector< double >& Configuration::log_probability_sigma ( ) const [inline]
```

Getter for the log of the probabilities of a given sigma.

Returns

The log of the probabilities of given sigma

Definition at line 162 of file Configuration.hpp.

References mLogProbabilitySigma.

Referenced by Cluster::UpdateLogScore().

4.2.2.15 log_probability_sigma() [2/2]

```
const double& Configuration::log_probability_sigma (
    const std::size_t & i ) const [inline]
```

Getter for the log-probability of the i'th value of sigma.

Parameters

<i>i</i>	The index of the value of sigma to get the log-probability for
----------	--

Returns

The log-probability of sigma_i

Definition at line 179 of file Configuration.hpp.

References mLogProbabilitySigma.

4.2.2.16 logAlpha()

```
const double& Configuration::logAlpha ( ) const [inline]
```

Getter for the logarithm of the alpha parameter

Returns

The logarithm of the alpha parameter

Definition at line 232 of file Configuration.hpp.

References mLogAlpha.

4.2.2.17 logGammaAlpha()

```
const double& Configuration::logGammaAlpha ( ) const [inline]
```

Getter for the logarithm of the gamma function of alpha parameter

Returns

The logarithm of the gamma function of alpha parameter

Definition at line 235 of file Configuration.hpp.

References mLogGammaAlpha.

4.2.2.18 logPb()

```
const double& Configuration::logPb ( ) const [inline]
```

Logarithm of the P_b parameter

Returns

Logarithm of the P_b parameter

Definition at line 222 of file Configuration.hpp.

References mLogPb.

Referenced by EventProxy::UpdateLogScore().

4.2.2.19 logPbDagger()

```
const double& Configuration::logPbDagger ( ) const [inline]
```

Logarithm of the (1 - P_b) parameter

Returns

Logarithm of the (1 - P_b) parameter

Definition at line 225 of file Configuration.hpp.

References mLogPbDagger.

4.2.2.20 max2R()

```
const double& Configuration::max2R ( ) const [inline]
```

Getter for the maximum value of 2R.

Returns

The maximum value of 2R

Definition at line 189 of file Configuration.hpp.

References mMax2R.

Referenced by Data::Preprocess().

4.2.2.21 max2R2()

```
const double& Configuration::max2R2 ( ) const [inline]
```

Getter for the maximum value of 2R squared.

Returns

The maximum value of 2R squared

Definition at line 192 of file Configuration.hpp.

References mMax2R2.

4.2.2.22 maxR()

```
const double& Configuration::maxR ( ) const [inline]
```

Getter for the maximum value of R.

Returns

The maximum value of R

Definition at line 183 of file Configuration.hpp.

References mMaxR.

4.2.2.23 maxR2()

```
const double& Configuration::maxR2 ( ) const [inline]
```

Getter for the maximum value of R squared.

Returns

The maximum value of R squared

Definition at line 186 of file Configuration.hpp.

References mMaxR2.

4.2.2.24 maxScanR()

```
const double& Configuration::maxScanR ( ) const [inline]
```

Getter for the highest value of R to scan.

Returns

The highest value of R to scan

Definition at line 199 of file Configuration.hpp.

References mMaxScanR.

4.2.2.25 maxScanT()

```
const double& Configuration::maxScanT ( ) const [inline]
```

Getter for the highest value of T to scan.

Returns

The highest value of T to scan

Definition at line 205 of file Configuration.hpp.

References mMaxScanT.

Referenced by EventProxy::ScanRT().

4.2.2.26 minScanR()

```
const double& Configuration::minScanR ( ) const [inline]
```

Getter for the lowest value of R to scan.

Returns

The lowest value of R to scan

Definition at line 196 of file Configuration.hpp.

References mMinScanR.

4.2.2.27 minScanT()

```
const double& Configuration::minScanT ( ) const [inline]
```

Getter for the lowest value of T to scan.

Returns

The lowest value of T to scan

Definition at line 202 of file Configuration.hpp.

References mMinScanT.

4.2.2.28 outputFile()

```
const std::string& Configuration::outputFile ( ) const [inline]
```

Getter for the output file.

Returns

The name of the output file

Definition at line 247 of file Configuration.hpp.

References mOutputFile.

4.2.2.29 probability_sigma() [1/2]

```
const std::vector< double >& Configuration::probability_sigma ( ) const [inline]
```

Getter for the probabilities of a given sigma.

Returns

The probabilities of given sigma

Definition at line 159 of file Configuration.hpp.

References mProbabilitySigma.

4.2.2.30 probability_sigma() [2/2]

```
const double& Configuration::probability_sigma (
    const std::size_t & i ) const [inline]
```

Getter for the probability of the i'th value of sigma.

Parameters

<i>i</i>	The index of the value of sigma to get the probability for
----------	--

Returns

The probability of sigma_i

Definition at line 175 of file Configuration.hpp.

References mProbabilitySigma.

4.2.2.31 Rbins()

```
const std::size_t& Configuration::Rbins ( ) const [inline]
```

Getter for the number of R values to scan.

Returns

The number of R values to scan

Definition at line 212 of file Configuration.hpp.

References mRbins.

Referenced by Data::PreprocessLocalizationScores(), and EventProxy::ScanRT().

4.2.2.32 SetAlpha()

```
void Configuration::SetAlpha (
    const double & aAlpha )
```

Setter for the alpha parameter.

Parameters

<i>aAlpha</i>	The alpha parameter
---------------	---------------------

Definition at line 104 of file Configuration.cpp.

References mAlpha, mLogAlpha, and mLogGammaAlpha.

Referenced by FromVector().

4.2.2.33 SetCentre()

```
void Configuration::SetCentre (
    const double & aPhysicalCentreX,
    const double & aPhysicalCentreY )
```

Setter for the centre of the scan window.

Parameters

<i>aPhysicalCentreX</i>	The x-coordinate of the centre of the window in physical units (becomes 0 in algorithm units)
<i>aPhysicalCentreY</i>	The y-coordinate of the centre of the window in physical units (becomes 0 in algorithm units)

Definition at line 37 of file Configuration.cpp.

References mPhysicalCentreX, and mPhysicalCentreY.

Referenced by FromVector().

4.2.2.34 SetInputFile()

```
void Configuration::SetInputFile (
    const std::string & aFileName )
```

Setter for the input file.

Parameters

<i>aFileName</i>	The name of the file
------------------	----------------------

Definition at line 120 of file Configuration.cpp.

References mInputFile.

Referenced by FromVector().

4.2.2.35 SetOutputFile()

```
void Configuration::SetOutputFile (
    const std::string & aFileName )
```

Setter for the output file.

Parameters

<i>aFileName</i>	The name of the file
------------------	----------------------

Definition at line 127 of file Configuration.cpp.

References mOutputFile.

Referenced by FromVector().

4.2.2.36 SetPb()

```
void Configuration::SetPb (
    const double & aPB )
```

Setter for the P_b parameter.

Parameters

<i>aPB</i>	The P_b parameter
------------	-------------------

Definition at line 97 of file Configuration.cpp.

References mLogPb, and mLogPbDagger.

Referenced by FromVector().

4.2.2.37 SetRBins()

```
void Configuration::SetRBins (
    const std::size_t & aRbins,
```

```
const double & aMinScanR = 0.0,  
const double & aMaxScanR = -1 )
```

Setter for the R bins for the RT scan.

Parameters

<i>aRbins</i>	The number of R bins to scan over
<i>aMinScanR</i>	The lowest value of R to scan
<i>aMaxScanR</i>	The largest value of R to scan

Definition at line 72 of file Configuration.cpp.

References mDR, mMax2R, mMax2R2, mMaxR, mMaxR2, mMaxScanR, mMinScanR, and mRbins.

Referenced by FromVector().

4.2.2.38 SetSigmaParameters()

```
void Configuration::SetSigmaParameters (
    const std::size_t & aSigmacount,
    const double & aSigmaMin,
    const double & aSigmaMax,
    const std::function< double(const double &) > & aInterpolator )
```

Setter for the sigma-bins to be integrated over.

Parameters

<i>aSigmacount</i>	The number of sigma bins
<i>aSigmaMin</i>	The lowest sigma bin
<i>aSigmaMax</i>	The highest sigma bin
<i>aInterpolator</i>	Function-object to generate the probability of any given sigma

Definition at line 58 of file Configuration.cpp.

References mLogProbabilitySigma, mProbabilitySigma, mSigmbins, mSigmbins2, mSigmacount, and m↔
Sigmaspacing.

Referenced by FromVector().

4.2.2.39 SetTBins()

