

# Bayesian Cluster Tool

Generated by Doxygen 1.8.17

Thu Jun 23 2022 10:38:24



<b>1 Class Index</b>	<b>1</b>
1.1 Class List	1
<b>2 Class Documentation</b>	<b>3</b>
2.1 Cluster Class Reference	3
2.1.1 Detailed Description	4
2.1.2 Constructor & Destructor Documentation	4
2.1.2.1 Cluster()	4
2.1.3 Member Function Documentation	4
2.1.3.1 GetParent()	4
2.1.3.2 operator+=( )	5
2.2 Configuration Class Reference	5
2.2.1 Detailed Description	9
2.2.2 Member Function Documentation	9
2.2.2.1 alpha()	9
2.2.2.2 dR()	10
2.2.2.3 dT()	10
2.2.2.4 FromCommandline()	10
2.2.2.5 inputFile()	11
2.2.2.6 log_probability_sigma() [1/2]	11
2.2.2.7 log_probability_sigma() [2/2]	11
2.2.2.8 logAlpha()	12
2.2.2.9 logGammaAlpha()	12
2.2.2.10 logPb()	13
2.2.2.11 logPbDagger()	13
2.2.2.12 max2R()	13
2.2.2.13 max2R2()	14
2.2.2.14 maxR()	14
2.2.2.15 maxR2()	14
2.2.2.16 maxScanR()	15
2.2.2.17 maxScanT()	15
2.2.2.18 minScanR()	15
2.2.2.19 minScanT()	16
2.2.2.20 outputFile()	16
2.2.2.21 probability_sigma() [1/2]	16
2.2.2.22 probability_sigma() [2/2]	16
2.2.2.23 Rbins()	17
2.2.2.24 scale2()	17
2.2.2.25 SetAlpha()	17
2.2.2.26 SetCentre()	18
2.2.2.27 SetInputFile()	18
2.2.2.28 SetOutputFile()	19

2.2.2.29 SetPb()	19
2.2.2.30 SetRBins()	19
2.2.2.31 SetSigmaParameters()	20
2.2.2.32 SetTBins()	20
2.2.2.33 SetValidate()	21
2.2.2.34 SetZoom()	21
2.2.2.35 sigmabins() [1/2]	21
2.2.2.36 sigmabins() [2/2]	22
2.2.2.37 sigmabins2() [1/2]	22
2.2.2.38 sigmabins2() [2/2]	22
2.2.2.39 sigmacount()	23
2.2.2.40 sigmaspacing()	23
2.2.2.41 Tbins()	23
2.2.2.42 toAlgorithmUnits()	23
2.2.2.43 toAlgorithmX()	24
2.2.2.44 toAlgorithmY()	24
2.2.2.45 toPhysicalUnits()	25
2.2.2.46 toPhysicalX()	25
2.2.2.47 toPhysicalY()	26
2.2.2.48 validate()	26
2.3 Data Class Reference	27
2.3.1 Detailed Description	28
2.3.2 Constructor & Destructor Documentation	28
2.3.2.1 Data() [1/3]	28
2.3.2.2 Data() [2/3]	29
2.3.2.3 Data() [3/3]	29
2.3.3 Member Function Documentation	29
2.3.3.1 dPhi()	29
2.3.3.2 dR()	30
2.3.3.3 dR2()	30
2.3.3.4 operator<()	31
2.3.3.5 operator=() [1/2]	31
2.3.3.6 operator=() [2/2]	31
2.3.3.7 Preprocess()	32
2.4 DataProxy Class Reference	32
2.4.1 Detailed Description	34
2.4.2 Constructor & Destructor Documentation	34
2.4.2.1 DataProxy() [1/3]	34
2.4.2.2 DataProxy() [2/3]	34
2.4.2.3 DataProxy() [3/3]	34
2.4.3 Member Function Documentation	35
2.4.3.1 Clusterize() [1/2]	35

