

# Bayesian Cluster Tool

Generated by Doxygen 1.9.1

Wed Jun 7 2023 18:05:59



<b>1 Todo List</b>	<b>1</b>
<b>2 Class Index</b>	<b>3</b>
2.1 Class List	3
<b>3 File Index</b>	<b>5</b>
3.1 File List	5
<b>4 Class Documentation</b>	<b>7</b>
4.1 AuxConfiguration Class Reference	7
4.1.1 Detailed Description	8
4.1.2 Constructor & Destructor Documentation	8
4.1.2.1 AuxConfiguration() [1/2]	8
4.1.2.2 AuxConfiguration() [2/2]	9
4.1.3 Member Function Documentation	9
4.1.3.1 ClusterR()	9
4.1.3.2 ClusterT()	9
4.1.3.3 configFile()	10
4.1.3.4 FromVector()	10
4.1.3.5 inputFile()	10
4.1.3.6 outputFile()	11
4.1.3.7 SetConfigFile()	11
4.1.3.8 SetInputFile()	11
4.1.3.9 SetOutputFile()	12
4.1.3.10 SetValidate()	12
4.1.3.11 validate()	12
4.2 Cluster Class Reference	13
4.2.1 Detailed Description	14
4.2.2 Constructor & Destructor Documentation	14
4.2.2.1 Cluster() [1/4]	14
4.2.2.2 Cluster() [2/4]	14
4.2.2.3 Cluster() [3/4]	15
4.2.2.4 Cluster() [4/4]	15
4.2.3 Member Function Documentation	15
4.2.3.1 GetParent()	15
4.2.3.2 operator+=(())	16
4.2.3.3 operator=() [1/2]	16
4.2.3.4 operator=() [2/2]	17
4.2.3.5 UpdateLogScore()	17
4.3 Data Class Reference	17
4.3.1 Detailed Description	19
4.3.2 Constructor & Destructor Documentation	19
4.3.2.1 Data() [1/3]	19

4.3.2.2 Data() [2/3]	20
4.3.2.3 Data() [3/3]	20
4.3.3 Member Function Documentation	20
4.3.3.1 CalculateLocalizationScore()	20
4.3.3.2 dPhi()	21
4.3.3.3 dR()	21
4.3.3.4 dR2()	22
4.3.3.5 operator<()	22
4.3.3.6 operator=() [1/2]	22
4.3.3.7 operator=() [2/2]	23
4.3.3.8 Preprocess()	23
4.3.3.9 PreprocessLocalizationScores()	24
4.4 DataProxy Class Reference	24
4.4.1 Detailed Description	25
4.4.2 Constructor & Destructor Documentation	25
4.4.2.1 DataProxy() [1/3]	25
4.4.2.2 DataProxy() [2/3]	26
4.4.2.3 DataProxy() [3/3]	26
4.4.3 Member Function Documentation	26
4.4.3.1 Clusterize() [1/2]	26
4.4.3.2 Clusterize() [2/2]	27
4.4.3.3 GetCluster()	27
4.4.3.4 operator=() [1/2]	27
4.4.3.5 operator=() [2/2]	28
4.5 GSLInterpolator Class Reference	28
4.5.1 Detailed Description	29
4.5.2 Constructor & Destructor Documentation	29
4.5.2.1 GSLInterpolator() [1/4]	29
4.5.2.2 GSLInterpolator() [2/4]	30
4.5.2.3 GSLInterpolator() [3/4]	30
4.5.2.4 GSLInterpolator() [4/4]	30
4.5.3 Member Function Documentation	31
4.5.3.1 Deriv()	31
4.5.3.2 Deriv2()	31
4.5.3.3 Eval()	32
4.5.3.4 Evaluate()	32
4.5.3.5 Integ()	33
4.5.3.6 operator=() [1/2]	33
4.5.3.7 operator=() [2/2]	33
4.5.3.8 SetData() [1/2]	34
4.5.3.9 SetData() [2/2]	34
4.6 LocalizationFile Class Reference	35

4.6.1 Detailed Description . . . . .	35
4.6.2 Constructor & Destructor Documentation . . . . .	35
4.6.2.1 LocalizationFile() [1/3] . . . . .	35
4.6.2.2 LocalizationFile() [2/3] . . . . .	36
4.6.2.3 LocalizationFile() [3/3] . . . . .	36
4.6.3 Member Function Documentation . . . . .	36
4.6.3.1 ExtractRols() [1/2] . . . . .	36
4.6.3.2 ExtractRols() [2/2] . . . . .	37
4.6.3.3 operator=() [1/2] . . . . .	37
4.6.3.4 operator=() [2/2] . . . . .	38
4.7 ManualRol Struct Reference . . . . .	38
4.7.1 Detailed Description . . . . .	38
4.8 Cluster::Parameter Struct Reference . . . . .	39
4.8.1 Detailed Description . . . . .	39
4.8.2 Member Function Documentation . . . . .	39
4.8.2.1 alt_log_score() . . . . .	40
4.8.2.2 log_score() . . . . .	40
4.8.2.3 operator+=() . . . . .	40
4.9 ProgressBar Struct Reference . . . . .	41
4.9.1 Detailed Description . . . . .	41
4.9.2 Constructor & Destructor Documentation . . . . .	41
4.9.2.1 ProgressBar() . . . . .	41
4.9.3 Member Function Documentation . . . . .	42
4.9.3.1 operator++() . . . . .	42
4.10 ProgressBar2 Struct Reference . . . . .	42
4.10.1 Detailed Description . . . . .	43
4.10.2 Constructor & Destructor Documentation . . . . .	43
4.10.2.1 ProgressBar2() . . . . .	43
4.10.3 Member Function Documentation . . . . .	43
4.10.3.1 operator++() . . . . .	43
4.11 Rol Class Reference . . . . .	44
4.11.1 Detailed Description . . . . .	45
4.11.2 Constructor & Destructor Documentation . . . . .	45
4.11.2.1 Rol() [1/3] . . . . .	45
4.11.2.2 Rol() [2/3] . . . . .	46
4.11.2.3 Rol() [3/3] . . . . .	46
4.11.3 Member Function Documentation . . . . .	46
4.11.3.1 Clusterize() . . . . .	46
4.11.3.2 getArea() . . . . .	47
4.11.3.3 getCentreX() . . . . .	47
4.11.3.4 getCentreY() . . . . .	47
4.11.3.5 getWidthX() . . . . .	48

4.11.3.6 getWidthY()	48
4.11.3.7 operator=() [1/2]	48
4.11.3.8 operator=() [2/2]	49
4.11.3.9 Preprocess()	49
4.11.3.10 ScanRT()	49
4.11.3.11 SetCentre()	50
4.11.3.12 SetWidth()	50
4.12 Rolproxy Class Reference	51
4.12.1 Detailed Description	52
4.12.2 Constructor & Destructor Documentation	52
4.12.2.1 Rolproxy() [1/3]	52
4.12.2.2 Rolproxy() [2/3]	52
4.12.2.3 Rolproxy() [3/3]	53
4.12.3 Member Function Documentation	53
4.12.3.1 CheckClusterization()	53
4.12.3.2 Clusterize()	53
4.12.3.3 GetData()	54
4.12.3.4 operator=() [1/2]	54
4.12.3.5 operator=() [2/2]	55
4.12.3.6 ScanRT()	55
4.12.3.7 UpdateLogScore()	56
4.12.3.8 ValidateLogScore()	56
4.13 ScanConfiguration Class Reference	56
4.13.1 Detailed Description	59
4.13.2 Constructor & Destructor Documentation	59
4.13.2.1 ScanConfiguration() [1/3]	59
4.13.2.2 ScanConfiguration() [2/3]	59
4.13.2.3 ScanConfiguration() [3/3]	60
4.13.3 Member Function Documentation	60
4.13.3.1 alpha()	60
4.13.3.2 FromVector()	60
4.13.3.3 log_probability_sigma() [1/2]	61
4.13.3.4 log_probability_sigma() [2/2]	61
4.13.3.5 logAlpha()	62
4.13.3.6 logGammaAlpha()	62
4.13.3.7 logPb()	62
4.13.3.8 logPbDagger()	63
4.13.3.9 probability_sigma() [1/2]	63
4.13.3.10 probability_sigma() [2/2]	63
4.13.3.11 Rbounds()	64
4.13.3.12 SetAlpha()	64
4.13.3.13 SetPb()	64

4.13.3.14 SetRBins()	65
4.13.3.15 SetSigmaParameters()	65
4.13.3.16 SetTBins()	66
4.13.3.17 sigmabins() [1/2]	66
4.13.3.18 sigmabins() [2/2]	66
4.13.3.19 sigmabins2() [1/2]	67
4.13.3.20 sigmabins2() [2/2]	67
4.13.3.21 sigmacount()	68
4.13.3.22 sigmaspacing()	68
4.13.3.23 Tbounds()	68
4.14 ScanEntry Struct Reference	69
4.14.1 Detailed Description	69
4.14.2 Member Function Documentation	69
4.14.2.1 operator<()	69
4.15 ScanConfiguration::tBounds Struct Reference	70
4.15.1 Detailed Description	70
<b>5 File Documentation</b>	<b>71</b>
5.1 include/BayesianClustering/API.hpp File Reference	71
5.1.1 Function Documentation	72
5.1.1.1 AutoRoi_Cluster_Callback()	72
5.1.1.2 AutoRoi_Scan_FullCallback()	73
5.1.1.3 AutoRoi_Scan_SimpleCallback()	73
5.1.1.4 AutoRoi_Scan_ToJson()	74
5.1.1.5 ManualRoi_Cluster_Callback()	74
5.1.1.6 ManualRoi_Scan_FullCallback()	75
5.1.1.7 ManualRoi_Scan_SimpleCallback()	75
5.1.1.8 ManualRoi_Scan_ToJson()	76
5.2 include/BayesianClustering/Cluster.hpp File Reference	76
5.3 include/BayesianClustering/Configuration.hpp File Reference	77
5.4 include/BayesianClustering/Data.hpp File Reference	78
5.5 include/BayesianClustering/DataProxy.hpp File Reference	79
5.6 include/BayesianClustering/LocalizationFile.hpp File Reference	80
5.7 include/BayesianClustering/Precision.hpp File Reference	81
5.8 include/BayesianClustering/Roi.hpp File Reference	81
5.9 include/BayesianClustering/Roiproxy.hpp File Reference	82
5.10 include/Utilities/GSLInterpolator.hpp File Reference	83
5.11 include/Utilities/ListComprehension.hpp File Reference	84
5.11.1 Function Documentation	85
5.11.1.1 operator"   () [1/3]	85
5.11.1.2 operator"   () [2/3]	86
5.11.1.3 operator"   () [3/3]	86

5.11.1.4 range()	87
5.12 include/Utilities/ProgressBar.hpp File Reference	87
5.13 include/Utilities/Units.hpp File Reference	88
5.13.1 Function Documentation	89
5.13.1.1 operator""_micrometer() [1/2]	89
5.13.1.2 operator""_micrometer() [2/2]	90
5.13.1.3 operator""_nanometer() [1/2]	90
5.13.1.4 operator""_nanometer() [2/2]	90
5.13.1.5 StrToDist()	91
5.14 include/Utilities/Vectorize.hpp File Reference	91
5.14.1 Function Documentation	92
5.14.1.1 operator&&()	92
5.14.1.2 operator"   "   ()	93
5.14.2 Variable Documentation	93
5.14.2.1 Nthreads	94
5.15 src/BayesianClustering/API.cpp File Reference	94
5.15.1 Function Documentation	95
5.15.1.1 AutoRoi_Cluster_Callback()	95
5.15.1.2 AutoRoi_Scan_FullCallback()	95
5.15.1.3 AutoRoi_Scan_SimpleCallback()	97
5.15.1.4 AutoRoi_Scan_ToJson()	97
5.15.1.5 ManualRoi_Cluster_Callback()	98
5.15.1.6 ManualRoi_Scan_FullCallback()	98
5.15.1.7 ManualRoi_Scan_SimpleCallback()	99
5.15.1.8 ManualRoi_Scan_ToJson()	99
5.15.1.9 ScanCallback_Json()	100
5.16 src/BayesianClustering/Cluster.cpp File Reference	100
5.16.1 Function Documentation	100
5.16.1.1 normal_cdf()	101
5.17 src/BayesianClustering/Configuration.cpp File Reference	101
5.18 src/BayesianClustering/Data.cpp File Reference	101
5.19 src/BayesianClustering/DataProxy.cpp File Reference	102
5.20 src/BayesianClustering/LocalizationFile.cpp File Reference	103
5.20.1 Function Documentation	103
5.20.1.1 __LoadCSV__()	103
5.20.1.2 __RecursiveSearch__()	104
5.21 src/BayesianClustering/Roi.cpp File Reference	104
5.22 src/BayesianClustering/RoiProxy.cpp File Reference	105
5.23 src/Cluster.cxx File Reference	105
5.23.1 Function Documentation	106
5.23.1.1 main()	106
5.23.1.2 ReportClusters()	107