```
void Configuration::SetTBins (
    const std::size_t & aTbins,
    const double & aMinScanT = 0.0,
    const double & aMaxScanT = -1 )
```

Parameters

<i>aTbins</i>	The number of T bins to scan over
<i>aMinScanT</i>	The lowest value of T to scan
<i>aMaxScanT</i>	The largest value of T to scan

Definition at line 87 of file Configuration.cpp.

References mDT, mMaxScanT, mMinScanT, and mTbins.

Referenced by FromVector().

4.2.2.40 SetValidate()

```
void Configuration::SetValidate (
    const bool & aValidate )
```

Set whether to validate clusterization.

Parameters

<i>aValidate</i>	Whether to validate clusterization
------------------	------------------------------------

Definition at line 112 of file Configuration.cpp.

References mValidate.

Referenced by FromVector().

4.2.2.41 SetWidth()

```
void Configuration::SetWidth (
    const double & aWidthX,
    const double & aWidthY )
```

Setter for the size of the RoI window.

Parameters

<i>aWidthX</i>	The width of the window in physical units
<i>aWidthY</i>	The height of the window in physical units

Definition at line 51 of file Configuration.cpp.

References mWidthX, and mWidthY.

Referenced by FromVector().

4.2.2.42 sigmabins() [1/2]

```
const std::vector< double >& Configuration::sigmabins ( ) const [inline]
```

Getter for the values of sigma.

Returns

The values of sigma

Definition at line 153 of file Configuration.hpp.

References mSigmabins.

4.2.2.43 sigmabins() [2/2]

```
const double& Configuration::sigmabins (
    const std::size_t & i ) const [inline]
```

Getter for the i'th value of sigma.

Parameters

<i>i</i>	The index of the value of sigma to get
----------	--

Returns

The value of sigma_i

Definition at line 167 of file Configuration.hpp.

References mSigmabins.

4.2.2.44 sigmabins2() [1/2]

```
const std::vector< double >& Configuration::sigmabins2 ( ) const [inline]
```

Getter for the values of sigma squared.

Returns

The values of sigma squared

Definition at line 156 of file Configuration.hpp.

References mSigmabins2.

4.2.2.45 sigmabins2() [2/2]

```
const double& Configuration::sigmabins2 (
    const std::size_t & i ) const [inline]
```

Getter for the i'th value of sigma squared.

Parameters

<i>i</i>	The index of the value of sigma squared to get
----------	--

Returns

The value of sigma_i squared

Definition at line 171 of file Configuration.hpp.

References mSigmabins2.

4.2.2.46 sigmacount()

```
const std::size_t& Configuration::sigmacount ( ) const [inline]
```

Getter for the scale-parameter squared.

Returns

The scale-parameter squared Getter for the sigma count

The sigma count

Definition at line 145 of file Configuration.hpp.

References mSigmacount.

Referenced by Cluster::UpdateLogScore(), and EventProxy::ValidateLogScore().

4.2.2.47 sigmaspacing()

```
const double& Configuration::sigmaspacing ( ) const [inline]
```

Getter for the sigma spacing.

Returns

The sigma spacing

Definition at line 149 of file Configuration.hpp.

References mSigmaspacing.

4.2.2.48 Tbins()

```
const std::size_t& Configuration::Tbins ( ) const [inline]
```

Getter for the number of T values to scan.

Returns

The number of T values to scan

Definition at line 218 of file Configuration.hpp.

References mTbins.

Referenced by EventProxy::ScanRT().

4.2.2.49 validate()

```
const bool& Configuration::validate ( ) const [inline]
```

Getter for whether or not to run the validation on the clustering.

Returns

Whether or not to run the validation on the clustering

Definition at line 239 of file Configuration.hpp.

References mValidate.

4.2.3 Member Data Documentation

4.2.3.1 mPhysicalCentreX

```
double Configuration::mPhysicalCentreX [private]
```

The scale parameter.

The scale parameter squared The x-coordinate of the centre of the window in physical units

Definition at line 345 of file Configuration.hpp.

Referenced by getCentreX(), and SetCentre().

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

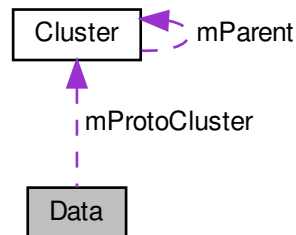
- include/BayesianClustering/Configuration.hpp
- src/BayesianClustering/Configuration.cpp
- src/BayesianClustering/Event.cpp

4.3 Data Class Reference

A class to store the raw data-points.

```
#include <Data.hpp>
```

Collaboration diagram for Data:



Public Member Functions

- [Data](#) (const PRECISION &aX, const PRECISION &aY, const PRECISION &aS)
Constructor.
- [Data](#) (const [Data](#) &aOther)=delete
Deleted copy constructor.
- [Data](#) & [operator=](#) (const [Data](#) &aOther)=delete
Deleted assignment operator.
- [Data](#) ([Data](#) &&aOther)=default
Default move constructor.
- [Data](#) & [operator=](#) ([Data](#) &&aOther)=default
Default move-assignment constructor.
- virtual [~Data](#) ()
Destructor.
- bool [operator<](#) (const [Data](#) &aOther) const
Comparison operator for sorting data-points by distance from the origin.
- PRECISION [dR2](#) (const [Data](#) &aOther) const
Return the squared-distance of this data-points from another.
- PRECISION [dR](#) (const [Data](#) &aOther) const
Return the distance of this data-points from another.
- PRECISION [dPhi](#) (const [Data](#) &aOther) const
Return the angle between this data-points and another.
- void [Preprocess](#) (std::vector< [Data](#) > &aData, const std::size_t &aIndex)
All the necessary pre-processing to get this data-point ready for an RT-scan.
- void [PreprocessLocalizationScores](#) (std::vector< [Data](#) > &aData)
Calculate the localization score from the local neighbourhood.
- PRECISION [CalculateLocalizationScore](#) (const std::vector< [Data](#) > &aData, const double &R) const
Calculate the localization score from the local neighbourhood.

Public Attributes

- PRECISION [x](#)
The x-position of the data-point.
- PRECISION [y](#)
The y-position of the data-point.
- PRECISION [s](#)
The sigma of the data-point
- PRECISION [r2](#)
The squared radial distance of the data-point.
- PRECISION [r](#)
The radial distance of the data-point.
- PRECISION [phi](#)
The phi-position of the data-point.
- `std::vector< PRECISION >` [mLocalizationScores](#)
The locaalization scores, one per R-bin.
- `std::vector< std::pair< PRECISION, std::size_t > >` [mNeighbours](#)
The list of neighbours as a pair of squared-distance and index into the list of points.
- [Cluster](#) * [mProtoCluster](#)
A cluster containing only this data-point.

4.3.1 Detailed Description

A class to store the raw data-points.

Definition at line 14 of file Data.hpp.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 Data() [1/3]

```
Data::Data (
    const PRECISION & aX,
    const PRECISION & aY,
    const PRECISION & aS )
```

Constructor.

Parameters

aX	The x-position of the data-point in algorithm units
aY	The y-position of the data-point in algorithm units
aS	The sigma of the data-point in algorithm units

Definition at line 13 of file Data.cpp.

4.3.2.2 Data() [2/3]

```
Data::Data (
    const Data & aOther ) [delete]
```

Deleted copy constructor.

Parameters

<i>aOther</i>	Anonymous argument
---------------	--------------------

4.3.2.3 Data() [3/3]

```
Data::Data (
    Data && aOther ) [default]
```

Default move constructor.

Parameters

<i>aOther</i>	Anonymous argument
---------------	--------------------

4.3.3 Member Function Documentation

4.3.3.1 CalculateLocalizationScore()