2.4.3.2 Clusterize() [2/2]	35
2.4.3.3 GetCluster()	36
2.4.3.4 operator=() [1/2]	36
2.4.3.5 operator=() [2/2]	36
2.5 Event Class Reference	37
2.5.1 Detailed Description	37
2.5.2 Constructor & Destructor Documentation	38
2.5.2.1 Event() [1/2]	38
2.5.2.2 Event() [2/2]	38
2.5.3 Member Function Documentation	38
2.5.3.1 LoadCSV()	38
2.5.3.2 operator=() [1/2]	39
2.5.3.3 operator=() [2/2]	39
2.5.3.4 ScanRT()	39
2.5.3.5 WriteCSV()	40
2.6 EventProxy Class Reference	40
2.6.1 Detailed Description	41
2.6.2 Constructor & Destructor Documentation	41
2.6.2.1 EventProxy() [1/3]	41
2.6.2.2 EventProxy() [2/3]	42
2.6.2.3 EventProxy() [3/3]	42
2.6.3 Member Function Documentation	42
2.6.3.1 CheckClusterization()	42
2.6.3.2 GetData()	43
2.6.3.3 operator=() [1/2]	43
2.6.3.4 operator=() [2/2]	44
2.6.3.5 ScanRT()	44
2.7 Cluster::Parameter Struct Reference	44
2.7.1 Detailed Description	45
2.7.2 Member Function Documentation	45
2.7.2.1 log_score()	45
2.7.2.2 operator+=()	45
2.8 ProgressBar Struct Reference	46
2.8.1 Detailed Description	47
2.8.2 Constructor & Destructor Documentation	47
2.8.2.1 ProgressBar()	47
2.8.3 Member Function Documentation	47
2.8.3.1 operator++()	47
2.9 ProgressBar2 Struct Reference	48
2.9.1 Detailed Description	48
2.9.2 Constructor & Destructor Documentation	48
2.9.2.1 ProgressBar2()	48

---

2.9.3 Member Function Documentation . . . . .	49
2.9.3.1 operator++() . . . . .	49
2.10 WrappedThread Class Reference . . . . .	49
2.10.1 Detailed Description . . . . .	50
2.10.2 Constructor & Destructor Documentation . . . . .	50
2.10.2.1 WrappedThread() [1/2] . . . . .	50
2.10.2.2 WrappedThread() [2/2] . . . . .	51
2.10.3 Member Function Documentation . . . . .	51
2.10.3.1 operator=() [1/2] . . . . .	51
2.10.3.2 operator=() [2/2] . . . . .	51
2.10.3.3 run_and_wait() . . . . .	52
2.10.3.4 submit() . . . . .	52
<b>Index</b>	<b>53</b>

# Chapter 1

## Class Index

### 1.1 Class List

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

<a href="#">Cluster</a>	A class representing a cluster . . . . .	3
<a href="#">Configuration</a>	Class for storing the configuration parameters . . . . .	5
<a href="#">Data</a>	A class to store the raw data-points . . . . .	27
<a href="#">DataProxy</a>	A light-weight proxy for the raw data-points . . . . .	32
<a href="#">Event</a>	A class which holds the raw event data and global parameters . . . . .	37
<a href="#">EventProxy</a>	A lightweight wrapper for the event to store clusters for a given scan . . . . .	40
<a href="#">Cluster::Parameter</a>	A struct representing the cluster parameters . . . . .	44
<a href="#">ProgressBar</a>	A utility progress-bar . . . . .	46
<a href="#">ProgressBar2</a>	A utility code timer . . . . .	48
<a href="#">WrappedThread</a>	A class to wrap a worker-thread . . . . .	49





## Chapter 2

# Class Documentation

### 2.1 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`  
The collection of parameters, each corresponding to a different sigma hypothesis.
- `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.

### 2.1.1 Detailed Description

A class representing a cluster.

Definition at line 14 of file Cluster.hpp.

### 2.1.2 Constructor & Destructor Documentation

#### 2.1.2.1 Cluster()

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

Construct a cluster from a single data-point.

##### Parameters

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

Definition at line 61 of file Cluster.cpp.

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

### 2.1.3 Member Function Documentation

#### 2.1.3.1 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 116 of file Cluster.cpp.

References GetParent(), and mParent.

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

**2.1.3.2 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 102 of file Cluster.cpp.

References mClusterSize, and mParams.