---

5.24 src/PythonBindings/PythonBindings.cpp File Reference . . . . .	107
5.25 src/Scan.cxx File Reference . . . . .	107
5.25.1 Function Documentation . . . . .	107
5.25.1.1 main() . . . . .	108
5.26 src/Utilities/GSLInterpolator.cpp File Reference . . . . .	108
5.27 src/Utilities/ProgressBar.cpp File Reference . . . . .	108
5.28 src/Utilities/Units.cpp File Reference . . . . .	109
5.28.1 Function Documentation . . . . .	110
5.28.1.1 StrToDist() . . . . .	110
5.29 src/Utilities/Vectorize.cpp File Reference . . . . .	110
5.29.1 Variable Documentation . . . . .	111
5.29.1.1 Nthreads . . . . .	111
<b>Index</b>	<b>113</b>



# Chapter 1

## Todo List

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

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

Member `Data::PreprocessLocalizationScores` (`std::vector< Data > &aData`, `const ScanConfiguration &aScanConfig`, `const double &aArea`)

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



## Chapter 2

# Class Index

### 2.1 Class List

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

<a href="#">AuxConfiguration</a>	Class for storing the auxilliary configuration parameters . . . . .	7
<a href="#">Cluster</a>	A class representing a cluster . . . . .	13
<a href="#">Data</a>	A class to store the raw data-points . . . . .	17
<a href="#">DataProxy</a>	A light-weight proxy for the raw data-points . . . . .	24
<a href="#">GSLInterpolator</a>	A utility wrapper around the GSL interpolator to give it a clean C++ interface . . . . .	28
<a href="#">LocalizationFile</a>	A class to store the raw data-points . . . . .	35
<a href="#">ManualRol</a>	A struct for storing the manual Rols . . . . .	38
<a href="#">Cluster::Parameter</a>	A struct representing the cluster parameters . . . . .	39
<a href="#">ProgressBar</a>	A utility progress-bar . . . . .	41
<a href="#">ProgressBar2</a>	A utility code timer . . . . .	42
<a href="#">Rol</a>	A class which holds the raw <a href="#">Rol</a> data and global parameters . . . . .	44
<a href="#">Rolproxy</a>	A lightweight wrapper for the <a href="#">Rol</a> to store clusters for a given scan . . . . .	51
<a href="#">ScanConfiguration</a>	Class for storing the scan configuration parameters . . . . .	56
<a href="#">ScanEntry</a>	A struct for storing a result of an individual scan configuration . . . . .	69
<a href="#">ScanConfiguration::tBounds</a>	A struct to store the bounds of a scan in either R or T . . . . .	70



## Chapter 3

# File Index

### 3.1 File List

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

include/BayesianClustering/API.hpp	71
include/BayesianClustering/Cluster.hpp	76
include/BayesianClustering/Configuration.hpp	77
include/BayesianClustering/Data.hpp	78
include/BayesianClustering/DataProxy.hpp	79
include/BayesianClustering/LocalizationFile.hpp	80
include/BayesianClustering/Precision.hpp	81
include/BayesianClustering/Rol.hpp	81
include/BayesianClustering/Rolproxy.hpp	82
include/Utilities/GSLInterpolator.hpp	83
include/Utilities/ListComprehension.hpp	84
include/Utilities/ProgressBar.hpp	87
include/Utilities/Units.hpp	88
include/Utilities/Vectorize.hpp	91
src/Cluster.cxx	105
src/Scan.cxx	107
src/BayesianClustering/API.cpp	94
src/BayesianClustering/Cluster.cpp	100
src/BayesianClustering/Configuration.cpp	101
src/BayesianClustering/Data.cpp	101
src/BayesianClustering/DataProxy.cpp	102
src/BayesianClustering/LocalizationFile.cpp	103
src/BayesianClustering/Rol.cpp	104
src/BayesianClustering/Rolproxy.cpp	105
src/PythonBindings/PythonBindings.cpp	107
src/Utilities/GSLInterpolator.cpp	108
src/Utilities/ProgressBar.cpp	108
src/Utilities/Units.cpp	109
src/Utilities/Vectorize.cpp	110





## Chapter 4

# Class Documentation

### 4.1 AuxConfiguration Class Reference

Class for storing the auxilliary configuration parameters.

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

#### Public Member Functions

- [AuxConfiguration](#) (int argc, char \*\*argv)  
*Default constructor.*
- [AuxConfiguration](#) (const std::vector< std::string > &aArgs)  
*Constructor which parses the parameters when passed in as commandline arguments.*
- 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 [SetConfigFile](#) (const std::string &aFileName)  
*Setter for the config file.*
- 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 std::string & [configFile](#) () const  
*Getter for the config 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.*

## Public Attributes

- bool [mValidate](#)  
*Whether or not to run the validation on the clustering.*
- std::string [mInputFile](#)  
*The input [RoI](#) file.*
- std::string [mOutputFile](#)  
*The output file.*
- std::string [mConfigFile](#)  
*The config file.*
- double [mClusterR](#)  
*The value of  $R$  for clustering.*
- double [mClusterT](#)  
*The value of  $T$  for clustering.*

## Private Member Functions

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

### 4.1.1 Detailed Description

Class for storing the auxilliary configuration parameters.

Definition at line 230 of file Configuration.hpp.

### 4.1.2 Constructor & Destructor Documentation

#### 4.1.2.1 AuxConfiguration() [1/2]

```
AuxConfiguration::AuxConfiguration (
    int argc,
    char ** argv )
```

Default constructor.

Constructor which parses 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 206 of file Configuration.cpp.

References FromVector().

#### 4.1.2.2 AuxConfiguration() [2/2]

```
AuxConfiguration::AuxConfiguration (
    const std::vector< std::string > & aArgs )
```

Constructor which parses the parameters when passed in as commandline arguments.

##### Parameters

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

Definition at line 212 of file Configuration.cpp.

References FromVector().

### 4.1.3 Member Function Documentation

#### 4.1.3.1 ClusterR()

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

Getter for the R value for a clusterization pass.

##### Returns

The R value for a clusterization pass

Definition at line 290 of file Configuration.hpp.

References mClusterR.

Referenced by main().

#### 4.1.3.2 ClusterT()

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

Getter for the T value for a clusterization pass.

##### Returns

The T value for a clusterization pass

Definition at line 297 of file Configuration.hpp.

References mClusterT.

Referenced by main().

#### 4.1.3.3 configFile()

```
const std::string& AuxConfiguration::configFile ( ) const [inline]
```

Getter for the config file.

##### Returns

The name of the config file

Definition at line 283 of file Configuration.hpp.

References mConfigFile.

Referenced by main().

#### 4.1.3.4 FromVector()

```
void AuxConfiguration::FromVector (
    const std::vector< std::string > & aArgs ) [private]
```

Parse the parameters when passed in as commandline arguments.

##### Parameters

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

Definition at line 248 of file Configuration.cpp.

References mClusterR, mClusterT, Nthreads, SetConfigFile(), SetInputFile(), SetOutputFile(), SetValidate(), and StrToDist().

Referenced by AuxConfiguration().

#### 4.1.3.5 inputFile()

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

Getter for the input file.

##### Returns

The name of the input [Rol](#) file

Definition at line 269 of file Configuration.hpp.

References mInputFile.

Referenced by main().

#### 4.1.3.6 outputFile()

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

Getter for the output file.

##### Returns

The name of the output file

Definition at line 276 of file Configuration.hpp.

References mOutputFile.

Referenced by main().

#### 4.1.3.7 SetConfigFile()

```
void AuxConfiguration::SetConfigFile (
    const std::string & aFileName )
```

Setter for the config file.

##### Parameters

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

Definition at line 236 of file Configuration.cpp.

References mConfigFile.

Referenced by FromVector().

#### 4.1.3.8 SetInputFile()

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

Setter for the input file.

##### Parameters

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

Definition at line 223 of file Configuration.cpp.

References mInputFile.

Referenced by FromVector().

#### 4.1.3.9 SetOutputFile()

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

Setter for the output file.

##### Parameters

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

Definition at line 229 of file Configuration.cpp.

References mOutputFile.

Referenced by FromVector().

#### 4.1.3.10 SetValidate()

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

Set whether to validate clusterization.

##### Parameters

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

Definition at line 217 of file Configuration.cpp.

References mValidate.

Referenced by FromVector().

#### 4.1.3.11 validate()

```
const bool& AuxConfiguration::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 262 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

## 4.2 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](#) (const std::size\_t &aParamSize)  
Default constructor.
- [Cluster](#) (const [Data](#) &aData, const std::vector< double > &aSigmabins2)  
Construct a cluster from a single data-point.
- [Cluster](#) (const [Cluster](#) &aOther)=delete  
Deleted copy constructor.
- [Cluster](#) & operator= (const [Cluster](#) &aOther)=delete  
Deleted assignment operator.
- [Cluster](#) ([Cluster](#) &&aOther)=default  
Default move constructor.
- [Cluster](#) & operator= ([Cluster](#) &&aOther)=default  
Default move-assignment constructor.
- [Cluster](#) & operator+= (const [Cluster](#) &aOther)  
Add another cluster to this one.
- [Cluster](#) \* [GetParent](#) ()  
Get a pointer to this cluster's ultimate parent.
- void [UpdateLogScore](#) (const [ScanConfiguration](#) &aScanConfig)  
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.*
- `std::vector< Data * > mData`  
*List of points in the cluster after clustering.*

### 4.2.1 Detailed Description

A class representing a cluster.

Definition at line 16 of file Cluster.hpp.

### 4.2.2 Constructor & Destructor Documentation

#### 4.2.2.1 Cluster() [1/4]

```
Cluster::Cluster (
    const std::size_t & aParamSize )
```

Default constructor.

##### Parameters

<code>aParamSize</code>	The number of sigma-bins
-------------------------	--------------------------

Definition at line 93 of file Cluster.cpp.

#### 4.2.2.2 Cluster() [2/4]

```
Cluster::Cluster (
    const Data & aData,
    const std::vector< double > & aSigmas2 )
```

Construct a cluster from a single data-point.



## Parameters

<i>aData</i>	A data-point with which to initialize the cluster
<i>aSigmas2</i>	The sigma-bins for initializing clusters

Definition at line 99 of file Cluster.cpp.

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

#### 4.2.2.3 Cluster() [3/4]

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

Deleted copy constructor.

## Parameters

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

#### 4.2.2.4 Cluster() [4/4]

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

Default move constructor.

## Parameters

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

### 4.2.3 Member Function Documentation

#### 4.2.3.1 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 159 of file Cluster.cpp.

References GetParent(), and mParent.

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

**4.2.3.2 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 149 of file Cluster.cpp.

References mClusterSize, and mParams.

**4.2.3.3 operator=() [1/2]**

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

Default move-assignment constructor.

**Returns**

Reference to this, for chaining calls

**Parameters**

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

#### 4.2.3.4 operator=() [2/2]

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

Deleted assignment operator.

##### Returns

Reference to this, for chaining calls

##### Parameters

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

#### 4.2.3.5 UpdateLogScore()

```
void Cluster::UpdateLogScore (
    const ScanConfiguration & aScanConfig )
```

Update log-probability after a scan.

##### Parameters

<i>aScanConfig</i>	The configuration parameters for the scan
--------------------	---

Definition at line 118 of file Cluster.cpp.

References `ScanConfiguration::log_probability_sigma()`, `mClusterScore`, `mClusterSize`, `mLastClusterSize`, `mParams`, and `ScanConfiguration::sigmabins()`.

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

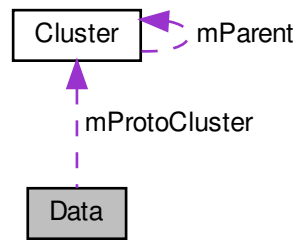
- `include/BayesianClustering/Cluster.hpp`
- `src/BayesianClustering/Cluster.cpp`

## 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, const double &aMax2R, const double &aMax2R2, const std::vector< double > &aSigmabins2)  
*All the necessary pre-processing to get this data-point ready for an RT-scan.*
- void [PreprocessLocalizationScores](#) (std::vector< [Data](#) > &aData, const [ScanConfiguration](#) &aScanConfig, const double &aArea)  
*Calculate the localization score from the local neighbourhood.*
- PRECISION [CalculateLocalizationScore](#) (const std::vector< [Data](#) > &aData, const double &R, const double &aArea) 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 localization 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 16 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

<code>aX</code>	The x-position of the data-point in algorithm units
<code>aY</code>	The y-position of the data-point in algorithm units
<code>aS</code>	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 double & aArea ) 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>	?
<i>aArea</i>	The area of the window for normalizing the log score

**Returns**

The localization score

Definition at line 103 of file Data.cpp.

References mNeighbours.

**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 70 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 62 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 53 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 45 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,
    const double & aMax2R,
    const double & aMax2R2,
    const std::vector< double > & aSigmabins2 )
```

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
<i>aMax2R</i>	Twice the maximum radius out to which we will cluster
<i>aMax2R2</i>	Square of twice the maximum radius out to which we will cluster
<i>aSigmabins2</i>	The sigma-bins for initializing clusters

Definition at line 27 of file Data.cpp.