```
PRECISION Data::CalculateLocalizationScore (
    const std::vector< Data > & aData,
    const double & R ) const
```

Calculate the localization score from the local neighbourhood.

Todo Remind myself how this works and what the difference is with above

Parameters

<i>aData</i>	?
<i>R</i>	?

Returns

The localization score

Definition at line 109 of file Data.cpp.

References mNeighbours, x, and y.

4.3.3.2 dPhi()

```
PRECISION Data::dPhi (
    const Data & aOther ) const [inline]
```

Return the angle between this data-points and another.

Returns

The angle between this data-points and another

Parameters

<i>aOther</i>	A data-point to compare against
---------------	---------------------------------

Definition at line 68 of file Data.hpp.

References phi.

4.3.3.3 dR()

```
PRECISION Data::dR (
    const Data & aOther ) const [inline]
```

Return the distance of this data-points from another.

Returns

The distance of this data-points from another

Parameters

<i>aOther</i>	A data-point to compare against
---------------	---------------------------------

Definition at line 60 of file Data.hpp.

References dR2().

4.3.3.4 dR2()

```
PRECISION Data::dR2 (
    const Data & aOther ) const [inline]
```

Return the squared-distance of this data-points from another.

Returns

The squared-distance of this data-points from another

Parameters

<i>aOther</i>	A data-point to compare against
---------------	---------------------------------

Definition at line 51 of file Data.hpp.

References x, and y.

Referenced by dR().

4.3.3.5 operator<()

```
bool Data::operator< (
    const Data & aOther ) const [inline]
```

Comparison operator for sorting data-points by distance from the origin.

Returns

Whether this data-point is closer to the origin than another

Parameters

<i>aOther</i>	A data-point to compare against
---------------	---------------------------------

Definition at line 43 of file Data.hpp.

References r.

4.3.3.6 operator=() [1/2]

```
Data& Data::operator= (
    const Data & aOther ) [delete]
```

Deleted assignment operator.

Returns

Reference to this, for chaining calls

Parameters

<i>aOther</i>	Anonymous argument
---------------	--------------------

4.3.3.7 operator=() [2/2]

```
Data& Data::operator= (
    Data && aOther ) [default]
```

Default move-assignment constructor.

Returns

Reference to this, for chaining calls

Parameters

<i>aOther</i>	Anonymous argument
---------------	--------------------

4.3.3.8 Preprocess()

```
void Data::Preprocess (
    std::vector< Data > & aData,
    const std::size_t & aIndex )
```

All the necessary pre-processing to get this data-point ready for an RT-scan.

Parameters

<i>aData</i>	The collection of data-points
<i>aIndex</i>	The index of the current data-point

Definition at line 27 of file Data.cpp.

References Configuration::Instance, and Configuration::max2R().

4.3.3.9 PreprocessLocalizationScores()

```
void Data::PreprocessLocalizationScores (
    std::vector< Data > & aData )
```

Calculate the localization score from the local neighbourhood.

Todo Remind myself how this works and what the difference is with below

Parameters

<i>aData</i>	?
--------------	---

Definition at line 71 of file Data.cpp.

References Configuration::dR(), Configuration::Instance, and Configuration::Rbins().

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

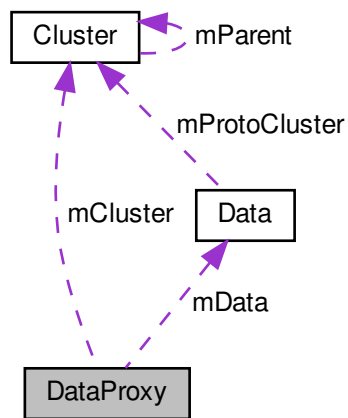
- include/BayesianClustering/Data.hpp
- src/BayesianClustering/Data.cpp

4.4 DataProxy Class Reference

A light-weight proxy for the raw data-points.

```
#include <DataProxy.hpp>
```

Collaboration diagram for DataProxy:



Public Member Functions

- [DataProxy](#) ([Data](#) &aData)
Default constructor.
- [DataProxy](#) (const [DataProxy](#) &aOther)=delete
Deleted copy constructor.
- [DataProxy](#) & operator= (const [DataProxy](#) &aOther)=delete
Deleted assignment operator.
- [DataProxy](#) ([DataProxy](#) &&aOther)=default
Default move constructor.
- [DataProxy](#) & operator= ([DataProxy](#) &&aOther)=default
Default move-assignment constructor.
- void [Clusterize](#) (const PRECISION &a2R2, [EventProxy](#) &aEvent)
Entry point clusterization function - a new cluster will be created.
- void [Clusterize](#) (const PRECISION &a2R2, [EventProxy](#) &aEvent, [Cluster](#) *aCluster)
Recursive clusterization function.
- [Cluster](#) * [GetCluster](#) ()
Get a pointer to this data-proxy's ultimate parent cluster (or null if unclustered).

Public Attributes

- [Data](#) * [mData](#)
The data-point for which this is the proxy.
- [Cluster](#) * [mCluster](#)
This data-proxy's immediate parent cluster.
- bool [mExclude](#)
Whether this data-point is to be included in the clusterization.

4.4.1 Detailed Description

A light-weight proxy for the raw data-points.

Definition at line 17 of file DataProxy.hpp.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 DataProxy() [1/3]

```
DataProxy::DataProxy (
    Data & aData )
```

Default constructor.

Parameters

<i>aData</i>	The data-point for which this is the proxy
--------------	--

Definition at line 10 of file DataProxy.cpp.

4.4.2.2 DataProxy() [2/3]

```
DataProxy::DataProxy (
    const DataProxy & aOther ) [delete]
```

Deleted copy constructor.

Parameters

<i>aOther</i>	Anonymous argument
---------------	--------------------

4.4.2.3 DataProxy() [3/3]

```
DataProxy::DataProxy (
    DataProxy && aOther ) [default]
```

Default move constructor.

Parameters

<i>aOther</i>	Anonymous argument
---------------	--------------------

4.4.3 Member Function Documentation

4.4.3.1 Clusterize() [1/2]

```
void DataProxy::Clusterize (
    const PRECISION & a2R2,
    EventProxy & aEvent )
```

Entry point clusterization function - a new cluster will be created.

Parameters

<i>a2R2</i>	The clusterization radius
<i>aEvent</i>	The event-proxy in which we are running

Definition at line 15 of file DataProxy.cpp.

References mCluster, EventProxy::mClusters, and mExclude.

Referenced by Clusterize().

4.4.3.2 Clusterize() [2/2]

```
void DataProxy::Clusterize (
    const PRECISION & a2R2,
    EventProxy & aEvent,
    Cluster * aCluster )
```

Recursive clusterization function.

Parameters

<i>a2R2</i>	The clusterization radius
<i>aEvent</i>	The event-proxy in which we are running
<i>aCluster</i>	The cluster we are building

Definition at line 23 of file DataProxy.cpp.

References Clusterize(), GetCluster(), EventProxy::GetData(), mCluster, Cluster::mClusterSize, mData, mExclude, Data::mNeighbours, Cluster::mParent, and Data::mProtoCluster.

4.4.3.3 GetCluster()

```
Cluster* DataProxy::GetCluster ( ) [inline]
```

Get a pointer to this data-proxy's ultimate parent cluster (or null if unclustered).

Returns

A pointer to this data-proxy's ultimate parent cluster

Definition at line 51 of file DataProxy.hpp.

References Cluster::GetParent(), and mCluster.

Referenced by Clusterize().

4.4.3.4 operator=() [1/2]

```
DataProxy& DataProxy::operator= (
    const DataProxy & aOther ) [delete]
```

Deleted assignment operator.

Returns

Reference to this, for chaining calls

Parameters

<i>aOther</i>	Anonymous argument
---------------	--------------------

4.4.3.5 operator=() [2/2]

```
DataProxy& DataProxy::operator= (
    DataProxy && aOther ) [default]
```

Default move-assignment constructor.

Returns

Reference to this, for chaining calls

Parameters

<i>aOther</i>	Anonymous argument
---------------	--------------------

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

- include/BayesianClustering/DataProxy.hpp
- src/BayesianClustering/DataProxy.cpp

4.5 Event Class Reference

A class which holds the raw event data and global parameters.