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

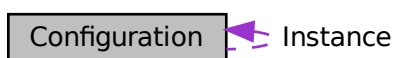
- include/BayesianClustering/Cluster.hpp
- src/BayesianClustering/Cluster.cpp

**2.2 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 [SetZoom](#) (const double &aScale)  
*Setter for the half-width of the scan 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.*
- const double & [scale2](#) () const  
*Getter for the scale-parameter squared.*
- const std::size\_t & [sigmacount](#) () const  
*Getter for the sigma count.*
- 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.*
- double [toPhysicalUnits](#) (const double &aAlgorithmUnits) const

*Utility function to convert a normalized algorithm distance to physical distance.*
- double [toAlgorithmUnits](#) (const double &aPhysicalUnits) const

*Utility function to convert physical distances to a normalized algorithm distances.*
- double [toPhysicalX](#) (const double &aAlgorithmX) const

*Utility function to convert a normalized algorithm x-coordinate to a physical x-coordinate.*
- double [toAlgorithmX](#) (const double &aPhysicalX) const

*Utility function to convert a physical x-coordinate to a normalized algorithm x-coordinate.*
- double [toPhysicalY](#) (const double &aAlgorithmY) const

*Utility function to convert a normalized algorithm y-coordinate to a physical y-coordinate.*
- double [toAlgorithmY](#) (const double &aPhysicalY) const

*Utility function to convert a physical y-coordinate to a normalized algorithm y-coordinate.*

## Static Public Attributes

- static [Configuration Instance](#)  
*A single global copy of the global variables.*

## Private Attributes

- double [mScale](#)  
*The scale parameter.*
- double [mScale2](#)  
*The scale parameter squared.*
- double [mPhysicalCentreX](#)  
*The x-coordinate of the centre of the window in physical units.*
- double [mPhysicalCentreY](#)  
*The y-coordinate of the centre of the window 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.*

### 2.2.1 Detailed Description

Class for storing the configuration parameters.

Definition at line 71 of file Configuration.hpp.

### 2.2.2 Member Function Documentation

#### 2.2.2.1 alpha()

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

Getter for the alpha parameter

##### Returns

The alpha parameter

Definition at line 221 of file Configuration.hpp.

References `mAlpha`.

#### 2.2.2.2 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 201 of file Configuration.hpp.

References mDR.

Referenced by Data::Preprocess().

#### 2.2.2.3 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 207 of file Configuration.hpp.

References mDT.

Referenced by EventProxy::ScanRT().

#### 2.2.2.4 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 147 of file Configuration.cpp.

References SetAlpha(), SetCentre(), SetInputFile(), SetOutputFile(), SetPb(), SetRBins(), SetSigmaParameters(), SetTBins(), SetValidate(), and SetZoom().

### 2.2.2.5 inputFile()

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

Getter for the input file.

#### Returns

The name of the input event file

Definition at line 236 of file Configuration.hpp.

References mInputFile.

Referenced by Event::Event().

### 2.2.2.6 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 154 of file Configuration.hpp.

References mLogProbabilitySigma.

### 2.2.2.7 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 171 of file Configuration.hpp.

References mLogProbabilitySigma.

**2.2.2.8 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 224 of file Configuration.hpp.

References mLogAlpha.

**2.2.2.9 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 227 of file Configuration.hpp.

References mLogGammaAlpha.

### 2.2.2.10 logPb()

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

Logarithm of the  $P_b$  parameter

#### Returns

Logarithm of the  $P_b$  parameter

Definition at line 214 of file Configuration.hpp.

References mLogPb.

Referenced by EventProxy::UpdateLogScore().

### 2.2.2.11 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 217 of file Configuration.hpp.

References mLogPbDagger.

### 2.2.2.12 max2R()

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

Getter for the maximum value of 2R.

#### Returns

The maximum value of 2R

Definition at line 181 of file Configuration.hpp.

References mMax2R.

Referenced by Data::Preprocess().

#### 2.2.2.13 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 184 of file Configuration.hpp.

References mMax2R2.

#### 2.2.2.14 maxR()

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

Getter for the maximum value of R.

##### Returns

The maximum value of R

Definition at line 175 of file Configuration.hpp.

References mMaxR.

#### 2.2.2.15 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 178 of file Configuration.hpp.

References mMaxR2.

### 2.2.2.16 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 191 of file Configuration.hpp.

References mMaxScanR.

### 2.2.2.17 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 197 of file Configuration.hpp.

References mMaxScanT.

Referenced by EventProxy::ScanRT().

### 2.2.2.18 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 188 of file Configuration.hpp.

References mMinScanR.

### 2.2.2.19 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 194 of file Configuration.hpp.

References mMinScanT.

### 2.2.2.20 outputFile()

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

Getter for the output file.

#### Returns

The name of the output file

Definition at line 239 of file Configuration.hpp.

References mOutputFile.

### 2.2.2.21 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 151 of file Configuration.hpp.

References mProbabilitySigma.

### 2.2.2.22 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 167 of file Configuration.hpp.

References mProbabilitySigma.

#### 2.2.2.23 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 204 of file Configuration.hpp.

References mRbins.