#### 4.3.3.9 PreprocessLocalizationScores()

```
void Data::PreprocessLocalizationScores (
    std::vector< Data > & aData,
    const ScanConfiguration & aScanConfig,
    const double & aArea )
```

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>	?
<i>aScanConfig</i>	The configuration parameters for the scan
<i>aArea</i>	The area of the window for normalizing the log score

Definition at line 74 of file Data.cpp.

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

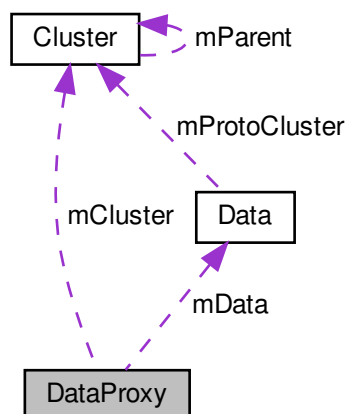
- include/BayesianClustering/Data.hpp
- src/BayesianClustering/Data.cpp

## 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, [Rolproxy](#) &aRol)  
*Entry point clusterization function - a new cluster will be created.*
- void [Clusterize](#) (const PRECISION &a2R2, [Rolproxy](#) &aRol, [Cluster](#) \*aCluster, const std::size\_t &d=0)  
*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 18 of file DataProxy.hpp.

### 4.4.2 Constructor & Destructor Documentation

#### 4.4.2.1 DataProxy() [1/3]

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

Default constructor.

#### Parameters

<a href="#">aData</a>	The data-point for which this is the proxy
-----------------------	--

Definition at line 17 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,
    RoIproxy & aRoI )
```

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

##### Parameters

<i>a2R2</i>	The clusterization radius
<i>aRoI</i>	The RoI-proxy in which we are running

Definition at line 23 of file DataProxy.cpp.

References mCluster, Rolproxy::mClusters, mData, mExclude, Cluster::mParams, and Data::mProtoCluster.

Referenced by Clusterize().

#### 4.4.3.2 Clusterize() [2/2]

```
void DataProxy::Clusterize (
    const PRECISION & a2R2,
    Rolproxy & aRoI,
    Cluster * aCluster,
    const std::size_t & d = 0 )
```

Recursive clusterization function.

##### Parameters

<i>a2R2</i>	The clusterization radius
<i>aRoI</i>	The Rol-proxy in which we are running
<i>aCluster</i>	The cluster we are building
<i>d</i>	The recursion depth

Definition at line 34 of file DataProxy.cpp.

References Clusterize(), GetCluster(), Rolproxy::GetData(), mCluster, Cluster::mClusterSize, mData, mExclude, Data::mNeighbours, Cluster::mParent, Data::mProtoCluster, and RECURSION\_LIMIT.

#### 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 53 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 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.*

### 4.5.1 Detailed Description

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

Definition at line 19 of file `GSLInterpolator.hpp`.

### 4.5.2 Constructor & Destructor Documentation

#### 4.5.2.1 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 9 of file GSLInterpolator.cpp.

References fInterpType, and fSpline.

#### 4.5.2.2 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 19 of file GSLInterpolator.cpp.

References fInterpType, fSpline, and SetData().

#### 4.5.2.3 GSLInterpolator() [3/4]

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

Deleted copy constructor.

## Parameters

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

#### 4.5.2.4 GSLInterpolator() [4/4]

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



Default move constructor.

#### Parameters

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

### 4.5.3 Member Function Documentation

#### 4.5.3.1 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 104 of file GSLInterpolator.hpp.

References Evaluate(), fAccel, and fSpline.

#### 4.5.3.2 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 114 of file GSLInterpolator.hpp.

References Evaluate(), fAccel, and fSpline.

#### 4.5.3.3 Eval()

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

Evaluate the spline at the given x.

##### Parameters

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

##### Returns

The value of the spline at the given x-coordinate

Definition at line 94 of file GSLInterpolator.hpp.

References Evaluate(), fAccel, and fSpline.

Referenced by ScanConfiguration::FromVector().

#### 4.5.3.4 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 75 of file GSLInterpolator.hpp.

References fAccel, and nErrors.

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

#### 4.5.3.5 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 125 of file GSLInterpolator.hpp.

References Evaluate(), fAccel, and fSpline.

#### 4.5.3.6 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
---------------	--------------------

#### 4.5.3.7 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
---------------	--------------------

**4.5.3.8 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 58 of file GSLInterpolator.hpp.

Referenced by GSLInterpolator().

**4.5.3.9 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 38 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

## 4.6 LocalizationFile Class Reference

A class to store the raw data-points.

```
#include <LocalizationFile.hpp>
```

### Public Member Functions

- [LocalizationFile](#) (const std::string &aFilename)  
*Constructor.*
- [LocalizationFile](#) (const [LocalizationFile](#) &aOther)=delete  
*Deleted copy constructor.*
- [LocalizationFile](#) & operator= (const [LocalizationFile](#) &aOther)=delete  
*Deleted assignment operator.*
- [LocalizationFile](#) ([LocalizationFile](#) &&aOther)=default  
*Default move constructor.*
- [LocalizationFile](#) & operator= ([LocalizationFile](#) &&aOther)=default  
*Default move-assignment constructor.*
- [~LocalizationFile](#) ()=default  
*Default destructor.*
- void [ExtractRols](#) (const std::function< void([Rol](#) &) > &aCallback) const  
*Automatically extract the Rols.*
- void [ExtractRols](#) (const [ManualRol](#) &aRol, const std::function< void([Rol](#) &) > &aCallback) const  
*Manually extract an [Rol](#).*

### Private Attributes

- std::vector< [Data](#) > [mData](#)  
*The localizations in the file.*

#### 4.6.1 Detailed Description

A class to store the raw data-points.

Definition at line 25 of file LocalizationFile.hpp.

#### 4.6.2 Constructor & Destructor Documentation

##### 4.6.2.1 LocalizationFile() [1/3]

```
LocalizationFile::LocalizationFile (
    const std::string & aFilename )
```

Constructor.

## Parameters

<i>aFilename</i>	The name of the localizations file
------------------	------------------------------------

Definition at line 67 of file LocalizationFile.cpp.

References `__LoadCSV__()`, `mData`, `Nthreads`, and `range()`.

#### 4.6.2.2 LocalizationFile() [2/3]

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

Deleted copy constructor.

## Parameters

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

#### 4.6.2.3 LocalizationFile() [3/3]

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

Default move constructor.

## Parameters

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

### 4.6.3 Member Function Documentation

#### 4.6.3.1 ExtractRols() [1/2]

```
void LocalizationFile::ExtractRoIs (
    const ManualRoI & aRoI,
    const std::function< void(RoI &) > & aCallback ) const
```

Manually extract an [RoI](#).

## Parameters

<i>aRol</i>	The manual <a href="#">Rol</a> window
<i>aCallback</i>	A handler for each <a href="#">Rol</a> found

Definition at line 256 of file LocalizationFile.cpp.

References [ManualRol::height](#), [mData](#), [Rol::SetCentre\(\)](#), [Rol::SetWidth\(\)](#), [ManualRol::width](#), [ManualRol::x](#), and [ManualRol::y](#).

**4.6.3.2 ExtractRols()** [2/2]

```
void LocalizationFile::ExtractRols (
    const std::function< void(Rol &) > & aCallback ) const
```

Automatically extract the Rols.

## Parameters

<i>aCallback</i>	A handler for each <a href="#">Rol</a> found
------------------	--

Definition at line 119 of file LocalizationFile.cpp.

References [\\_\\_RecursiveSearch\\_\\_\(\)](#), [mData](#), [Rol::mData](#), [Rol::SetCentre\(\)](#), and [Rol::SetWidth\(\)](#).

Referenced by [AutoRoi\\_Cluster\\_Callback\(\)](#), [AutoRoi\\_Scan\\_FullCallback\(\)](#), [ManualRoi\\_Cluster\\_Callback\(\)](#), and [ManualRoi\\_Scan\\_FullCallback\(\)](#).

**4.6.3.3 operator=()** [1/2]

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

Deleted assignment operator.

## Returns

Reference to this, for chaining calls

## Parameters

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

#### 4.6.3.4 operator=() [2/2]

```
LocalizationFile& LocalizationFile::operator= (
    LocalizationFile && 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/[LocalizationFile.hpp](#)
- src/BayesianClustering/[LocalizationFile.cpp](#)

## 4.7 ManualRol Struct Reference

A struct for storing the manual Rols.

```
#include <LocalizationFile.hpp>
```

### Public Attributes

- double [x](#)  
*The x-centre of the [Rol](#).*
- double [y](#)  
*The y-centre of the [Rol](#).*
- double [width](#)  
*The width of the [Rol](#).*
- double [height](#)  
*The height of the [Rol](#).*

#### 4.7.1 Detailed Description

A struct for storing the manual Rols.

Definition at line 15 of file [LocalizationFile.hpp](#).

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

- include/BayesianClustering/[LocalizationFile.hpp](#)



## 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 21 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 47 of file Cluster.cpp.

References `normal_cdf()`.

#### 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 72 of file Cluster.cpp.

References `normal_cdf()`.

#### 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 37 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 7 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 26 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 34 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 46 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 Rol Class Reference

A class which holds the raw [Rol](#) data and global parameters.

```
#include <RoI.hpp>
```

### Public Member Functions

- [Rol](#) (std::vector< [Data](#) > &&aData)  
*Default Constructor.*
- [Rol](#) (const [Rol](#) &aOther)=delete  
*Deleted copy constructor.*
- [Rol](#) & operator= (const [Rol](#) &aOther)=delete  
*Deleted assignment operator.*
- [Rol](#) ([Rol](#) &&aOther)=default  
*Default move constructor.*
- [Rol](#) & operator= ([Rol](#) &&aOther)=default  
*Default move-assignment constructor.*
- void [Preprocess](#) (const double &aMaxR, const std::vector< double > &aSigmabins2)  
*All the necessary pre-processing to get the [Rol](#) ready for an RT-scan.*
- void [ScanRT](#) (const [ScanConfiguration](#) &aScanConfig, const std::function< void([Rolproxy](#) &, const double &, const double &) > &aCallback)  
*Run the scan.*
- void [Clusterize](#) (const double &R, const double &T, const std::function< void([Rolproxy](#) &) > &aCallback)  
*Run clusterization for a specific choice of R and T.*
- 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 [Rol](#) window.*
- 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 width of the ROI window.*
- double [getWidthY](#) () const  
*Getter for the height of the ROI window.*
- double [getArea](#) () const  
*Getter for the height of the ROI window.*

## Public Attributes

- `std::vector< Data > mData`  
*The collection of raw data points.*

## Private Attributes

- `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.*
- `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.*
- `double mArea`  
*The area of the window in physical units.*

### 4.11.1 Detailed Description

A class which holds the raw [RoI](#) data and global parameters.

Definition at line 17 of file `RoI.hpp`.

### 4.11.2 Constructor & Destructor Documentation

#### 4.11.2.1 `RoI()` [1/3]

```
RoI::RoI (
    std::vector< Data > && aData )
```

Default Constructor.

#### Parameters

<code>aData</code>	The set of data-points in the <a href="#">RoI</a>
--------------------	---

Definition at line 17 of file `RoI.cpp`.

References `mData`.

#### 4.11.2.2 Rol() [2/3]

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

Deleted copy constructor.

##### Parameters

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

#### 4.11.2.3 Rol() [3/3]

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

Default move constructor.