```
#include <Event.hpp>
```

Public Member Functions

- [Event](#) ()
Default Constructor.
- [Event](#) (const [Event](#) &aOther)=delete
Deleted copy constructor.
- [Event](#) & [operator=](#) (const [Event](#) &aOther)=delete
Deleted assignment operator.
- [Event](#) ([Event](#) &&aOther)=default
Default move constructor.
- [Event](#) & [operator=](#) ([Event](#) &&aOther)=default
Default move-assignment constructor.
- void [Preprocess](#) ()
All the necessary pre-processing to get the event ready for an RT-scan.
- void [ScanRT](#) (const std::function< void(const [EventProxy](#) &, const double &, const double &, std::pair< int, int >) > &aCallback)
Run the scan.
- void [Clusterize](#) (const double &R, const double &T, const std::function< void(const [EventProxy](#) &) > &aCallback)
Run clusterization for a specific choice of R and T.
- void [LoadCSV](#) (const std::string &aFilename)
Load an event from given file.
- void [WriteCSV](#) (const std::string &aFilename)
Save an event to a file.

Public Attributes

- std::vector< [Data](#) > [mData](#)
The collection of raw data points.

4.5.1 Detailed Description

A class which holds the raw event data and global parameters.

Definition at line 16 of file Event.hpp.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 Event() [1/2]

```
Event::Event (
    const Event & aOther ) [delete]
```

Deleted copy constructor.

Parameters

<i>aOther</i>	Anonymous argument
---------------	--------------------

4.5.2.2 Event() [2/2]

```
Event::Event (
    Event && aOther ) [default]
```

Default move constructor.

Parameters

<i>aOther</i>	Anonymous argument
---------------	--------------------

4.5.3 Member Function Documentation

4.5.3.1 Clusterize()

```
void Event::Clusterize (
    const double & R,
    const double & T,
    const std::function< void(const EventProxy &) > & aCallback )
```

Run clusterization for a specific choice of R and T.

Parameters

<i>R</i>	The R parameter for clusterization
<i>T</i>	The T parameter for clusterization
<i>aCallback</i>	A callback for the clusterization results

Definition at line 49 of file Event.cpp.

References EventProxy::Clusterize(), and Preprocess().

Referenced by OneStopGetClusters().

4.5.3.2 LoadCSV()

```
void Event::LoadCSV (
    const std::string & aFilename )
```

Load an event from given file.

Parameters

<i>aFilename</i>	The name of the file to load
------------------	------------------------------

Definition at line 108 of file Event.cpp.

References mData.

Referenced by Event().

4.5.3.3 operator=() [1/2]

```
Event& Event::operator= (
    const Event & aOther ) [delete]
```

Deleted assignment operator.

Returns

Reference to this, for chaining calls

Parameters

<i>aOther</i>	Anonymous argument
---------------	--------------------

4.5.3.4 operator=() [2/2]

```
Event& Event::operator= (
    Event && aOther ) [default]
```

Default move-assignment constructor.

Returns

Reference to this, for chaining calls

Parameters

<i>aOther</i>	Anonymous argument
---------------	--------------------

4.5.3.5 ScanRT()

```
void Event::ScanRT (
    const std::function< void(const EventProxy &, const double &, const double &,
std::pair< int, int >) > & aCallback )
```

Run the scan.

Parameters

<i>aCallback</i>	A callback for each RT-scan result
------------------	------------------------------------

Definition at line 33 of file Event.cpp.

References mData, and Preprocess().

4.5.3.6 WriteCSV()

```
void Event::WriteCSV (
    const std::string & aFilename )
```

Save an event to a file.

Parameters

<i>aFilename</i>	The name of the file to which to save
------------------	---------------------------------------

Definition at line 146 of file Event.cpp.

References Configuration::getCentreX(), Configuration::getCentreY(), Configuration::Instance, and mData.

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

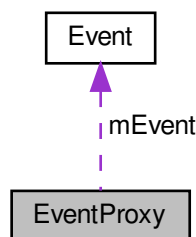
- include/BayesianClustering/Event.hpp
- src/BayesianClustering/Event.cpp

4.6 EventProxy Class Reference

A lightweight wrapper for the event to store clusters for a given scan.

```
#include <EventProxy.hpp>
```

Collaboration diagram for EventProxy:



Public Member Functions

- [EventProxy](#) ([Event](#) &aEvent)
Default constructor.
- [EventProxy](#) (const [EventProxy](#) &aOther)=delete
Deleted copy constructor.
- [EventProxy](#) & [operator=](#) (const [EventProxy](#) &aOther)=delete
Deleted assignment operator.
- [EventProxy](#) ([EventProxy](#) &&aOther)=default
Default move constructor.
- [EventProxy](#) & [operator=](#) ([EventProxy](#) &&aOther)=default
Default move-assignment constructor.
- void [CheckClusterization](#) (const double &R, const double &T)
Run validation tests on the clusters.
- void [ScanRT](#) (const std::function< void(const [EventProxy](#) &, const double &, const double &, std::pair< int, int >) > &aCallback, const uint8_t &aParallelization=1, const uint8_t &aOffset=0)
Run an RT-scan.
- void [Clusterize](#) (const double &R, const double &T, const std::function< void(const [EventProxy](#) &) > &aCallback)

- *Run clusterization for a specific choice of R and T.*
void [UpdateLogScore](#) ()
- *Update log-probability after a scan.*
void [ValidateLogScore](#) ()
- *Sean's validation code for testing when the running log-score fails.*
[DataProxy](#) & [GetData](#) (const std::size_t &aIndex)
- *Get the proxy for the Nth neighbour of this data-point.*

Public Attributes

- std::vector< [DataProxy](#) > [mData](#)
The collection of lightweight data-point wrappers used by this event wrapper.
- std::vector< [Cluster](#) > [mClusters](#)
The collection of clusters found by this scan.
- std::size_t [mClusteredCount](#)
The number of clustered data-points.
- std::size_t [mBackgroundCount](#)
The number of background data-points.
- std::size_t [mClusterCount](#)
The number of non-Null clusters.
- double [mLogP](#)
The log-probability density associated with the last scan.

Private Attributes

- const [Event](#) & [mEvent](#)
The underlying event this is a proxy to.

4.6.1 Detailed Description

A lightweight wrapper for the event to store clusters for a given scan.

Definition at line 16 of file EventProxy.hpp.

4.6.2 Constructor & Destructor Documentation

4.6.2.1 EventProxy() [1/3]

```
EventProxy::EventProxy (
    Event & aEvent )
```

Default constructor.

Parameters

<i>aEvent</i>	An event for which this is a lightweight proxy
---------------	--

Definition at line 17 of file EventProxy.cpp.

References mClusters, Event::mData, and mData.

4.6.2.2 EventProxy() [2/3]

```
EventProxy::EventProxy (
    const EventProxy & aOther ) [delete]
```

Deleted copy constructor.

Parameters

<i>aOther</i>	Anonymous argument
---------------	--------------------

4.6.2.3 EventProxy() [3/3]

```
EventProxy::EventProxy (
    EventProxy && aOther ) [default]
```

Default move constructor.

Parameters

<i>aOther</i>	Anonymous argument
---------------	--------------------

4.6.3 Member Function Documentation

4.6.3.1 CheckClusterization()

```
void EventProxy::CheckClusterization (
    const double & R,
    const double & T )
```

Run validation tests on the clusters.

Parameters

R	The R of the last run scan
T	The T of the last run scan

Definition at line 25 of file EventProxy.cpp.

References `GetData()`, `mBackgroundCount`, `mClusterCount`, `mClusters`, and `mData`.

4.6.3.2 Clusterize()

```
void EventProxy::Clusterize (
    const double & R,
    const double & T,
    const std::function< void(const EventProxy &) > & aCallback )
```

Run clusterization for a specific choice of R and T .

Parameters

R	The R parameter for clusterization
T	The T parameter for clusterization
$aCallback$	A callback for the clusterization results

Definition at line 135 of file EventProxy.cpp.

References `mClusters`, `Event::mData`, `mData`, `mEvent`, and `UpdateLogScore()`.

Referenced by `Event::Clusterize()`.

4.6.3.3 GetData()