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

#### 2.2.2.24 scale2()

```
const double& Configuration::scale2 ( ) const [inline]
```

Getter for the scale-parameter squared.

### Returns

The scale-parameter squared

Definition at line 133 of file Configuration.hpp.

References mScale2.

#### 2.2.2.25 SetAlpha()

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

Setter for the alpha parameter.

**Parameters**

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

Definition at line 100 of file Configuration.cpp.

References mAlpha, mLogAlpha, and mLogGammaAlpha.

Referenced by FromCommandline().

**2.2.2.26 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 36 of file Configuration.cpp.

References mPhysicalCentreX, and mPhysicalCentreY.

Referenced by FromCommandline().

**2.2.2.27 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 116 of file Configuration.cpp.

References mInputFile.

Referenced by FromCommandline().



**2.2.2.28 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 123 of file Configuration.cpp.

References mOutputFile.

Referenced by FromCommandline().

**2.2.2.29 SetPb()**

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

Setter for the P\_b parameter.

**Parameters**

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

Definition at line 93 of file Configuration.cpp.

References mLogPb, and mLogPbDagger.

Referenced by FromCommandline().

**2.2.2.30 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 68 of file Configuration.cpp.

References mDR, mMax2R, mMax2R2, mMaxR, mMaxR2, mMaxScanR, mMinScanR, mRbins, toAlgorithmUnits(), and toPhysicalUnits().

Referenced by FromCommandline().

### 2.2.2.31 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 50 of file Configuration.cpp.

References mLogProbabilitySigma, mProbabilitySigma, mScale, mSigabins, mSigabins2, mSigmaCount, mSigaspacing, and toAlgorithmUnits().

Referenced by FromCommandline().

### 2.2.2.32 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 83 of file Configuration.cpp.

References mDT, mMaxScanT, mMinScanT, mTbins, toAlgorithmUnits(), and toPhysicalUnits().

Referenced by FromCommandline().

### 2.2.2.33 SetValidate()

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

Set whether to validate clusterization.

#### Parameters

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

Definition at line 108 of file Configuration.cpp.

References mValidate.

Referenced by FromCommandline().

### 2.2.2.34 SetZoom()

```
void Configuration::SetZoom (
    const double & aScale )
```

Setter for the half-width of the scan window.

#### Parameters

<i>aScale</i>	The scale of the window in physical units (becomes $\pm 1$ in algorithm units)
---------------	--

Definition at line 43 of file Configuration.cpp.

References mScale, and mScale2.

Referenced by FromCommandline().

### 2.2.2.35 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 145 of file Configuration.hpp.

References mSigmabins.

### 2.2.2.36 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 159 of file Configuration.hpp.

References mSigmabins.

### 2.2.2.37 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 148 of file Configuration.hpp.

References mSigmabins2.

### 2.2.2.38 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 163 of file `Configuration.hpp`.

References `mSigabins2`.

**2.2.2.39 sigmacount()**

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

Getter for the sigma count.

**Returns**

The sigma count

Definition at line 137 of file `Configuration.hpp`.

References `mSigmacount`.

Referenced by `Cluster::UpdateLogScore()`.

**2.2.2.40 sigmaspacing()**

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

Getter for the sigma spacing.

**Returns**

The sigma spacing

Definition at line 141 of file `Configuration.hpp`.

References `mSigmaspacing`.

**2.2.2.41 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 210 of file `Configuration.hpp`.

References `mTbins`.

Referenced by `EventProxy::ScanRT()`.

**2.2.2.42 toAlgorithmUnits()**

```
double Configuration::toAlgorithmUnits (
    const double & aPhysicalUnits ) const [inline]
```

Utility function to convert physical distances to a normalized algorithm distances.

**Parameters**

<i>aPhysicalUnits</i>	A physical distance
-----------------------	---------------------

**Returns**

A normalized algorithm distances

Definition at line 253 of file Configuration.hpp.

References mScale.

Referenced by SetRBins(), SetSigmaParameters(), SetTBins(), toAlgorithmX(), and toAlgorithmY().

**2.2.2.43 toAlgorithmX()**

```
double Configuration::toAlgorithmX (
    const double & aPhysicalX ) const [inline]
```

Utility function to convert a physical x-coordinate to a normalized algorithm x-coordinate.

**Parameters**

<i>aPhysicalX</i>	A physical x-coordinate
-------------------	-------------------------

**Returns**

A normalized x-coordinate

Definition at line 269 of file Configuration.hpp.