##### Parameters

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

### 4.11.3 Member Function Documentation

#### 4.11.3.1 Clusterize()

```
void RoI::Clusterize (
    const double & R,
    const double & T,
    const std::function< void(RoIproxy &) > & 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 54 of file Rol.cpp.

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

Referenced by AutoRoi\_Cluster\_Callback(), and ManualRoi\_Cluster\_Callback().



#### 4.11.3.2 getArea()

```
double RoI::getArea ( ) const [inline]
```

Getter for the height of the ROI window.

##### Returns

The height of the ROI window

Definition at line 98 of file RoI.hpp.

References mArea.

Referenced by RoIproxy::Clusterize(), and ScanRT().

#### 4.11.3.3 getCentreX()

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

Getter for the x-coordinate of the physical centre.

##### Returns

The x-coordinate of the physical centre

Definition at line 71 of file RoI.hpp.

References mPhysicalCentreX.

#### 4.11.3.4 getCentreY()

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

Getter for the y-coordinate of the physical centre.

##### Returns

The y-coordinate of the physical centre

Definition at line 77 of file RoI.hpp.

References mPhysicalCentreY.

#### 4.11.3.5 getWidthX()

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

Getter for the width of the ROI window.

##### Returns

The width of the ROI window

Definition at line 84 of file RoI.hpp.

References mWidthX.

#### 4.11.3.6 getWidthY()

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

Getter for the height of the ROI window.

##### Returns

The height of the ROI window

Definition at line 91 of file RoI.hpp.

References mWidthY.

#### 4.11.3.7 operator=() [1/2]

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

Deleted assignment operator.

##### Returns

Reference to this, for chaining calls

##### Parameters

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

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

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

Default move-assignment constructor.

**Returns**

Reference to this, for chaining calls

**Parameters**

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

**4.11.3.9 Preprocess()**

```
void RoI::Preprocess (
    const double & aMaxR,
    const std::vector< double > & aSigmabins2 )
```

All the necessary pre-processing to get the [RoI](#) ready for an RT-scan.

**Parameters**

<i>aMaxR</i>	The maximum radius out to which we should pre-process
<i>aSigmabins2</i>	The number of sigma bins

Definition at line 28 of file RoI.cpp.

References `mData`, and `range()`.

Referenced by `Clusterize()`, and `ScanRT()`.

**4.11.3.10 ScanRT()**

```
void RoI::ScanRT (
    const ScanConfiguration & aScanConfig,
    const std::function< void(RoIproxy &, const double &, const double &) > & a←
    Callback )
```

Run the scan.

**Parameters**

<i>aScanConfig</i>	The configuration parameters for the scan
<i>aCallback</i>	A callback for each RT-scan result

Definition at line 37 of file RoI.cpp.

References `getArea()`, `mData`, `Nthreads`, `Preprocess()`, `range()`, `ScanConfiguration::Rbounds()`, and `ScanConfiguration::sigmabins2()`.

Referenced by `AutoRoi_Scan_FullCallback()`, and `ManualRoi_Scan_FullCallback()`.

#### 4.11.3.11 SetCentre()

```
void RoI::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 66 of file RoI.cpp.

References `mPhysicalCentreX`, and `mPhysicalCentreY`.

Referenced by `LocalizationFile::ExtractRols()`.

#### 4.11.3.12 SetWidth()

```
void RoI::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 73 of file RoI.cpp.

References `mArea`, `mWidthX`, and `mWidthY`.

Referenced by `LocalizationFile::ExtractRols()`.

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

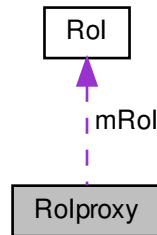
- `include/BayesianClustering/RoI.hpp`
- `src/BayesianClustering/RoI.cpp`

## 4.12 Rolproxy Class Reference

A lightweight wrapper for the [Rol](#) to store clusters for a given scan.

```
#include <Rolproxy.hpp>
```

Collaboration diagram for Rolproxy:



### Public Member Functions

- [Rolproxy](#) ([Rol](#) &aRol)  
*Default constructor.*
- [Rolproxy](#) (const [Rolproxy](#) &aOther)=delete  
*Deleted copy constructor.*
- [Rolproxy](#) & [operator=](#) (const [Rolproxy](#) &aOther)=delete  
*Deleted assignment operator.*
- [Rolproxy](#) ([Rolproxy](#) &&aOther)=default  
*Default move constructor.*
- [Rolproxy](#) & [operator=](#) ([Rolproxy](#) &&aOther)=default  
*Default move-assignment constructor.*
- void [CheckClusterization](#) (const double &R, const double &T)  
*Run validation tests on the clusters.*
- void [ScanRT](#) (const [ScanConfiguration](#) &aScanConfig, const std::function< void([Rolproxy](#) &, const double &, const double &) > &aCallback, const uint8\_t &aParallelization=1, const uint8\_t &aOffset=0, const bool &aValidate=false)  
*Run an RT-scan.*
- void [Clusterize](#) (const double &R, const double &T, const std::function< void([Rolproxy](#) &) > &aCallback)  
*Run clusterization for a specific choice of R and T.*
- void [UpdateLogScore](#) (const [ScanConfiguration](#) &aScanConfig)  
*Update log-probability after a scan.*
- void [ValidateLogScore](#) (const [ScanConfiguration](#) &aScanConfig)  
*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 [RoI](#) 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.
- `const RoI & mRoI`  
The underlying [RoI](#) this is a proxy to.

### 4.12.1 Detailed Description

A lightweight wrapper for the [RoI](#) to store clusters for a given scan.

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

### 4.12.2 Constructor & Destructor Documentation

#### 4.12.2.1 `Rolproxy()` [1/3]

```
RoIproxy::RoIproxy (
    RoI & aRoI )
```

Default constructor.

##### Parameters

<i>aRoI</i>	An <a href="#">RoI</a> for which this is a lightweight proxy
-------------	--

Definition at line 17 of file `Rolproxy.cpp`.

References `mClusters`, `Rol::mData`, and `mData`.

#### 4.12.2.2 `Rolproxy()` [2/3]

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

Deleted copy constructor.

## Parameters

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

**4.12.2.3 Rolproxy() [3/3]**

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

Default move constructor.

## Parameters

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

**4.12.3 Member Function Documentation****4.12.3.1 CheckClusterization()**

```
void RoIproxy::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 25 of file Rolproxy.cpp.

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

**4.12.3.2 Clusterize()**

```
void RoIproxy::Clusterize (
    const double & R,
    const double & T,
    const std::function< void(RoIproxy &) > & 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 129 of file Rolproxy.cpp.

References Rol::getArea(), mClusters, Rol::mData, mData, and mRol.

Referenced by Rol::Clusterize().

#### 4.12.3.3 GetData()

```
DataProxy& Rolproxy::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 69 of file Rolproxy.hpp.

References mData.

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

#### 4.12.3.4 operator=() [1/2]

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

Deleted assignment operator.

## Returns

Reference to this, for chaining calls



## Parameters

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

## 4.12.3.5 operator=() [2/2]

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

Default move-assignment constructor.

## Returns

Reference to this, for chaining calls

## Parameters

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

## 4.12.3.6 ScanRT()

```
void RoIproxy::ScanRT (
    const ScanConfiguration & aScanConfig,
    const std::function< void(RoIproxy &, const double &, const double &) > & aCallback,
    const uint8_t & aParallelization = 1,
    const uint8_t & aOffset = 0,
    const bool & aValidate = false )
```

Run an RT-scan.

## Parameters

<i>aScanConfig</i>	The configuration parameters for the scan
<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
<i>aValidate</i>	Run validation of the score calculation

Definition at line 95 of file RoIproxy.cpp.

#### 4.12.3.7 UpdateLogScore()

```
void RoIproxy::UpdateLogScore (
    const ScanConfiguration & aScanConfig )
```

Update log-probability after a scan.

##### Parameters

<i>aScanConfig</i>	The configuration parameters for the scan
--------------------	---

Definition at line 209 of file Rolproxy.cpp.

References ScanConfiguration::alpha(), ScanConfiguration::logAlpha(), ScanConfiguration::logGammaAlpha(), ScanConfiguration::logPb(), ScanConfiguration::logPbDagger(), mBackgroundCount, mClusterCount, mClusteredCount, mClusters, mData, mLogP, and ScanConfiguration::sigmabins().

#### 4.12.3.8 ValidateLogScore()

```
void RoIproxy::ValidateLogScore (
    const ScanConfiguration & aScanConfig )
```

Sean's validation code for testing when the running log-score fails.

##### Parameters

<i>aScanConfig</i>	The configuration parameters for the scan
--------------------	---

Definition at line 150 of file Rolproxy.cpp.

References mClusters, mData, Cluster::mParams, Data::s, ScanConfiguration::sigmabins2(), ScanConfiguration::sigmacount(), Data::x, and Data::y.

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

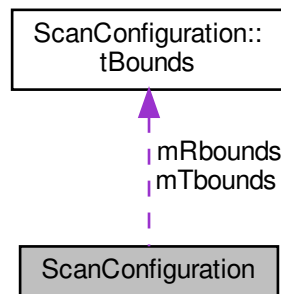
- include/BayesianClustering/Rolproxy.hpp
- src/BayesianClustering/Rolproxy.cpp

## 4.13 ScanConfiguration Class Reference

Class for storing the scan configuration parameters.

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

Collaboration diagram for ScanConfiguration:



## Classes

- struct [tBounds](#)

*A struct to store the bounds of a scan in either R or T.*

## Public Member Functions

- [ScanConfiguration](#) (int argc, char \*\*argv)  
*Default constructor.*
- [ScanConfiguration](#) (const std::vector< std::string > &aArgs)  
*Constructor which parses the parameters when passed in as commandline arguments.*
- [ScanConfiguration](#) (const std::string &aCfgFile)  
*Constructor which parses the parameters when passed in as commandline arguments.*
- 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, const double &aMaxScanR)  
*Setter for the R bins for the RT scan.*
- void [SetTBins](#) (const std::size\_t &aTbins, const double &aMinScanT, const double &aMaxScanT)
- void [SetPb](#) (const double &aPB)  
*Setter for the P\_b parameter.*
- void [SetAlpha](#) (const double &aAlpha)  
*Setter for the alpha parameter.*
- 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 tBounds & Rbounds () const`
- Getter for the bounds of R to scan.*

  - `const tBounds & Tbounds () const`
- Getter for the bounds of T 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.*

## Private Member Functions

- `void FromVector (const std::vector< std::string > &aArgs)`  
*Parse the parameters when passed in as commandline arguments.*

## Private Attributes

- `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.*
- `tBounds mRbounds`  
*The bounds of R to scan.*
- `tBounds mTbounds`  
*The bounds of T 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.*

### 4.13.1 Detailed Description

Class for storing the scan configuration parameters.

Definition at line 11 of file Configuration.hpp.

### 4.13.2 Constructor & Destructor Documentation

#### 4.13.2.1 ScanConfiguration() [1/3]

```
ScanConfiguration::ScanConfiguration (
    int argc,
    char ** argv )
```

Default constructor.

Constructor which parses 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 22 of file Configuration.cpp.

References [FromVector\(\)](#).

#### 4.13.2.2 ScanConfiguration() [2/3]

```
ScanConfiguration::ScanConfiguration (
    const std::vector< std::string > & aArgs )
```

Constructor which parses the parameters when passed in as commandline arguments.

**Parameters**

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

Definition at line 32 of file Configuration.cpp.

References FromVector().

**4.13.2.3 ScanConfiguration() [3/3]**

```
ScanConfiguration::ScanConfiguration (
    const std::string & aCfgFile )
```

Constructor which parses the parameters when passed in as commandline arguments.

**Parameters**

<i>aCfgFile</i>	A Scan-parameter config file name
-----------------	-----------------------------------

Definition at line 41 of file Configuration.cpp.

References FromVector().

**4.13.3 Member Function Documentation****4.13.3.1 alpha()**

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

Getter for the alpha parameter.

**Returns**

The alpha parameter

Definition at line 178 of file Configuration.hpp.

References mAlpha.

Referenced by Rolproxy::UpdateLogScore().

**4.13.3.2 FromVector()**

```
void ScanConfiguration::FromVector (
    const std::vector< std::string > & aArgs ) [private]
```

Parse the parameters when passed in as commandline arguments.

**Parameters**

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

Definition at line 149 of file Configuration.cpp.

References `GSLInterpolator::Eval()`, `SetAlpha()`, `SetPb()`, `SetRBins()`, `SetSigmaParameters()`, `SetTBins()`, and `Str↵ToDist()`.

Referenced by `ScanConfiguration()`.

**4.13.3.3 log\_probability\_sigma() [1/2]**

```
const std::vector< double >& ScanConfiguration::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 116 of file Configuration.hpp.

References `mLogProbabilitySigma`.

Referenced by `Cluster::UpdateLogScore()`.

**4.13.3.4 log\_probability\_sigma() [2/2]**

```
const double& ScanConfiguration::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 145 of file Configuration.hpp.

References `mLogProbabilitySigma`.

#### 4.13.3.5 logAlpha()

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

Getter for the logarithm of the alpha parameter.

##### Returns

The logarithm of the alpha parameter

Definition at line 184 of file Configuration.hpp.

References mLogAlpha.

Referenced by Rolproxy::UpdateLogScore().