```
DataProxy& EventProxy::GetData (
    const std::size_t & aIndex ) [inline]
```

Get the proxy for the N th neighbour of this data-point.

Returns

A reference to the neighbour data-proxy

Parameters

$aIndex$	The index of the neighbour we are looking for
----------	---

Definition at line 63 of file EventProxy.hpp.

References mData.

Referenced by CheckClusterization(), and DataProxy::Clusterize().

4.6.3.4 operator=() [1/2]

```
EventProxy& EventProxy::operator= (
    const EventProxy & aOther ) [delete]
```

Deleted assignment operator.

Returns

Reference to this, for chaining calls

Parameters

<i>aOther</i>	Anonymous argument
---------------	--------------------

4.6.3.5 operator=() [2/2]

```
EventProxy& EventProxy::operator= (
    EventProxy && aOther ) [default]
```

Default move-assignment constructor.

Returns

Reference to this, for chaining calls

Parameters

<i>aOther</i>	Anonymous argument
---------------	--------------------

4.6.3.6 ScanRT()

```
void EventProxy::ScanRT (
    const std::function< void(const EventProxy &, const double &, const double &,
std::pair< int, int >) > & aCallback,
```

```
const uint8_t & aParallelization = 1,
const uint8_t & aOffset = 0 )
```

Run an RT-scan.

Parameters

<i>aCallback</i>	A callback for each RT-scan result
<i>aParallelization</i>	The stride with which we will iterate across RT parameters
<i>aOffset</i>	The starting point for the strides as we iterate across RT parameters

Definition at line 97 of file EventProxy.cpp.

References `Configuration::dT()`, `Configuration::Instance`, `Configuration::maxScanT()`, `Configuration::Rbins()`, and `Configuration::Tbins()`.

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

- include/BayesianClustering/EventProxy.hpp
- src/BayesianClustering/EventProxy.cpp

4.7 GSLInterpolator Class Reference

A utility wrapper around the GSL interpolator to give it a clean C++ interface.

```
#include <GSLInterpolator.hpp>
```

Public Member Functions

- [GSLInterpolator](#) (const gsl_interp_type *type, const unsigned int &ndata)
Empty splice constructor.
- [GSLInterpolator](#) (const gsl_interp_type *type, const std::vector< double > &x, const std::vector< double > &y)
Initialised splice constructor.
- virtual [~GSLInterpolator](#) ()
Destructor.
- [GSLInterpolator](#) (const [GSLInterpolator](#) &aOther)=delete
Deleted copy constructor.
- [GSLInterpolator](#) & [operator=](#) (const [GSLInterpolator](#) &aOther)=delete
Deleted assignment operator.
- [GSLInterpolator](#) ([GSLInterpolator](#) &&aOther)=default
Default move constructor.
- [GSLInterpolator](#) & [operator=](#) ([GSLInterpolator](#) &&aOther)=default
Default move-assignment constructor.
- bool [SetData](#) (const std::vector< double > &x, const std::vector< double > &y)
Set the spline data points.
- bool [SetData](#) (const unsigned int &ndata, const double *x, const double *y)
Set the spline data points.
- double [Evaluate](#) (const std::function< int(double &) > &aFunction, const std::string &aName)

Utility function that runs the GSL function that has been wrapped in a lambda below.

- double [Eval](#) (const double &x)

Evaluate the spline at the given x.

- double [Deriv](#) (const double &x)

The first derivative of the spline at the given x.

- double [Deriv2](#) (const double &x)

The second derivative of the spline at the given x.

- double [Integ](#) (const double &a, const double &b)

The integral over the spline between two bounds.

Private Attributes

- unsigned int [nErrors](#)

An error counter to suppress excess messages.

- gsl_interp_accel * [fAccel](#)

Underlying GSL machinery.

- gsl_spline * [fSpline](#)

Underlying GSL machinery for the spline itself.

- const gsl_interp_type * [fInterpType](#)

Underlying GSL machinery for the interpolation type.

4.7.1 Detailed Description

A utility wrapper around the GSL interpolator to give it a clean C++ interface.

Definition at line 18 of file `GSLInterpolator.hpp`.

4.7.2 Constructor & Destructor Documentation

4.7.2.1 `GSLInterpolator()` [1/4]

```
GSLInterpolator::GSLInterpolator (
    const gsl_interp_type * type,
    const unsigned int & ndata )
```

Empty splice constructor.

Parameters

<i>type</i>	The spline type
<i>ndata</i>	The number of points that will be added to the spline

Definition at line 7 of file `GSLInterpolator.cpp`.

References [fInterpType](#), and [fSpline](#).

4.7.2.2 GSLInterpolator() [2/4]

```
GSLInterpolator::GSLInterpolator (
    const gsl_interp_type * type,
    const std::vector< double > & x,
    const std::vector< double > & y )
```

Initialised splice constructor.

Parameters

<i>type</i>	The spline type
<i>x</i>	The points on the x-axis
<i>y</i>	The points on the y-axis

Definition at line 17 of file GSLInterpolator.cpp.

References fInterpType, fSpline, and SetData().

4.7.2.3 GSLInterpolator() [3/4]

```
GSLInterpolator::GSLInterpolator (
    const GSLInterpolator & aOther ) [delete]
```

Deleted copy constructor.

Parameters

<i>aOther</i>	Anonymous argument
---------------	--------------------

4.7.2.4 GSLInterpolator() [4/4]

```
GSLInterpolator::GSLInterpolator (
    GSLInterpolator && aOther ) [default]
```

Default move constructor.

Parameters

<i>aOther</i>	Anonymous argument
---------------	--------------------

4.7.3 Member Function Documentation

4.7.3.1 Deriv()

```
double GSLInterpolator::Deriv (
    const double & x ) [inline]
```

The first derivative of the spline at the given x.

Parameters

x	The x-coordinate at which to evaluate the derivative
---	--

Returns

The first derivative of the spline at the given x-coordinate

Definition at line 100 of file GSLInterpolator.hpp.

References Evaluate(), fAccel, and fSpline.

4.7.3.2 Deriv2()

```
double GSLInterpolator::Deriv2 (
    const double & x ) [inline]
```

The second derivative of the spline at the given x.

Parameters

x	The x-coordinate at which to evaluate the derivative
---	--

Returns

The second derivative of the spline at the given x-coordinate

Definition at line 108 of file GSLInterpolator.hpp.

References Evaluate(), fAccel, and fSpline.

4.7.3.3 Eval()

```
double GSLInterpolator::Eval (
    const double & x ) [inline]
```

Evaluate the spline at the given x.

Parameters

<i>x</i>	The x-coordinate at which to evaluate the spline
----------	--

Returns

The value of the spline at the given x-coordinate

Definition at line 92 of file GSLInterpolator.hpp.

References Evaluate(), fAccel, and fSpline.

Referenced by Configuration::FromVector().

4.7.3.4 Evaluate()

```
double GSLInterpolator::Evaluate (
    const std::function< int(double &) > & aFunction,
    const std::string & aName ) [inline]
```

Utility function that runs the GSL function that has been wrapped in a lambda below.

Parameters

<i>aFunction</i>	A lambda that will be evaluated
<i>aName</i>	The operation name for the debugging messages

Returns

The interpolated value

Definition at line 73 of file GSLInterpolator.hpp.

References fAccel, and nErrors.

Referenced by Deriv(), Deriv2(), Eval(), and Integ().

4.7.3.5 Integ()

```
double GSLInterpolator::Integ (
    const double & a,
    const double & b ) [inline]
```

The integral over the spline between two bounds.

Parameters

<i>a</i>	The lower bound of the integral
<i>b</i>	The upper bound of the integral

Returns

The integral over the spline between a and b

Definition at line 117 of file GSLInterpolator.hpp.

References Evaluate(), fAccel, and fSpline.

4.7.3.6 operator=() [1/2]

```
GSLInterpolator& GSLInterpolator::operator= (
    const GSLInterpolator & aOther ) [delete]
```

Deleted assignment operator.

Returns

Reference to this, for chaining calls

Parameters

<i>aOther</i>	Anonymous argument
---------------	--------------------

4.7.3.7 operator=() [2/2]

```
GSLInterpolator& GSLInterpolator::operator= (
    GSLInterpolator && aOther ) [default]
```

Default move-assignment constructor.

Returns

Reference to this, for chaining calls

Parameters

<i>aOther</i>	Anonymous argument
---------------	--------------------

4.7.3.8 SetData() [1/2]

```
bool GSLInterpolator::SetData (
    const std::vector< double > & x,
    const std::vector< double > & y ) [inline]
```

Set the spline data points.

Parameters

<i>x</i>	The x-coordinates of the datapoints
<i>y</i>	The y-coordinates of the datapoints

Returns

success or fail

Definition at line 56 of file GSLInterpolator.hpp.

Referenced by GSLInterpolator().

4.7.3.9 SetData() [2/2]

```
bool GSLInterpolator::SetData (
    const unsigned int & ndata,
    const double * x,
    const double * y )
```

Set the spline data points.

Parameters

<i>ndata</i>	The number of data points
<i>x</i>	Pointer to the first element of an array of x-coordinates
<i>y</i>	Pointer to the first element of an array of y-coordinates

Returns

success or fail

Definition at line 36 of file GSLInterpolator.cpp.

References fAccel, fInterpType, fSpline, and nErrors.

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

- include/Utilities/GSLInterpolator.hpp
- src/Utilities/GSLInterpolator.cpp

4.8 Cluster::Parameter Struct Reference

A struct representing the cluster parameters.