References mPhysicalCentreX, and toAlgorithmUnits().

**2.2.2.44 toAlgorithmY()**

```
double Configuration::toAlgorithmY (
    const double & aPhysicalY ) const [inline]
```

Utility function to convert a physical y-coordinate to a normalized algorithm y-coordinate.

**Parameters**

<i>aPhysicalY</i>	A physical y-coordinate
-------------------	-------------------------

**Returns**

A normalized y-coordinate

Definition at line 285 of file Configuration.hpp.

References `mPhysicalCentreY`, and `toAlgorithmUnits()`.

**2.2.2.45 toPhysicalUnits()**

```
double Configuration::toPhysicalUnits (
    const double & aAlgorithmUnits ) const [inline]
```

Utility function to convert a normalized algorithm distance to physical distance.

**Parameters**

<i>aAlgorithmUnits</i>	A normalized algorithm distance
------------------------	---------------------------------

**Returns**

A physical distances

Definition at line 245 of file Configuration.hpp.

References `mScale`.

Referenced by `SetRBins()`, `SetTBins()`, `toPhysicalX()`, `toPhysicalY()`, and `Event::WriteCSV()`.

**2.2.2.46 toPhysicalX()**

```
double Configuration::toPhysicalX (
    const double & aAlgorithmX ) const [inline]
```

Utility function to convert a normalized algorithm x-coordinate to a physical x-coordinate.

**Parameters**

<i>aAlgorithmX</i>	A normalized x-coordinate
--------------------	---------------------------

**Returns**

A physical x-coordinate

Definition at line 261 of file Configuration.hpp.

References mPhysicalCentreX, and toPhysicalUnits().

#### 2.2.2.47 toPhysicalY()

```
double Configuration::toPhysicalY (
    const double & aAlgorithmY ) const [inline]
```

Utility function to convert a normalized algorithm y-coordinate to a physical y-coordinate.

##### Parameters

<i>aAlgorithmY</i>	A normalized y-coordinate
--------------------	---------------------------

##### Returns

A physical y-coordinate

Definition at line 277 of file Configuration.hpp.

References mPhysicalCentreY, and toPhysicalUnits().

Referenced by Event::WriteCSV().

#### 2.2.2.48 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 231 of file Configuration.hpp.

References mValidate.

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

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

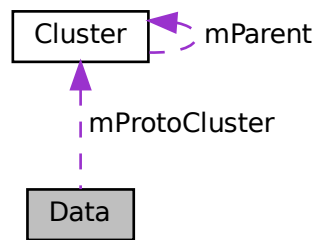


## 2.3 Data Class Reference

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.*

## 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.*

### 2.3.1 Detailed Description

A class to store the raw data-points.

Definition at line 14 of file Data.hpp.

### 2.3.2 Constructor & Destructor Documentation

#### 2.3.2.1 Data() [1/3]

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

Constructor.

#### Parameters

<a href="#">aX</a>	The x-position of the data-point in algorithm units
<a href="#">aY</a>	The y-position of the data-point in algorithm units
<a href="#">aS</a>	The sigma of the data-point in algorithm units

Definition at line 13 of file Data.cpp.

### 2.3.2.2 Data() [2/3]

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

Deleted copy constructor.

#### Parameters

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

### 2.3.2.3 Data() [3/3]

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

Default move constructor.

#### Parameters

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

## 2.3.3 Member Function Documentation

### 2.3.3.1 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`.

### 2.3.3.2 `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()`.

### 2.3.3.3 `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()`.

#### 2.3.3.4 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.

#### 2.3.3.5 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
---------------	--------------------

#### 2.3.3.6 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
---------------	--------------------

**2.3.3.7 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::dR(), Configuration::Instance, Configuration::max2R(), and Configuration::Rbins().

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

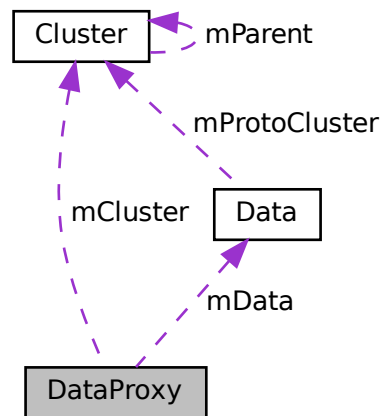
- include/BayesianClustering/Data.hpp
- src/BayesianClustering/Data.cpp

**2.4 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.*

### 2.4.1 Detailed Description

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

Definition at line 17 of file DataProxy.hpp.