#### 4.13.3.6 logGammaAlpha()

```
const double& ScanConfiguration::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 190 of file Configuration.hpp.

References mLogGammaAlpha.

Referenced by Rolproxy::UpdateLogScore().

#### 4.13.3.7 logPb()

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

Logarithm of the P\_b parameter.

##### Returns

Logarithm of the P\_b parameter

Definition at line 165 of file Configuration.hpp.

References mLogPb.

Referenced by Rolproxy::UpdateLogScore().



**4.13.3.8 logPbDagger()**

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

Logarithm of the ( 1 - P\_b ) parameter.

**Returns**

Logarithm of the ( 1 - P\_b ) parameter

Definition at line 171 of file Configuration.hpp.

References mLogPbDagger.

Referenced by Rolproxy::UpdateLogScore().

**4.13.3.9 probability\_sigma() [1/2]**

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

Getter for the probabilities of a given sigma.

**Returns**

The probabilities of given sigma

Definition at line 110 of file Configuration.hpp.

References mProbabilitySigma.

**4.13.3.10 probability\_sigma() [2/2]**

```
const double& ScanConfiguration::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 138 of file Configuration.hpp.

References mProbabilitySigma.

#### 4.13.3.11 Rbounds()

```
const tBounds& ScanConfiguration::Rbounds ( ) const [inline]
```

Getter for the bounds of R to scan.

##### Returns

The lbounds of R to scan

Definition at line 152 of file Configuration.hpp.

References mRbounds.

Referenced by Rol::ScanRT().

#### 4.13.3.12 SetAlpha()

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

Setter for the alpha parameter.

##### Parameters

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

Definition at line 111 of file Configuration.cpp.

References mAlpha, mLogAlpha, and mLogGammaAlpha.

Referenced by FromVector().

#### 4.13.3.13 SetPb()

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

Setter for the P\_b parameter.

## Parameters

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

Definition at line 104 of file Configuration.cpp.

References mLogPb, and mLogPbDagger.

Referenced by FromVector().

**4.13.3.14 SetRBins()**

```
void ScanConfiguration::SetRBins (
    const std::size_t & aRbins,
    const double & aMinScanR,
    const double & aMaxScanR )
```

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 84 of file Configuration.cpp.

References ScanConfiguration::tBounds::bins, ScanConfiguration::tBounds::max, ScanConfiguration::tBounds::min, mRbounds, and ScanConfiguration::tBounds::spacing.

Referenced by FromVector().

**4.13.3.15 SetSigmaParameters()**

```
void ScanConfiguration::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 56 of file Configuration.cpp.

References `mLogProbabilitySigma`, `mProbabilitySigma`, `mSigmabins`, `mSigmabins2`, `mSigmacount`, `mSigmaspacing`, and `range()`.

Referenced by `FromVector()`.

#### 4.13.3.16 SetTBins()

```
void ScanConfiguration::SetTBins (
    const std::size_t & aTbins,
    const double & aMinScanT,
    const double & aMaxScanT )
```

##### 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 94 of file Configuration.cpp.

References `ScanConfiguration::tBounds::bins`, `ScanConfiguration::tBounds::max`, `ScanConfiguration::tBounds::min`, `mTbounds`, and `ScanConfiguration::tBounds::spacing`.

Referenced by `FromVector()`.

#### 4.13.3.17 sigmabins() [1/2]

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

Getter for the values of sigma.

##### Returns

The values of sigma

Definition at line 98 of file Configuration.hpp.

References `mSigmabins`.

Referenced by `Cluster::UpdateLogScore()`, and `Rolproxy::UpdateLogScore()`.

#### 4.13.3.18 sigmabins() [2/2]

```
const double& ScanConfiguration::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 124 of file Configuration.hpp.

References mSigmabins.

**4.13.3.19 sigmabins2() [1/2]**

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

Getter for the values of sigma squared.

**Returns**

The values of sigma squared

Definition at line 104 of file Configuration.hpp.

References mSigmabins2.

Referenced by Rol::ScanRT(), and Rolproxy::ValidateLogScore().

**4.13.3.20 sigmabins2() [2/2]**

```
const double& ScanConfiguration::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 131 of file Configuration.hpp.

References mSigmabins2.

#### 4.13.3.21 **sigmacount()**

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

Getter for the sigma count.

##### Returns

The sigma count

Definition at line 84 of file Configuration.hpp.

References mSigmacount.

Referenced by Rolproxy::ValidateLogScore().

#### 4.13.3.22 **sigmaspacing()**

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

Getter for the sigma spacing.

##### Returns

The sigma spacing

Definition at line 91 of file Configuration.hpp.

References mSigmaspacing.

#### 4.13.3.23 **Tbounds()**

```
const tBounds& ScanConfiguration::Tbounds ( ) const [inline]
```

Getter for the bounds of T to scan.

##### Returns

The lbounds of T to scan

Definition at line 158 of file Configuration.hpp.

References mTbounds.

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

- include/BayesianClustering/[Configuration.hpp](#)
- src/BayesianClustering/[Configuration.cpp](#)

## 4.14 ScanEntry Struct Reference

A struct for storing a result of an individual scan configuration.

```
#include <API.hpp>
```

### Public Member Functions

- bool [operator<](#) (const [ScanEntry](#) &aOther)  
*Comparison operator for sorting.*

### Public Attributes

- double [r](#)  
*The R parameter*
- double [t](#)  
*The T parameter.*
- double [score](#)  
*The score.*

#### 4.14.1 Detailed Description

A struct for storing a result of an individual scan configuration.

Definition at line 15 of file API.hpp.

#### 4.14.2 Member Function Documentation

##### 4.14.2.1 [operator<\(\)](#)

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

Comparison operator for sorting.

##### Returns

Whether we are smaller than the others

##### Parameters

<i>aOther</i>	Another <a href="#">ScanEntry</a> to compare against
---------------	--

Definition at line 23 of file API.hpp.

References `r`, and `t`.

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

- `include/BayesianClustering/API.hpp`

## 4.15 ScanConfiguration::tBounds Struct Reference

A struct to store the bounds of a scan in either `R` or `T`.

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

### Public Attributes

- `double min`  
*The lowest value of `R` to scan.*
- `double max`  
*The largest value of `R` to scan.*
- `double spacing`  
*The spacing of value of `R` to scan.*
- `std::size_t bins`  
*The number of `R` values to scan.*

### 4.15.1 Detailed Description

A struct to store the bounds of a scan in either `R` or `T`.

Definition at line 16 of file Configuration.hpp.

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

- `include/BayesianClustering/Configuration.hpp`

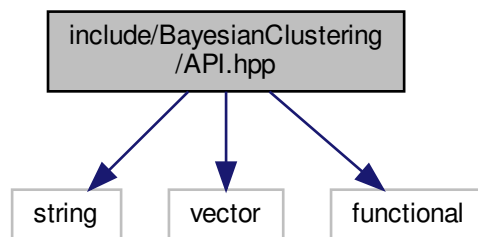


## Chapter 5

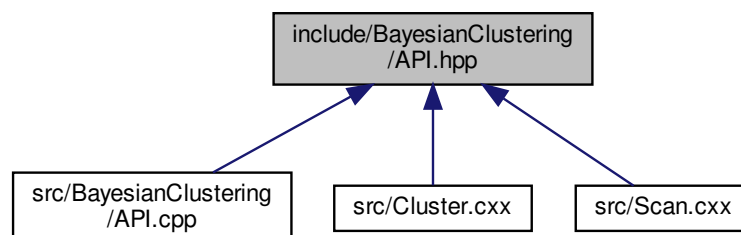
# File Documentation

### 5.1 include/BayesianClustering/API.hpp File Reference

```
#include <string>
#include <vector>
#include <functional>
Include dependency graph for API.hpp:
```



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



## Classes

- struct [ScanEntry](#)

*A struct for storing a result of an individual scan configuration.*

## Functions

- void [AutoRoi\\_Scan\\_FullCallback](#) (const std::string &InFile, const [ScanConfiguration](#) &aScanConfig, const std::function< void([RoiProxy](#) &, const double &, const double &) > &aCallback)  
*Automatically extract [Roi](#), run scan and apply a full call-back.*
- void [AutoRoi\\_Scan\\_SimpleCallback](#) (const std::string &InFile, const [ScanConfiguration](#) &aScanConfig, const std::function< void(const std::vector< [ScanEntry](#) > &) > &aCallback)  
*Automatically extract [Roi](#), run scan and apply a simple call-back.*
- void [AutoRoi\\_Scan\\_ToJson](#) (const std::string &InFile, const [ScanConfiguration](#) &aScanConfig, const std::string &OutFile)  
*Automatically extract [Roi](#), run scan and dump to JSON file.*
- void [AutoRoi\\_Cluster\\_Callback](#) (const std::string &InFile, const double &aR, const double &aT, const std::function< void([RoiProxy](#) &) > &aCallback)  
*Automatically extract [Roi](#), clusterize and apply a call-back.*
- void [ManualRoi\\_Scan\\_FullCallback](#) (const std::string &InFile, const [ManualRoi](#) &aManualRoi, const [ScanConfiguration](#) &aScanConfig, const std::function< void([RoiProxy](#) &, const double &, const double &) > &aCallback)  
*Manually extract [Roi](#), run scan and apply a full call-back.*
- void [ManualRoi\\_Scan\\_SimpleCallback](#) (const std::string &InFile, const [ManualRoi](#) &aManualRoi, const [ScanConfiguration](#) &aScanConfig, const std::function< void(const std::vector< [ScanEntry](#) > &) > &aCallback)  
*Manually extract [Roi](#), run scan and apply a simple call-back.*
- void [ManualRoi\\_Scan\\_ToJson](#) (const std::string &InFile, const [ManualRoi](#) &aManualRoi, const [ScanConfiguration](#) &aScanConfig, const std::string &OutFile)  
*Manually extract [Roi](#), run scan and dump to JSON file.*
- void [ManualRoi\\_Cluster\\_Callback](#) (const std::string &InFile, const [ManualRoi](#) &aManualRoi, const double &aR, const double &aT, const std::function< void([RoiProxy](#) &) > &aCallback)  
*Manually extract [Roi](#), clusterize and apply a call-back.*

## 5.1.1 Function Documentation

### 5.1.1.1 AutoRoi\_Cluster\_Callback()

```
void AutoRoi_Cluster_Callback (
    const std::string & aInFile,
    const double & aR,
    const double & aT,
    const std::function< void(RoiProxy &) > & aCallback )
```

Automatically extract [Roi](#), clusterize and apply a call-back.

#### Parameters

<i>aInFile</i>	The name of the localization file
<i>aR</i>	The R value of the clusterizer
<i>aT</i>	The T value of the clusterizer
<i>aCallback</i>	The callback to be applied

Definition at line 51 of file API.cpp.

References Rol::Clusterize(), and LocalizationFile::ExtractRols().

Referenced by main().

#### 5.1.1.2 AutoRoi\_Scan\_FullCallback()

```
void AutoRoi_Scan_FullCallback (
    const std::string & aInFile,
    const ScanConfiguration & aScanConfig,
    const std::function< void(RoIproxy &, const double &, const double &) > & aCallback )
```

Automatically extract Rol, run scan and apply a full call-back.

##### Parameters

<i>aInFile</i>	The name of the localization file
<i>aScanConfig</i>	The configuration for the scan
<i>aCallback</i>	The full callback to be applied

Definition at line 32 of file API.cpp.

References LocalizationFile::ExtractRols(), and Rol::ScanRT().

Referenced by AutoRoi\_Scan\_SimpleCallback().

#### 5.1.1.3 AutoRoi\_Scan\_SimpleCallback()

```
void AutoRoi_Scan_SimpleCallback (
    const std::string & aInFile,
    const ScanConfiguration & aScanConfig,
    const std::function< void(const std::vector< ScanEntry > &) > & aCallback )
```

Automatically extract Rol, run scan and apply a simple call-back.

##### Parameters

<i>aInFile</i>	The name of the localization file
<i>aScanConfig</i>	The configuration for the scan
<i>aCallback</i>	The simple callback to be applied

Definition at line 37 of file API.cpp.

References AutoRoi\_Scan\_FullCallback(), and Rolproxy::mLogP.

Referenced by `AutoRoi_Scan_ToJson()`.

#### 5.1.1.4 AutoRoi\_Scan\_ToJson()

```
void AutoRoi_Scan_ToJson (
    const std::string & aInFile,
    const ScanConfiguration & aScanConfig,
    const std::string & aOutFile )
```

Automatically extract [Roi](#), run scan and dump to JSON file.

##### Parameters

<i>aInFile</i>	The name of the localization file
<i>aScanConfig</i>	The configuration for the scan
<i>aOutFile</i>	The name of the output JSON file

Definition at line 46 of file `API.cpp`.

References `AutoRoi_Scan_SimpleCallback()`, and `ScanCallback_Json()`.

Referenced by `main()`.

#### 5.1.1.5 ManualRoi\_Cluster\_Callback()