```
#include <Cluster.hpp>
```

Public Member Functions

- [Parameter](#) ()
Default constructor.
- [Parameter](#) & [operator+=](#) (const [Parameter](#) &aOther)
Add another set of parameters to this set.
- double [log_score](#) () const
Convert the parameters to a log-probability.
- double [alt_log_score](#) () const
Sean's alternative function to calculate the log-score using only the A's and B's as per the original paper for debugging.

Public Attributes

- PRECISION [A](#)
[Parameter](#) A defined in the math.
- PRECISION [Bx](#)
[Parameter](#) Bx defined in the math.
- PRECISION [By](#)
[Parameter](#) By defined in the math.
- PRECISION [C](#)
[Parameter](#) C defined in the math.
- PRECISION [logF](#)
[Parameter](#) logF defined in the math.
- PRECISION [weightedCentreX](#)
Parameters added by Sean for validation.
- PRECISION [weightedCentreY](#)
Parameters added by Sean for validation.
- PRECISION [S2](#)
Parameters added by Sean for validation.

4.8.1 Detailed Description

A struct representing the cluster parameters.

Definition at line 19 of file Cluster.hpp.

4.8.2 Member Function Documentation

4.8.2.1 alt_log_score()

```
double Cluster::Parameter::alt_log_score ( ) const
```

Sean's alternative function to calculate the log-score using only the A's and B's as per the original paper for debugging.

Returns

the log-probability of this set of cluster parameters

Definition at line 50 of file Cluster.cpp.

4.8.2.2 log_score()

```
double Cluster::Parameter::log_score ( ) const
```

Convert the parameters to a log-probability.

Returns

the log-probability of this set of cluster parameters

Definition at line 75 of file Cluster.cpp.

4.8.2.3 operator+=()

```
Cluster::Parameter & Cluster::Parameter::operator+= (
    const Parameter & aOther )
```

Add another set of parameters to this set.

Parameters

<i>aOther</i>	Another set of parameters to add to this set
---------------	--

Returns

Reference to this, for chaining calls

Definition at line 32 of file Cluster.cpp.

References A, Bx, By, C, and logF.

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

- include/BayesianClustering/Cluster.hpp
- src/BayesianClustering/Cluster.cpp

4.9 ProgressBar Struct Reference

A utility progress-bar.

```
#include <ProgressBar.hpp>
```

Public Member Functions

- [ProgressBar](#) (const std::string &aLabel, const uint32_t &aMax)
Constructor.
- virtual [~ProgressBar](#) ()
Destructor.
- void [operator++](#) ()
Postfix increment.
- void [operator++](#) (int aDummy)
Prefix increment.

Public Attributes

- float [mBlockSize](#)
The size of each increment.
- float [mNextThreshold](#)
The next threshold at which we will write a block to stdout.
- std::size_t [mCount](#)
The number of times we have incremented.
- std::chrono::high_resolution_clock::time_point [mStart](#)
A timer for end-of-task stats.

4.9.1 Detailed Description

A utility progress-bar.

Definition at line 6 of file ProgressBar.hpp.

4.9.2 Constructor & Destructor Documentation

4.9.2.1 ProgressBar()

```
ProgressBar::ProgressBar (  
    const std::string & aLabel,  
    const uint32_t & aMax )
```

Constructor.

Parameters

<i>aLabel</i>	A description of the task being timed
<i>aMax</i>	The number of calls equalling 100%

Definition at line 7 of file ProgressBar.cpp.

4.9.3 Member Function Documentation

4.9.3.1 operator++()

```
void ProgressBar::operator++ (
    int aDummy )
```

Prefix increment.

Parameters

<i>aDummy</i>	Anonymous argument
---------------	--------------------

Definition at line 27 of file ProgressBar.cpp.

References operator++().

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

- include/Utilities/ProgressBar.hpp
- src/Utilities/ProgressBar.cpp

4.10 ProgressBar2 Struct Reference

A utility code timer.

```
#include <ProgressBar.hpp>
```

Public Member Functions

- [ProgressBar2](#) (const std::string &aLabel, const uint32_t &aMax)
Constructor.
- virtual [~ProgressBar2](#) ()
Destructor.
- void [operator++](#) ()
Postfix increment.
- void [operator++](#) (int aDummy)
Prefix increment.

Public Attributes

- `std::chrono::high_resolution_clock::time_point` [mStart](#)
A timer for end-of-task stats.

4.10.1 Detailed Description

A utility code timer.

Definition at line 34 of file `ProgressBar.hpp`.

4.10.2 Constructor & Destructor Documentation

4.10.2.1 ProgressBar2()

```
ProgressBar2::ProgressBar2 (
    const std::string & aLabel,
    const uint32_t & aMax )
```

Constructor.

Parameters

<i>aLabel</i>	A description of the task being timed
<i>aMax</i>	The number of calls equalling 100%

Definition at line 32 of file `ProgressBar.cpp`.

4.10.3 Member Function Documentation

4.10.3.1 operator++()

```
void ProgressBar2::operator++ (
    int aDummy )
```

Prefix increment.

Parameters

<i>aDummy</i>	Anonymous argument
---------------	--------------------

Definition at line 44 of file ProgressBar.cpp.

References operator++().

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

- include/Utilities/ProgressBar.hpp
- src/Utilities/ProgressBar.cpp

4.11 PyIterator< U > Struct Template Reference

A python iterator over a C++ container.

Public Member Functions

- [PyIterator](#) (const std::vector< U > &aData)
Constructor.
- const U & [next](#) ()
Return the current value and advance the iterator.

Public Attributes

- std::vector< U >::const_iterator [mIt](#)
The current location of the iterator.
- const std::vector< U >::const_iterator [mEnd](#)
The end of the underlying container.

4.11.1 Detailed Description

```
template<typename U>
struct PyIterator< U >
```

A python iterator over a C++ container.

Todo There must be an out-of-the box way, but I can't find it

Template Parameters

<i>U</i>	The type of the data in the C++ container
----------	---

Definition at line 45 of file PythonBindings.cpp.

4.11.2 Constructor & Destructor Documentation

4.11.2.1 PyIterator()

```
template<typename U >
PyIterator< U >::PyIterator (
    const std::vector< U > & aData ) [inline]
```

Constructor.

Parameters

<i>aData</i>	The underlying data to be iterated over
--------------	---

Definition at line 54 of file PythonBindings.cpp.

4.11.3 Member Function Documentation

4.11.3.1 next()

```
template<typename U >
const U& PyIterator< U >::next ( ) [inline]
```

Return the current value and advance the iterator.

Returns

The current value

Definition at line 58 of file PythonBindings.cpp.

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

- src/PythonBindings/[PythonBindings.cpp](#)

4.12 PyIterator< U * > Struct Template Reference

A partial specialization creating a python iterator over a C++ container of pointers.

Public Member Functions

- [PyIterator](#) (const std::vector< U * > &aData)
Constructor.
- const U & [next](#) ()
Return the current value and advance the iterator.

Public Attributes

- std::vector< U * >::const_iterator [mIt](#)
The current location of the iterator.
- const std::vector< U * >::const_iterator [mEnd](#)
The end of the underlying container.

4.12.1 Detailed Description