### 2.4.2 Constructor & Destructor Documentation

#### 2.4.2.1 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.

#### 2.4.2.2 DataProxy() [2/3]

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

Deleted copy constructor.

Parameters

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

#### 2.4.2.3 DataProxy() [3/3]

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

Default move constructor.



## Parameters

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

## 2.4.3 Member Function Documentation

### 2.4.3.1 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().

### 2.4.3.2 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.

### 2.4.3.3 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().

### 2.4.3.4 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
---------------	--------------------

### 2.4.3.5 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

## 2.5 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 &) > &aCallback)  
*Run the scan.*
- 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.*

#### 2.5.1 Detailed Description

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

Definition at line 16 of file Event.hpp.

## 2.5.2 Constructor & Destructor Documentation

### 2.5.2.1 Event() [1/2]

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

Deleted copy constructor.

#### Parameters

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

### 2.5.2.2 Event() [2/2]

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

Default move constructor.

#### Parameters

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

## 2.5.3 Member Function Documentation

### 2.5.3.1 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 90 of file Event.cpp.

References mData.

Referenced by Event().

### 2.5.3.2 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
---------------	--------------------

### 2.5.3.3 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
---------------	--------------------

### 2.5.3.4 ScanRT()

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

Run the scan.

**Parameters**

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

Definition at line 33 of file Event.cpp.

References Preprocess().

**2.5.3.5 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 121 of file Event.cpp.

References Configuration::Instance, mData, Configuration::toPhysicalUnits(), and Configuration::toPhysicalY().

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

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

**2.6 EventProxy Class Reference**

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

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

**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 &) > &aCallback, const uint8\_t &aParallelization=1, const uint8\_t &aOffset=0)  
*Run an RT-scan.*
- void [UpdateLogScore](#) ()  
*Update log-probability after a scan.*
- [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.*

### 2.6.1 Detailed Description

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

Definition at line 15 of file EventProxy.hpp.

### 2.6.2 Constructor & Destructor Documentation

#### 2.6.2.1 EventProxy() [1/3]

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

Default constructor.

#### Parameters

<a href="#">aEvent</a>	An event for which this is a lightweight proxy
------------------------	--

Definition at line 14 of file EventProxy.cpp.

References mClusters, Event::mData, and mData.

### 2.6.2.2 EventProxy() [2/3]

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

Deleted copy constructor.

#### Parameters

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

### 2.6.2.3 EventProxy() [3/3]

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

Default move constructor.

#### Parameters

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

## 2.6.3 Member Function Documentation

### 2.6.3.1 CheckClusterization()

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

Run validation tests on the clusters.

#### Parameters

<i>R</i>	The R of the last run scan
<i>T</i>	The T of the last run scan



Definition at line 22 of file EventProxy.cpp.

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

### 2.6.3.2 GetData()

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

Get the proxy for the Nth neighbour of this data-point.

#### Returns

A reference to the neighbour data-proxy

#### Parameters

<i>aIndex</i>	The index of the neighbour we are looking for
---------------	---

Definition at line 53 of file EventProxy.hpp.

References mData.

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

### 2.6.3.3 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
---------------	--------------------

### 2.6.3.4 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
---------------	--------------------

### 2.6.3.5 ScanRT()

```
void EventProxy::ScanRT (
    const std::function< void(const EventProxy &, const double &, const double &) > &
    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 94 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

## 2.7 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.*

## 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.*

### 2.7.1 Detailed Description

A struct representing the cluster parameters.

Definition at line 19 of file Cluster.hpp.

### 2.7.2 Member Function Documentation

#### 2.7.2.1 [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 35 of file Cluster.cpp.

#### 2.7.2.2 [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 16 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

## 2.8 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.*

## 2.8.1 Detailed Description

A utility progress-bar.

Definition at line 6 of file ProgressBar.hpp.

## 2.8.2 Constructor & Destructor Documentation

### 2.8.2.1 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.

## 2.8.3 Member Function Documentation

### 2.8.3.1 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

## 2.9 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.*

### 2.9.1 Detailed Description

A utility code timer.

Definition at line 34 of file ProgressBar.hpp.

### 2.9.2 Constructor & Destructor Documentation

#### 2.9.2.1 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.

## 2.9.3 Member Function Documentation

### 2.9.3.1 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

## 2.10 WrappedThread Class Reference

A class to wrap a worker-thread.