```
void ManualRoi_Cluster_Callback (
    const std::string & aInFile,
    const ManualRoi & aManualRoi,
    const double & aR,
    const double & aT,
    const std::function< void(RoiProxy &) > & aCallback )
```

Manually extract [Roi](#), clusterize and apply a call-back.

##### Parameters

<i>aInFile</i>	The name of the localization file
<i>aManualRoi</i>	The manually-specified <a href="#">Roi</a> window
<i>aR</i>	The R value of the clusterizer
<i>aT</i>	The T value of the clusterizer
<i>aCallback</i>	The callback to be applied

Definition at line 77 of file `API.cpp`.

References `Roi::Clusterize()`, and `LocalizationFile::ExtractRois()`.

### 5.1.1.6 ManualRoi\_Scan\_FullCallback()

```
void ManualRoi_Scan_FullCallback (
    const std::string & aInFile,
    const ManualRoi & aManualRoi,
    const ScanConfiguration & aScanConfig,
    const std::function< void(RoiProxy &, const double &, const double &) > & aCallback )
```

Manually extract [Roi](#), run scan and apply a full call-back.

#### Parameters

<i>aInFile</i>	The name of the localization file
<i>aManualRoi</i>	The manually-specified <a href="#">Roi</a> window
<i>aScanConfig</i>	The configuration for the scan
<i>aCallback</i>	The full callback to be applied

Definition at line 58 of file API.cpp.

References [LocalizationFile::ExtractRols\(\)](#), and [Roi::ScanRT\(\)](#).

Referenced by [ManualRoi\\_Scan\\_SimpleCallback\(\)](#).

### 5.1.1.7 ManualRoi\_Scan\_SimpleCallback()

```
void ManualRoi_Scan_SimpleCallback (
    const std::string & aInFile,
    const ManualRoi & aManualRoi,
    const ScanConfiguration & aScanConfig,
    const std::function< void(const std::vector< ScanEntry > &) > & aCallback )
```

Manually extract [Roi](#), run scan and apply a simple call-back.

#### Parameters

<i>aInFile</i>	The name of the localization file
<i>aManualRoi</i>	The manually-specified <a href="#">Roi</a> window
<i>aScanConfig</i>	The configuration for the scan
<i>aCallback</i>	The simple callback to be applied

Definition at line 63 of file API.cpp.

References [ManualRoi\\_Scan\\_FullCallback\(\)](#), and [RoiProxy::mLogP](#).

Referenced by [ManualRoi\\_Scan\\_ToJson\(\)](#).

### 5.1.1.8 ManualRoi\_Scan\_ToJson()

```
void ManualRoi_Scan_ToJson (
    const std::string & aInFile,
    const ManualRoi & aManualRoi,
    const ScanConfiguration & aScanConfig,
    const std::string & aOutFile )
```

Manually extract [Roi](#), run scan and dump to JSON file.

#### Parameters

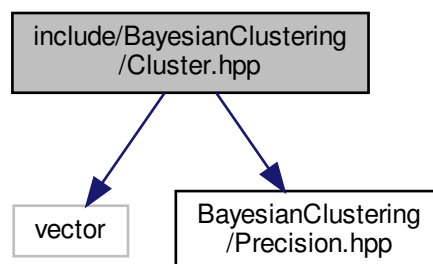
<i>aInFile</i>	The name of the localization file
<i>aManualRoi</i>	The manually-specified <a href="#">Roi</a> window
<i>aScanConfig</i>	The configuration for the scan
<i>aOutFile</i>	The name of the output JSON file

Definition at line 72 of file API.cpp.

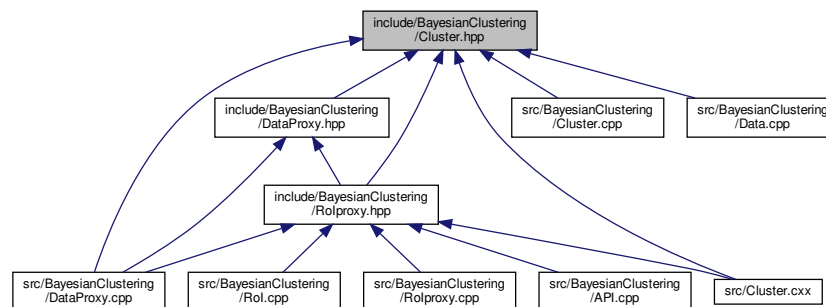
References [ManualRoi\\_Scan\\_SimpleCallback\(\)](#), and [ScanCallback\\_Json\(\)](#).

## 5.2 include/BayesianClustering/Cluster.hpp File Reference

```
#include <vector>
#include "BayesianClustering/Precision.hpp"
Include dependency graph for Cluster.hpp:
```



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



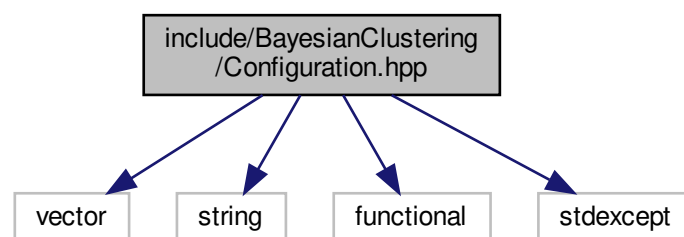
## Classes

- class `Cluster`  
A class representing a cluster.
- struct `Cluster::Parameter`  
A struct representing the cluster parameters.

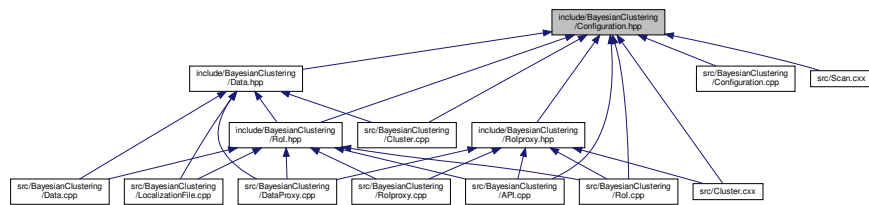
## 5.3 include/BayesianClustering/Configuration.hpp File Reference

```
#include <vector>
#include <string>
#include <functional>
#include <stdexcept>
```

Include dependency graph for Configuration.hpp:



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

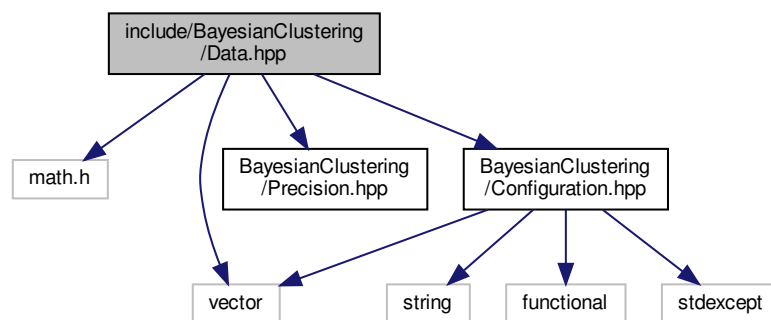


## Classes

- class [ScanConfiguration](#)  
Class for storing the scan configuration parameters.
- struct [ScanConfiguration::tBounds](#)  
A struct to store the bounds of a scan in either R or T.
- class [AuxConfiguration](#)  
Class for storing the auxilliary configuration parameters.

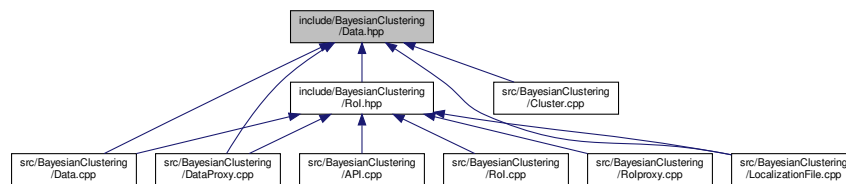
## 5.4 include/BayesianClustering/Data.hpp File Reference

```
#include <math.h>
#include <vector>
#include "BayesianClustering/Precision.hpp"
#include "BayesianClustering/Configuration.hpp"
Include dependency graph for Data.hpp:
```





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



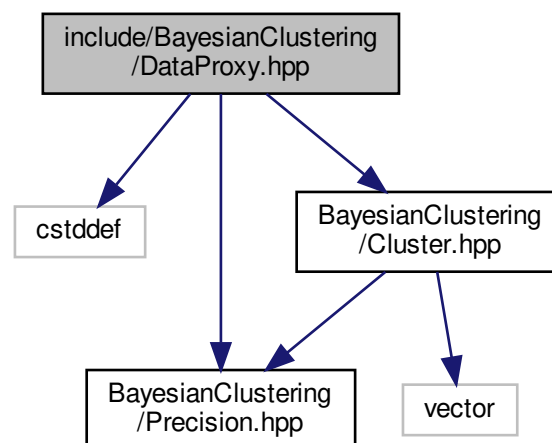
## Classes

- class [Data](#)

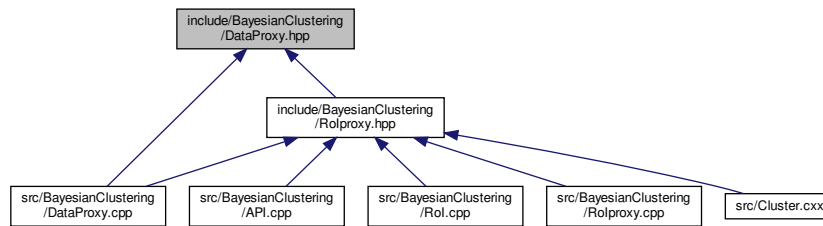
*A class to store the raw data-points.*

## 5.5 include/BayesianClustering/DataProxy.hpp File Reference

```
#include <cstdint>
#include "BayesianClustering/Precision.hpp"
#include "BayesianClustering/Cluster.hpp"
Include dependency graph for DataProxy.hpp:
```



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



## Classes

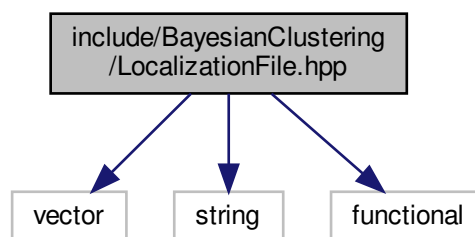
- class [DataProxy](#)

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

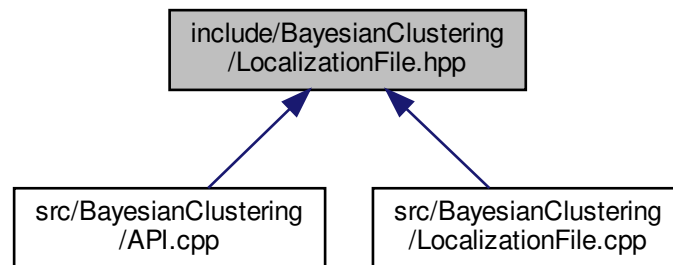
## 5.6 include/BayesianClustering/LocalizationFile.hpp File Reference

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

Include dependency graph for LocalizationFile.hpp:



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

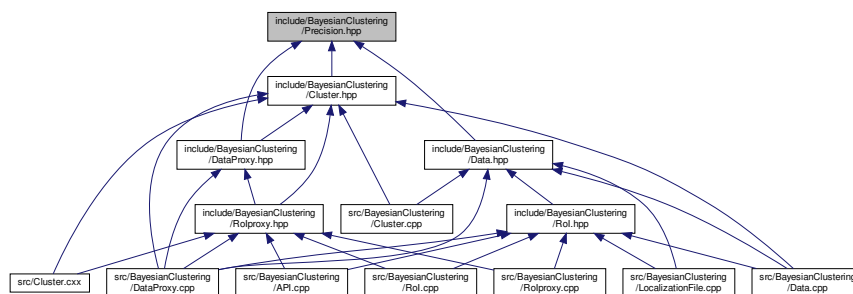


## Classes

- struct **ManualRol**  
*A struct for storing the manual Rols.*
- class **LocalizationFile**  
*A class to store the raw data-points.*

## 5.7 include/BayesianClustering/Precision.hpp File Reference

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

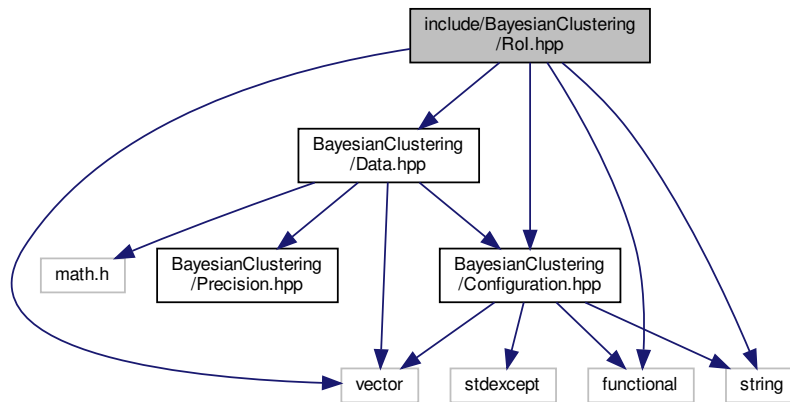


## 5.8 include/BayesianClustering/Rol.hpp File Reference

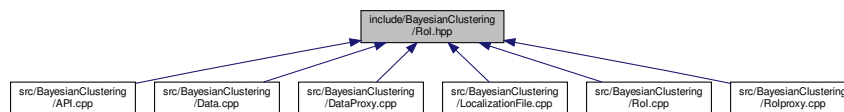
```
#include <vector>
#include <functional>
#include <string>
#include "BayesianClustering/Data.hpp"
```