```
template<typename U>
struct PyIterator< U * >
```

A partial specialization creating a python iterator over a C++ container of pointers.

Todo There must be an out-of-the box way, but I can't find it

Template Parameters

<i>U</i>	The type of the pointers in the C++ container
----------	---

Definition at line 74 of file PythonBindings.cpp.

4.12.2 Constructor & Destructor Documentation

4.12.2.1 PyIterator()

```
template<typename U >
PyIterator< U * >::PyIterator (
    const std::vector< U * > & aData ) [inline]
```

Constructor.

Parameters

<i>aData</i>	The underlying data to be iterated over
--------------	---

Definition at line 83 of file PythonBindings.cpp.

4.12.3 Member Function Documentation

4.12.3.1 next()

```
template<typename U >
const U& PyIterator< U * >::next ( ) [inline]
```

Return the current value and advance the iterator.

Returns

The current value

Definition at line 87 of file PythonBindings.cpp.

References `PyIterator< U >::mEnd`, and `PyIterator< U >::mIt`.

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

- `src/PythonBindings/PythonBindings.cpp`

Chapter 5

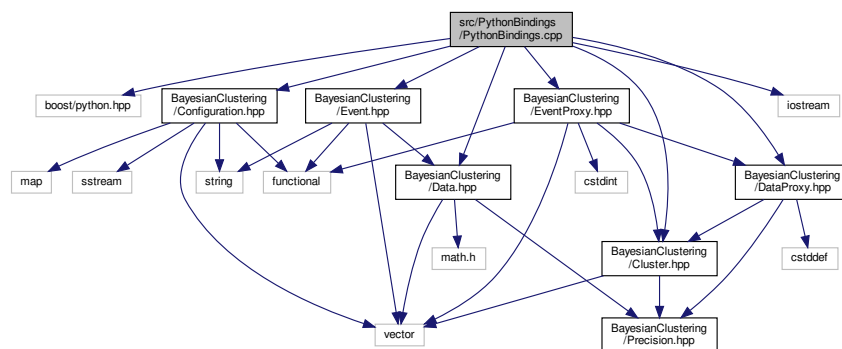
File Documentation

5.1 src/PythonBindings/PythonBindings.cpp File Reference

Self-contained sourcefile for producing python-bindings.

```
#include <boost/python.hpp>
#include "BayesianClustering/Configuration.hpp"
#include "BayesianClustering/Event.hpp"
#include "BayesianClustering/EventProxy.hpp"
#include "BayesianClustering/Cluster.hpp"
#include "BayesianClustering/Data.hpp"
#include "BayesianClustering/DataProxy.hpp"
#include <iostream>
```

Include dependency graph for PythonBindings.cpp:



Classes

- struct `PyIterator< U >`
A python iterator over a C++ container.
- struct `PyIterator< U * >`
A partial specialization creating a python iterator over a C++ container of pointers.

Functions

- `template<typename T >`
`std::vector< T > py_list_to_std_vector (const boost::python::object &alterable)`
Utility function to convert a python list to STL vector.
- `template<class T >`
`boost::python::list std_vector_to_py_list (const std::vector< T > &aVector)`
Utility function to convert an STL vector to python list.
- `void ConfigFromVector (const boost::python::object &aList)`
Set the Bayesian-clustering configuration from a vector of arguments.
- `void OneStopGetClusters (const boost::python::object &aCallback)`
Run a 1-pass clustering for a specified R & T and pass the results to a callback function.
- `PyIterator< Data * > Cluster_GetIterator (const Cluster &aCluster)`
Utility function to get a python iterator over all the data points in a clusters.
- `std::size_t Cluster_GetSize (Cluster &aCluster)`
Utility function to get the number of data points in a clusters.
- `PyIterator< Data > Event_GetIterator (const Event &aEvent)`
Utility function to get a python iterator over all the data points in an event.
- `std::size_t Event_GetSize (Event &aEvent)`
Utility function to get the number of data points in an event.
- `BOOST_PYTHON_MODULE (BayesianClustering)`
Boost Python Wrapper providing bindings for our C++ functions.

5.1.1 Detailed Description

Self-contained sourcefile for producing python-bindings.

5.1.2 Function Documentation

5.1.2.1 [Cluster_GetIterator\(\)](#)

```
PyIterator<Data*> Cluster\_GetIterator (
    const Cluster & aCluster )
```

Utility function to get a python iterator over all the data points in a clusters.

Parameters

<i>aCluster</i>	The cluster over which we are iterating
-----------------	---

Returns

An iterator object pointing to a member of the cluster

Definition at line 138 of file PythonBindings.cpp.

References Cluster::mData.

Referenced by BOOST_PYTHON_MODULE().

5.1.2.2 Cluster_GetSize()

```
std::size_t Cluster_GetSize (
    Cluster & aCluster )
```

Utility function to get the number of data points in a clusters.

Parameters

<i>aCluster</i>	The cluster we are inspecting
-----------------	-------------------------------

Returns

The number of data points in the clusters

Definition at line 143 of file PythonBindings.cpp.

References Cluster::mData.

Referenced by BOOST_PYTHON_MODULE().

5.1.2.3 ConfigFromVector()

```
void ConfigFromVector (
    const boost::python::object & aList )
```

Set the Bayesian-clustering configuration from a vector of arguments.

Parameters

<i>aList</i>	A list of strings to parse as config arguments
--------------	--

Definition at line 102 of file PythonBindings.cpp.

References Configuration::FromVector(), and Configuration::Instance.

Referenced by BOOST_PYTHON_MODULE().

5.1.2.4 Event_GetIterator()

```
PyIterator<Data> Event_GetIterator (
    const Event & aEvent )
```

Utility function to get a python iterator over all the data points in an event.

Parameters

<i>aEvent</i>	The event over which we are iterating
---------------	---------------------------------------

Returns

An iterator object pointing to a member of the event

Definition at line 151 of file PythonBindings.cpp.

References Event::mData.

Referenced by BOOST_PYTHON_MODULE().

5.1.2.5 Event_GetSize()

```
std::size_t Event_GetSize (
    Event & aEvent )
```

Utility function to get the number of data points in an event.

Parameters

<i>aEvent</i>	The event we are inspecting
---------------	-----------------------------

Returns

The number of data points in the event

Definition at line 156 of file PythonBindings.cpp.

References Event::mData.

Referenced by BOOST_PYTHON_MODULE().

5.1.2.6 OneStopGetClusters()

```
void OneStopGetClusters (
    const boost::python::object & aCallback )
```

Run a 1-pass clustering for a specified R & T and pass the results to a callback function.

Parameters

<i>aCallback</i>	A callback to which results are passed
------------------	--

Definition at line 109 of file PythonBindings.cpp.

References `Event::Clusterize()`, `Configuration::Instance`, `EventProxy::mClusters`, and `EventProxy::mData`.

Referenced by `BOOST_PYTHON_MODULE()`.

5.1.2.7 `py_list_to_std_vector()`

```
template<typename T >
std::vector< T > py_list_to_std_vector (
    const boost::python::object & aIterable ) [inline]
```

Utility function to convert a python list to STL vector.

Template Parameters

<i>T</i>	The object type in the STL container
----------	--------------------------------------

Parameters

<i>alterable</i>	The python list to convert
------------------	----------------------------

Returns

The python data in an STL vector

Definition at line 23 of file PythonBindings.cpp.

5.1.2.8 `std_vector_to_py_list()`