```
#include <Vectorize.hpp>
```

### Public Member Functions

- [WrappedThread](#) ()  
*Default constructor.*
- [WrappedThread](#) (const [WrappedThread](#) &aOther)=delete  
*Deleted copy constructor.*
- [WrappedThread](#) & [operator=](#) (const [WrappedThread](#) &aOther)=delete  
*Deleted assignment operator.*
- [WrappedThread](#) ([WrappedThread](#) &&aOther)=default  
*Default move constructor.*
- [WrappedThread](#) & [operator=](#) ([WrappedThread](#) &&aOther)=default  
*Default move-assignment constructor.*
- virtual [~WrappedThread](#) ()  
*Destructor.*
- void [submit](#) (const std::function< void() > &aFunc)  
*Submit a job to this thread.*

## Static Public Member Functions

- static void [run\\_and\\_wait](#) (const std::function< void() > &aFunc)  
*Submit a job to the current thread and then wait for all other threads to finish.*
- static void [wait](#) ()  
*Wait for all other threads to finish*

## Private Member Functions

- void [Runner](#) ()  
*The function run by the raw thread.*

## Private Attributes

- const std::uint64\_t [mMask](#)  
*A mask indicating which thread we are on.*
- std::condition\_variable [mConditionVariable](#)  
*A condition variable for talking across threads.*
- std::function< void() > [mFunc](#)  
*The function call to be run by the thread.*
- std::atomic< bool > [mTerminate](#)  
*An atomic flag indicating termination.*
- std::mutex [mMutex](#)  
*The access mutex.*
- std::thread [mThread](#)  
*The raw thread.*

## Static Private Attributes

- static std::atomic< std::uint64\_t > [sBusy](#)  
*An atomic static register keeping track of which threads are busy.*
- static std::uint64\_t [sInstanceCounter](#)  
*A static counter to tell us how many threads are available.*

### 2.10.1 Detailed Description

A class to wrap a worker-thread.

Definition at line 13 of file Vectorize.hpp.

### 2.10.2 Constructor & Destructor Documentation

#### 2.10.2.1 [WrappedThread\(\)](#) [1/2]

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

Deleted copy constructor.



## Parameters

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

**2.10.2.2 WrappedThread()** [2/2]

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

Default move constructor.

## Parameters

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

**2.10.3 Member Function Documentation****2.10.3.1 operator=()** [1/2]

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

Deleted assignment operator.

## Returns

Reference to this, for chaining calls

## Parameters

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

**2.10.3.2 operator=()** [2/2]

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

Default move-assignment constructor.

## Returns

Reference to this, for chaining calls

## Parameters

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

**2.10.3.3 run\_and\_wait()**

```
void WrappedThread::run_and_wait (
    const std::function< void() > & aFunc ) [static]
```

Submit a job to the current thread and then wait for all other threads to finish.

## Parameters

<i>aFunc</i>	The job to run
--------------	----------------

Definition at line 31 of file Vectorize.cpp.

References `wait()`.

**2.10.3.4 submit()**