```
#include "BayesianClustering/Configuration.hpp"
```

Include dependency graph for Rol.hpp:



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



## Classes

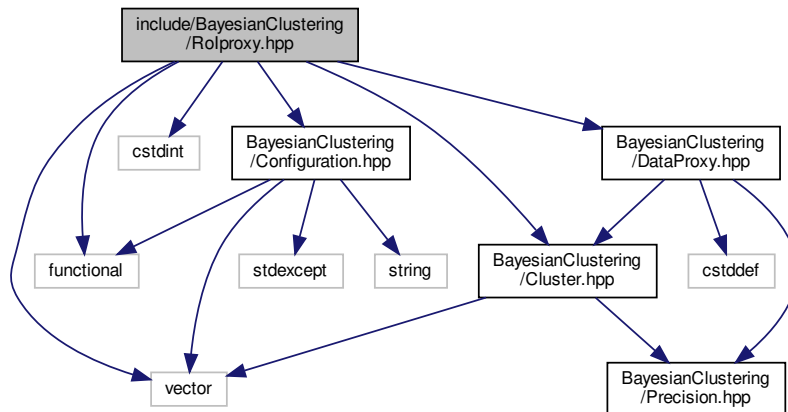
- class [Rol](#)

*A class which holds the raw [Rol](#) data and global parameters.*

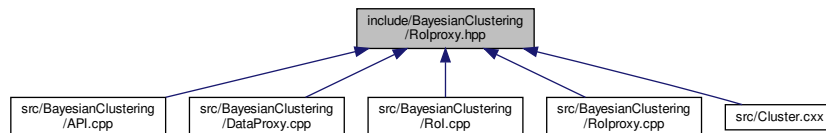
## 5.9 include/BayesianClustering/Rolproxy.hpp File Reference

```
#include <vector>
#include <functional>
#include <cstdint>
#include "BayesianClustering/Cluster.hpp"
#include "BayesianClustering/DataProxy.hpp"
#include "BayesianClustering/Configuration.hpp"
```

Include dependency graph for Rolproxy.hpp:



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



## Classes

- class [Rolproxy](#)

*A lightweight wrapper for the [Rol](#) to store clusters for a given scan.*

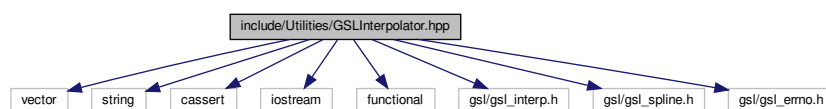
## 5.10 include/Utilities/GSLInterpolator.hpp File Reference

```

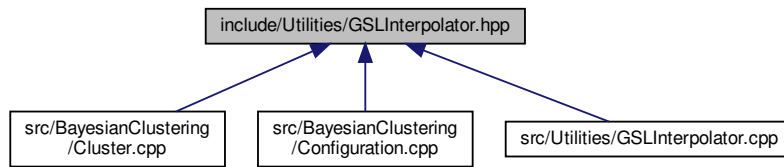
#include <vector>
#include <string>
#include <cassert>
#include <iostream>
#include <functional>
#include "gsl/gsl_interp.h"
#include "gsl/gsl_spline.h"
#include "gsl/gsl_errno.h"

```

Include dependency graph for GSLInterpolator.hpp:



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



## Classes

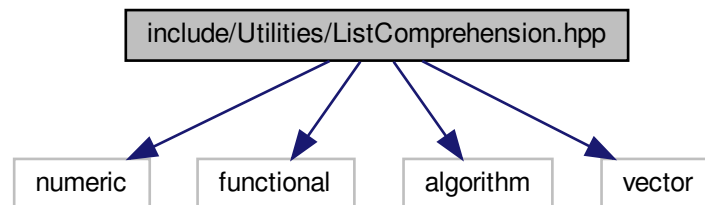
- class [GSLInterpolator](#)

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

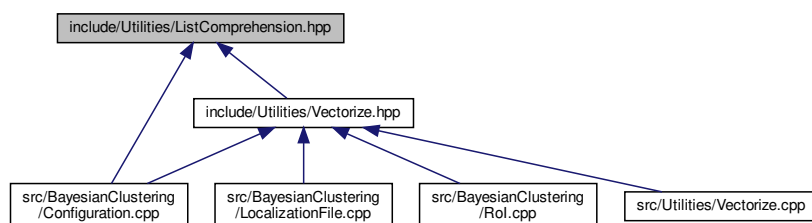
## 5.11 include/Utilities/ListComprehension.hpp File Reference

```
#include <numeric>
#include <functional>
#include <algorithm>
#include <vector>
```

Include dependency graph for ListComprehension.hpp:



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



## Functions

- `template<typename tContainer , typename tExpr , typename T = typename std::remove_reference<tContainer>::type::value_type, typename U = decltype( std::declval<tExpr>().operator()( std::declval<T>() ) )>`  
`std::enable_if< not std::is_same< U, void >::value, std::vector< U > >::type operator| (tExpr &&aExpr, tContainer &&aContainer)`

*Super nerd template magic emulating list comprehension for function with return type.*

- `template<typename tContainer , typename tExpr , typename T = typename std::remove_reference<tContainer>::type::value_type, typename U = decltype( std::declval<tExpr>().operator()( std::declval<T>() ) )>`  
`std::enable_if< std::is_same< U, void >::value, void >::type operator| (tExpr &&aExpr, tContainer &&aContainer)`

*Super nerd template magic emulating list comprehension for function with void return type.*

- `template<typename tContainer , typename tType , typename tContainerType = typename std::remove_reference<tContainer>::type::value_type>`  
`std::vector< tType > operator| (tType tContainerType::*aPtr, tContainer &&aContainer)`

*Return a container holding copies of a member-variable from each object in a container.*

- `std::vector< std::size_t > range (const std::size_t &N)`

*Emulate the python range function to generate a vector of ints.*

### 5.11.1 Function Documentation

#### 5.11.1.1 operator" | () [1/3]

```
template<typename tContainer , typename tExpr , typename T = typename std::remove_reference<tContainer>::type::value_type, typename U = decltype( std::declval<tExpr>().operator()( std::declval<T>() ) )>
std::enable_if< not std::is_same<U, void>::value, std::vector< U > >::type operator| (
    tExpr && aExpr,
    tContainer && aContainer ) [inline]
```

Super nerd template magic emulating list comprehension for function with return type.

#### Template Parameters

<i>tContainer</i>	A container type
<i>tExpr</i>	A function-call type
<i>T</i>	Template magic to determine the type of the data in the container
<i>U</i>	Template magic to determine the return-type of the function, given the type of the data in the container

#### Parameters

<i>aExpr</i>	A function-call to be applied to each element of the container
<i>aContainer</i>	A container holding the arguments to be fed to the expression

#### Returns

A vector of the results of the vectorized operations

Definition at line 20 of file ListComprehension.hpp.

### 5.11.1.2 operator" | () [2/3]

```
template<typename tContainer , typename tExpr , typename T = typename std::remove_reference<t↵
Container>::type::value_type, typename U = decltype( std::declval<tExpr>().operator() ( std↵
::declval<T>() ) )>
std::enable_if< std::is_same<U, void>::value, void >::type operator| (
    tExpr && aExpr,
    tContainer && aContainer ) [inline]
```

Super nerd template magic emulating list comprehension for function with void return type.

#### Template Parameters

<i>tContainer</i>	A container type
<i>tExpr</i>	A function-call type
<i>T</i>	Template magic to determine the type of the data in the container
<i>U</i>	Template magic to determine the return-type of the function, given the type of the data in the container

#### Parameters

<i>aExpr</i>	A function-call to be applied to each element of the container
<i>aContainer</i>	A container holding the arguments to be fed to the expression

#### Returns

Specialization of the vectorization for functions returning void

Definition at line 39 of file ListComprehension.hpp.

### 5.11.1.3 operator" | () [3/3]

```
template<typename tContainer , typename tType , typename tContainerType = typename std::remove↵
_reference<tContainer>::type::value_type>
std::vector< tType > operator| (
    tType tContainerType::* aPtr,
    tContainer && aContainer ) [inline]
```

Return a container holding copies of a member-variable from each object in a container.

#### Template Parameters

<i>tType</i>	A container type
<i>tContainerType</i>	Template magic to determine the type of the data in the container



## Parameters

<i>aPtr</i>	A pointer-to-member-variable to be applied to each element of the container
<i>aContainer</i>	A container holding the objects whose member variable is to be extracted

## Returns

A vector of the results of the vectorized operations

Definition at line 51 of file ListComprehension.hpp.

**5.11.1.4 range()**

```
std::vector< std::size_t > range (
    const std::size_t & N ) [inline]
```

Emulate the python range function to generate a vector of ints.

## Parameters

<i>N</i>	The number of elements
----------	------------------------

## Returns

A vector of ints

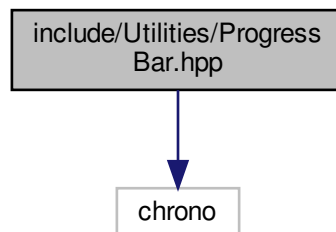
Definition at line 70 of file ListComprehension.hpp.

Referenced by LocalizationFile::LocalizationFile(), Rol::Preprocess(), Rol::ScanRT(), and ScanConfiguration::Set↔SigmaParameters().

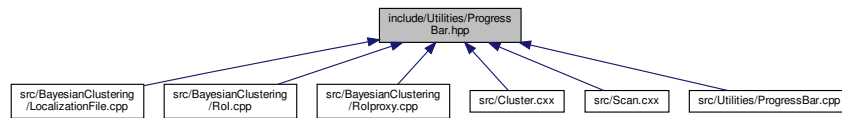
**5.12 include/Utilities/ProgressBar.hpp File Reference**

```
#include <chrono>
```

Include dependency graph for ProgressBar.hpp:



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



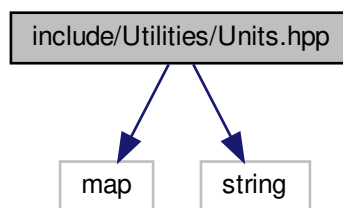
## Classes

- struct [ProgressBar](#)  
A utility progress-bar.
- struct [ProgressBar2](#)  
A utility code timer.

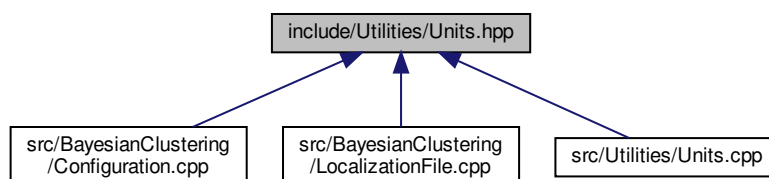
## 5.13 include/Utilities/Units.hpp File Reference

```
#include <map>
#include <string>
```

Include dependency graph for Units.hpp:



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



## Functions

- long double `operator""_nanometer` (long double aVal)  
*User-defined literals for nanometer quantities.*
- long double `operator""_nanometer` (unsigned long long aVal)  
*User-defined literals for nanometer quantities.*
- long double `operator""_micrometer` (long double aVal)  
*User-defined literals for micrometer quantities.*
- long double `operator""_micrometer` (unsigned long long aVal)  
*User-defined literals for micrometer quantities.*
- long double `StrToDist` (const std::string &aStr)  
*Convert a string representation to a distance.*

## Variables

- double `nanometer` = 1e-9  
*Define a constant for converting nanometers to meters.*
- double `micrometer` = 1e-6  
*Define a constant for converting micrometers to meters.*
- double `millimeter` = 1e-3  
*Define a constant for converting millimeters to meters.*
- double `meter` = 1e-0  
*Define a constant for converting meters to meters.*
- const std::map< std::string, double > `UnitMap`  
*A map for converting string representations of SI units to scaling factors.*

### 5.13.1 Function Documentation

#### 5.13.1.1 `operator""_micrometer()` [1/2]

```
long double operator""_micrometer (
    long double aVal )
```

User-defined literals for micrometer quantities.

#### Parameters

<code>aVal</code>	The specified value
-------------------	---------------------

#### Returns

The literal value

Definition at line 39 of file Units.hpp.

References `micrometer`.

### 5.13.1.2 `operator""_micrometer()` [2/2]