```
template<class T >
boost::python::list std_vector_to_py_list (
    const std::vector< T > & aVector ) [inline]
```

Utility function to convert an STL vector to python list.

Template Parameters

<i>T</i>	The object type in the STL container
----------	--------------------------------------

Parameters

<i>aVector</i>	The STL vector to convert
----------------	---------------------------

Returns

The STL data in a python list

Definition at line 33 of file PythonBindings.cpp.

Index

- alpha
 - Configuration, [14](#)
- alt_log_score
 - Cluster::Parameter, [60](#)
- CalculateLocalizationScore
 - Data, [35](#)
- CheckClusterization
 - EventProxy, [50](#)
- Cluster, [7](#)
 - Cluster, [8](#)
 - GetParent, [8](#)
 - mParams, [9](#)
 - operator+=, [9](#)
- Cluster::Parameter, [60](#)
 - alt_log_score, [60](#)
 - log_score, [61](#)
 - operator+=, [61](#)
- Cluster_GetIterator
 - PythonBindings.cpp, [70](#)
- Cluster_GetSize
 - PythonBindings.cpp, [71](#)
- Clusterize
 - DataProxy, [42](#)
 - Event, [45](#)
 - EventProxy, [51](#)
- ClusterR
 - Configuration, [14](#)
- ClusterT
 - Configuration, [15](#)
- ConfigFromVector
 - PythonBindings.cpp, [71](#)
- Configuration, [10](#)
 - alpha, [14](#)
 - ClusterR, [14](#)
 - ClusterT, [15](#)
 - dR, [15](#)
 - dT, [15](#)
 - FromCommandline, [16](#)
 - FromVector, [16](#)
 - getCentreX, [17](#)
 - getCentreY, [17](#)
 - getInstance, [17](#)
 - getWidthX, [17](#)
 - getWidthY, [18](#)
 - inputFile, [18](#)
 - log_probability_sigma, [18](#), [19](#)
 - logAlpha, [19](#)
 - logGammaAlpha, [19](#)
 - logPb, [20](#)
 - logPbDagger, [20](#)
 - max2R, [20](#)
 - max2R2, [21](#)
 - maxR, [21](#)
 - maxR2, [21](#)
 - maxScanR, [22](#)
 - maxScanT, [22](#)
 - minScanR, [22](#)
 - minScanT, [23](#)
 - mPhysicalCentreX, [32](#)
 - outputFile, [23](#)
 - probability_sigma, [23](#), [24](#)
 - Rbins, [24](#)
 - SetAlpha, [24](#)
 - SetCentre, [25](#)
 - SetInputFile, [25](#)
 - SetOutputFile, [26](#)
 - SetPb, [26](#)
 - SetRBins, [26](#)
 - SetSigmaParameters, [28](#)
 - SetTBins, [28](#)
 - SetValidate, [29](#)
 - SetWidth, [29](#)
 - sigmabins, [29](#), [30](#)
 - sigmabins2, [30](#)
 - sigmacount, [31](#)
 - sigmaspacing, [31](#)
 - Tbins, [31](#)
 - validate, [32](#)
- Data, [33](#)
 - CalculateLocalizationScore, [35](#)
 - Data, [34](#), [35](#)
 - dPhi, [36](#)
 - dR, [36](#)
 - dR2, [37](#)
 - operator<, [37](#)
 - operator=, [37](#), [38](#)
 - Preprocess, [38](#)
 - PreprocessLocalizationScores, [39](#)
- DataProxy, [39](#)
 - Clusterize, [42](#)
 - DataProxy, [41](#)
 - GetCluster, [43](#)
 - operator=, [43](#)
- Deriv
 - GSLInterpolator, [56](#)
- Deriv2
 - GSLInterpolator, [56](#)
- dPhi

- Data, 36
- dR
 - Configuration, 15
 - Data, 36
- dR2
 - Data, 37
- dT
 - Configuration, 15
- Eval
 - GSLInterpolator, 56
- Evaluate
 - GSLInterpolator, 57
- Event, 44
 - Clusterize, 45
 - Event, 45
 - LoadCSV, 46
 - operator=, 46, 47
 - ScanRT, 47
 - WriteCSV, 47
- Event_GetIterator
 - PythonBindings.cpp, 71
- Event_GetSize
 - PythonBindings.cpp, 72
- EventProxy, 48
 - CheckClusterization, 50
 - Clusterize, 51
 - EventProxy, 49, 50
 - GetData, 51
 - operator=, 52
 - ScanRT, 52
- FromCommandline
 - Configuration, 16
- FromVector
 - Configuration, 16
- getCentreX
 - Configuration, 17
- getCentreY
 - Configuration, 17
- GetCluster
 - DataProxy, 43
- GetData
 - EventProxy, 51
- getInstance
 - Configuration, 17
- GetParent
 - Cluster, 8
- getWidthX
 - Configuration, 17
- getWidthY
 - Configuration, 18
- GSLInterpolator, 53
 - Deriv, 56
 - Deriv2, 56
 - Eval, 56
 - Evaluate, 57
 - GSLInterpolator, 54, 55
 - Integ, 57
 - operator=, 58
 - SetData, 59
- inputFile
 - Configuration, 18
- Integ
 - GSLInterpolator, 57
- LoadCSV
 - Event, 46
- log_probability_sigma
 - Configuration, 18, 19
- log_score
 - Cluster::Parameter, 61
- logAlpha
 - Configuration, 19
- logGammaAlpha
 - Configuration, 19
- logPb
 - Configuration, 20
- logPbDagger
 - Configuration, 20
- max2R
 - Configuration, 20
- max2R2
 - Configuration, 21
- maxR
 - Configuration, 21
- maxR2
 - Configuration, 21
- maxScanR
 - Configuration, 22
- maxScanT
 - Configuration, 22
- minScanR
 - Configuration, 22
- minScanT
 - Configuration, 23
- mParams
 - Cluster, 9
- mPhysicalCentreX
 - Configuration, 32
- next
 - PyIterator< U >, 66
 - PyIterator< U * >, 68
- OneStopGetClusters
 - PythonBindings.cpp, 72
- operator<
 - Data, 37
- operator++
 - ProgressBar, 63
 - ProgressBar2, 64
- operator+=
 - Cluster, 9
 - Cluster::Parameter, 61

- operator=
 - Data, [37](#), [38](#)
 - DataProxy, [43](#)
 - Event, [46](#), [47](#)
 - EventProxy, [52](#)
 - GSLInterpolator, [58](#)
- outputFile
 - Configuration, [23](#)
- Preprocess
 - Data, [38](#)
- PreprocessLocalizationScores
 - Data, [39](#)
- probability_sigma
 - Configuration, [23](#), [24](#)
- ProgressBar, [62](#)
 - operator++, [63](#)
 - ProgressBar, [62](#)
- ProgressBar2, [63](#)
 - operator++, [64](#)
 - ProgressBar2, [64](#)
- py_list_to_std_vector
 - PythonBindings.cpp, [73](#)
- PyIterator
 - PyIterator< U >, [66](#)
 - PyIterator< U * >, [67](#)
- PyIterator< U >, [65](#)
 - next, [66](#)
 - PyIterator, [66](#)
- PyIterator< U * >, [66](#)
 - next, [68](#)
 - PyIterator, [67](#)
- PythonBindings.cpp
 - Cluster_GetIterator, [70](#)
 - Cluster_GetSize, [71](#)
 - ConfigFromVector, [71](#)
 - Event_GetIterator, [71](#)
 - Event_GetSize, [72](#)
 - OneStopGetClusters, [72](#)
 - py_list_to_std_vector, [73](#)
 - std_vector_to_py_list, [73](#)
- Rbins
 - Configuration, [24](#)
- ScanRT
 - Event, [47](#)
 - EventProxy, [52](#)
- SetAlpha
 - Configuration, [24](#)
- SetCentre
 - Configuration, [25](#)
- SetData
 - GSLInterpolator, [59](#)
- SetInputFile
 - Configuration, [25](#)
- SetOutputFile
 - Configuration, [26](#)
- SetPb
 - Configuration, [26](#)
- SetRBins
 - Configuration, [26](#)
- SetSigmaParameters
 - Configuration, [28](#)
- SetTBins
 - Configuration, [28](#)
- SetValidate
 - Configuration, [29](#)
- SetWidth
 - Configuration, [29](#)
- sigmabins
 - Configuration, [29](#), [30](#)
- sigmabins2
 - Configuration, [30](#)
- sigmacount
 - Configuration, [31](#)
- sigmaspacing
 - Configuration, [31](#)
- src/PythonBindings/PythonBindings.cpp, [69](#)
- std_vector_to_py_list
 - PythonBindings.cpp, [73](#)
- Tbins
 - Configuration, [31](#)
- validate
 - Configuration, [32](#)
- WriteCSV
 - Event, [47](#)