```
void WrappedThread::submit (
    const std::function< void() > & aFunc )
```

Submit a job to this thread.

## Parameters

<i>aFunc</i>	The job to run
--------------	----------------

Definition at line 18 of file Vectorize.cpp.

References `mConditionVariable`, `mFunc`, `mMask`, `mMutex`, and `sBusy`.

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

- `include/Utilities/Vectorize.hpp`
- `src/Utilities/Vectorize.cpp`

# Index

- alpha
  - Configuration, 9
- CheckClusterization
  - EventProxy, 42
- Cluster, 3
  - Cluster, 4
  - GetParent, 4
  - operator+=, 5
- Cluster::Parameter, 44
  - log\_score, 45
  - operator+=, 45
- Clusterize
  - DataProxy, 35
- Configuration, 5
  - alpha, 9
  - dR, 9
  - dT, 10
  - FromCommandline, 10
  - inputFile, 11
  - log\_probability\_sigma, 11
  - logAlpha, 12
  - logGammaAlpha, 12
  - logPb, 12
  - logPbDagger, 13
  - max2R, 13
  - max2R2, 13
  - maxR, 14
  - maxR2, 14
  - maxScanR, 14
  - maxScanT, 15
  - minScanR, 15
  - minScanT, 15
  - outputFile, 16
  - probability\_sigma, 16
  - Rbins, 17
  - scale2, 17
  - SetAlpha, 17
  - SetCentre, 18
  - SetInputFile, 18
  - SetOutputFile, 18
  - SetPb, 19
  - SetRBins, 19
  - SetSigmaParameters, 20
  - SetTBins, 20
  - SetValidate, 21
  - SetZoom, 21
  - sigmabins, 21
  - sigmabins2, 22
  - sigmacount, 23
  - sigmaspacing, 23
  - Tbins, 23
  - toAlgorithmUnits, 23
  - toAlgorithmX, 24
  - toAlgorithmY, 24
  - toPhysicalUnits, 25
  - toPhysicalX, 25
  - toPhysicalY, 26
  - validate, 26
- Data, 27
  - Data, 28, 29
  - dPhi, 29
  - dR, 30
  - dR2, 30
  - operator<, 30
  - operator=, 31
  - Preprocess, 32
- DataProxy, 32
  - Clusterize, 35
  - DataProxy, 34
  - GetCluster, 36
  - operator=, 36
- dPhi
  - Data, 29
- dR
  - Configuration, 9
  - Data, 30
- dR2
  - Data, 30
- dT
  - Configuration, 10
- Event, 37
  - Event, 38
  - LoadCSV, 38
  - operator=, 39
  - ScanRT, 39
  - WriteCSV, 40
- EventProxy, 40
  - CheckClusterization, 42
  - EventProxy, 41, 42
  - GetData, 43
  - operator=, 43
  - ScanRT, 44
- FromCommandline
  - Configuration, 10
- GetCluster

- DataProxy, 36
- GetData
  - EventProxy, 43
- GetParent
  - Cluster, 4
- inputFile
  - Configuration, 11
- LoadCSV
  - Event, 38
- log\_probability\_sigma
  - Configuration, 11
- log\_score
  - Cluster::Parameter, 45
- logAlpha
  - Configuration, 12
- logGammaAlpha
  - Configuration, 12
- logPb
  - Configuration, 12
- logPbDagger
  - Configuration, 13
- max2R
  - Configuration, 13
- max2R2
  - Configuration, 13
- maxR
  - Configuration, 14
- maxR2
  - Configuration, 14
- maxScanR
  - Configuration, 14
- maxScanT
  - Configuration, 15
- minScanR
  - Configuration, 15
- minScanT
  - Configuration, 15
- operator<
  - Data, 30
- operator++
  - ProgressBar, 47
  - ProgressBar2, 49
- operator+=
  - Cluster, 5
  - Cluster::Parameter, 45
- operator=
  - Data, 31
  - DataProxy, 36
  - Event, 39
  - EventProxy, 43
  - WrappedThread, 51
- outputFile
  - Configuration, 16
- Preprocess
  - Data, 32
  - probability\_sigma
    - Configuration, 16
  - ProgressBar, 46
    - operator++, 47
    - ProgressBar, 47
  - ProgressBar2, 48
    - operator++, 49
    - ProgressBar2, 48
  - Rbins
    - Configuration, 17
  - run\_and\_wait
    - WrappedThread, 52
  - scale2
    - Configuration, 17
  - ScanRT
    - Event, 39
    - EventProxy, 44
  - SetAlpha
    - Configuration, 17
  - SetCentre
    - Configuration, 18
  - SetInputFile
    - Configuration, 18
  - SetOutputFile
    - Configuration, 18
  - SetPb
    - Configuration, 19
  - SetRBins
    - Configuration, 19
  - SetSigmaParameters
    - Configuration, 20
  - SetTBins
    - Configuration, 20
  - SetValidate
    - Configuration, 21
  - SetZoom
    - Configuration, 21
  - sigmabins
    - Configuration, 21
  - sigmabins2
    - Configuration, 22
  - sigmacount
    - Configuration, 23
  - sigmaspacing
    - Configuration, 23
  - submit
    - WrappedThread, 52
  - Tbins
    - Configuration, 23
  - toAlgorithmUnits
    - Configuration, 23
  - toAlgorithmX
    - Configuration, 24
  - toAlgorithmY
    - Configuration, 24

- toPhysicalUnits
  - Configuration, [25](#)
- toPhysicalX
  - Configuration, [25](#)
- toPhysicalY
  - Configuration, [26](#)
- validate
  - Configuration, [26](#)
- WrappedThread, [49](#)
  - operator=, [51](#)
  - run\_and\_wait, [52](#)
  - submit, [52](#)
  - WrappedThread, [50](#), [51](#)
- WriteCSV
  - Event, [40](#)