```
long double operator""_micrometer (
    unsigned long long aVal )
```

User-defined literals for micrometer quantities.

#### Parameters

<i>aVal</i>	The specified value
-------------	---------------------

#### Returns

The literal value

Definition at line 47 of file Units.hpp.

References micrometer.

### 5.13.1.3 `operator""_nanometer()` [1/2]

```
long double operator""_nanometer (
    long double aVal )
```

User-defined literals for nanometer quantities.

#### Parameters

<i>aVal</i>	The specified value
-------------	---------------------

#### Returns

The literal value

Definition at line 23 of file Units.hpp.

References nanometer.

### 5.13.1.4 `operator""_nanometer()` [2/2]

```
long double operator""_nanometer (
    unsigned long long aVal )
```

User-defined literals for nanometer quantities.

**Parameters**

<i>aVal</i>	The specified value
-------------	---------------------

**Returns**

The literal value

Definition at line 31 of file Units.hpp.

References nanometer.

**5.13.1.5 StrToDist()**

```
long double StrToDist (  
    const std::string & aStr )
```

Convert a string representation to a distance.

**Parameters**

<i>aStr</i>	A string representation of a distance
-------------	---------------------------------------

**Returns**

The literal value

Definition at line 12 of file Units.cpp.

References UnitMap.

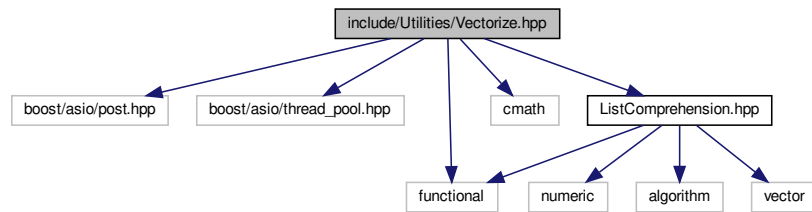
Referenced by ScanConfiguration::FromVector(), and AuxConfiguration::FromVector().

**5.14 include/Utilities/Vectorize.hpp File Reference**

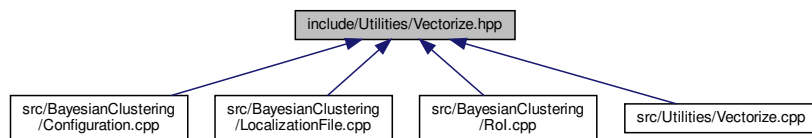
```
#include <boost/asio/post.hpp>  
#include <boost/asio/thread_pool.hpp>  
#include <functional>  
#include <cmath>
```

```
#include "ListComprehension.hpp"
```

Include dependency graph for Vectorize.hpp:



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



## Functions

- `template<typename tContainer , typename tExpr , typename tContainerType = typename std::remove_reference<tContainer>::type<-->::value_type>`  
`void operator|| (tExpr &&aExpr, tContainer &&aContainer)`  
*Syntactic sugar to allow you to interleave parallelize via operator.*
- `template<typename tContainer , typename tExpr , typename tContainerType = typename std::remove_reference<tContainer>::type<-->::value_type>`  
`void operator&& (tExpr &&aExpr, tContainer &&aContainer)`  
*Syntactic sugar to allow you to block parallelize via operator.*

## Variables

- `std::size_t Nthreads`  
*Utility variable for the concurrency.*

## 5.14.1 Function Documentation

### 5.14.1.1 operator&&()

```
template<typename tContainer , typename tExpr , typename tContainerType = typename std::remove_
_reference<tContainer>::type::value_type>
void operator&& (
    tExpr && aExpr,
    tContainer && aContainer ) [inline]
```

Syntactic sugar to allow you to block parallelize via operator.

## Template Parameters

<i>tContainer</i>	A container type
<i>tExpr</i>	A function-call type
<i>tContainerType</i>	A SFINAE hack to ensure that the container is a container

## Parameters

<i>aExpr</i>	A function-call to be applied to each element of the container
<i>aContainer</i>	A container holding the arguments to be distributed to the parallelized function calls

Definition at line 38 of file Vectorize.hpp.

References Nthreads.

## 5.14.1.2 operator" | " | ()

```
template<typename tContainer , typename tExpr , typename tContainerType = typename std::remove←
_reference<tContainer>::type::value_type>
void operator| (
    tExpr && aExpr,
    tContainer && aContainer ) [inline]
```

Syntactic sugar to allow you to interleave parallelize via operator.

## Template Parameters

<i>tContainer</i>	A container type
<i>tExpr</i>	A function-call type
<i>tContainerType</i>	A SFINAE hack to ensure that the container is a container

## Parameters

<i>aExpr</i>	A function-call to be applied to each element of the container
<i>aContainer</i>	A container holding the arguments to be distributed to the parallelized function calls

Definition at line 22 of file Vectorize.hpp.

References Nthreads.

## 5.14.2 Variable Documentation

### 5.14.2.1 Nthreads

```
std::size_t Nthreads [extern]
```

Utility variable for the concurrency.

Utility variable for the concurrency.

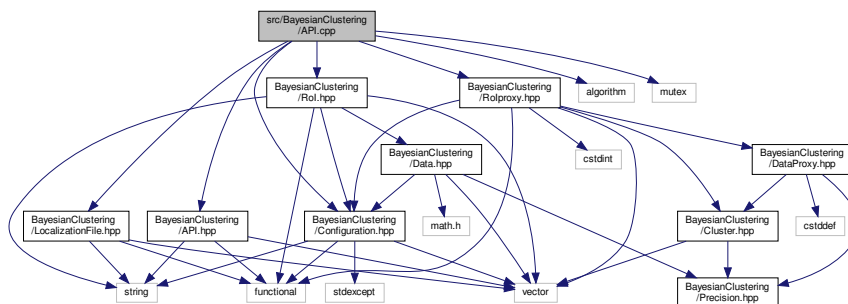
Definition at line 8 of file Vectorize.cpp.

Referenced by AuxConfiguration::FromVector(), LocalizationFile::LocalizationFile(), operator&&(), operator||(), and Rol::ScanRT().

## 5.15 src/BayesianClustering/API.cpp File Reference

```
#include "BayesianClustering/API.hpp"
#include "BayesianClustering/LocalizationFile.hpp"
#include "BayesianClustering/Configuration.hpp"
#include "BayesianClustering/RoI.hpp"
#include "BayesianClustering/RoIproxy.hpp"
#include <algorithm>
#include <mutex>
```

Include dependency graph for API.cpp:



## Functions

- void [ScanCallback\\_Json](#) (const std::vector< [ScanEntry](#) > &aVector, const std::string &aOutFile)  
*A callback to dump a scan to a JSON file.*
- void [AutoRoi\\_Scan\\_FullCallback](#) (const std::string &aInFile, const [ScanConfiguration](#) &aScanConfig, const std::function< void([RoIproxy](#) &, const double &, const double &) > &aCallback)  
*Automatically extract [RoI](#), run scan and apply a full call-back.*
- void [AutoRoi\\_Scan\\_SimpleCallback](#) (const std::string &aInFile, const [ScanConfiguration](#) &aScanConfig, const std::function< void(const std::vector< [ScanEntry](#) > &) > &aCallback)  
*Automatically extract [RoI](#), run scan and apply a simple call-back.*
- void [AutoRoi\\_Scan\\_ToJson](#) (const std::string &aInFile, const [ScanConfiguration](#) &aScanConfig, const std::function< void(const std::vector< [ScanEntry](#) > &, const std::string &aOutFile)  
*Automatically extract [RoI](#), run scan and dump to JSON file.*
- void [AutoRoi\\_Cluster\\_Callback](#) (const std::string &aInFile, const double &aR, const double &aT, const std::function< void([RoIproxy](#) &) > &aCallback)



*Automatically extract [Rol](#), clusterize and apply a call-back.*

- void [ManualRoi\\_Scan\\_FullCallback](#) (const std::string &alnFile, const [ManualRol](#) &aManualRol, const [ScanConfiguration](#) &aScanConfig, const std::function< void([Rolproxy](#) &, const double &, const double &) > &aCallback)

*Manually extract [Rol](#), run scan and apply a full call-back.*

- void [ManualRoi\\_Scan\\_SimpleCallback](#) (const std::string &alnFile, const [ManualRol](#) &aManualRol, const [ScanConfiguration](#) &aScanConfig, const std::function< void(const std::vector< [ScanEntry](#) > &) > &aCallback)

*Manually extract [Rol](#), run scan and apply a simple call-back.*

- void [ManualRoi\\_Scan\\_ToJson](#) (const std::string &alnFile, const [ManualRol](#) &aManualRol, const [ScanConfiguration](#) &aScanConfig, const std::string &aOutFile)

*Manually extract [Rol](#), run scan and dump to JSON file.*

- void [ManualRoi\\_Cluster\\_Callback](#) (const std::string &alnFile, const [ManualRol](#) &aManualRol, const double &aR, const double &aT, const std::function< void([Rolproxy](#) &) > &aCallback)

*Manually extract [Rol](#), clusterize and apply a call-back.*

## 5.15.1 Function Documentation

### 5.15.1.1 AutoRoi\_Cluster\_Callback()

```
void AutoRoi_Cluster_Callback (
    const std::string & aInFile,
    const double & aR,
    const double & aT,
    const std::function< void(RoIproxy &) > & aCallback )
```

Automatically extract [Rol](#), clusterize and apply a call-back.

#### Parameters

<i>aInFile</i>	The name of the localization file
<i>aR</i>	The R value of the clusterizer
<i>aT</i>	The T value of the clusterizer
<i>aCallback</i>	The callback to be applied

Definition at line 51 of file API.cpp.

References [Rol::Clusterize\(\)](#), and [LocalizationFile::ExtractRols\(\)](#).

Referenced by [main\(\)](#).

### 5.15.1.2 AutoRoi\_Scan\_FullCallback()

```
void AutoRoi_Scan_FullCallback (
    const std::string & aInFile,
```

```
const ScanConfiguration & aScanConfig,  
const std::function< void(RoIproxy &, const double &, const double &) > & a←  
Callback )
```

Automatically extract [RoI](#), run scan and apply a full call-back.

## Parameters

<i>aInFile</i>	The name of the localization file
<i>aScanConfig</i>	The configuration for the scan
<i>aCallback</i>	The full callback to be applied

Definition at line 32 of file API.cpp.

References LocalizationFile::ExtractRols(), and Rol::ScanRT().

Referenced by AutoRoi\_Scan\_SimpleCallback().

### 5.15.1.3 AutoRoi\_Scan\_SimpleCallback()

```
void AutoRoi_Scan_SimpleCallback (
    const std::string & aInFile,
    const ScanConfiguration & aScanConfig,
    const std::function< void(const std::vector< ScanEntry > &) > & aCallback )
```

Automatically extract Rol, run scan and apply a simple call-back.

## Parameters

<i>aInFile</i>	The name of the localization file
<i>aScanConfig</i>	The configuration for the scan
<i>aCallback</i>	The simple callback to be applied

Definition at line 37 of file API.cpp.

References AutoRoi\_Scan\_FullCallback(), and Rolproxy::mLogP.

Referenced by AutoRoi\_Scan\_ToJson().

### 5.15.1.4 AutoRoi\_Scan\_ToJson()

```
void AutoRoi_Scan_ToJson (
    const std::string & aInFile,
    const ScanConfiguration & aScanConfig,
    const std::string & aOutFile )
```

Automatically extract Rol, run scan and dump to JSON file.

## Parameters

<i>aInFile</i>	The name of the localization file
<i>aScanConfig</i>	The configuration for the scan
<i>aOutFile</i>	The name of the output JSON file

Definition at line 46 of file API.cpp.

References `AutoRoi_Scan_SimpleCallback()`, and `ScanCallback_Json()`.

Referenced by `main()`.

#### 5.15.1.5 ManualRoi\_Cluster\_Callback()

```
void ManualRoi_Cluster_Callback (
    const std::string & aInFile,
    const ManualRoi & aManualRoi,
    const double & aR,
    const double & aT,
    const std::function< void(RoiProxy &) > & aCallback )
```

Manually extract `Roi`, clusterize and apply a call-back.

##### Parameters

<i>aInFile</i>	The name of the localization file
<i>aManualRoi</i>	The manually-specified <code>Roi</code> window
<i>aR</i>	The R value of the clusterizer
<i>aT</i>	The T value of the clusterizer
<i>aCallback</i>	The callback to be applied

Definition at line 77 of file API.cpp.

References `Roi::Clusterize()`, and `LocalizationFile::ExtractRois()`.

#### 5.15.1.6 ManualRoi\_Scan\_FullCallback()

```
void ManualRoi_Scan_FullCallback (
    const std::string & aInFile,
    const ManualRoi & aManualRoi,
    const ScanConfiguration & aScanConfig,
    const std::function< void(RoiProxy &, const double &, const double &) > & a↵
    Callback )
```

Manually extract `Roi`, run scan and apply a full call-back.

##### Parameters

<i>aInFile</i>	The name of the localization file
<i>aManualRoi</i>	The manually-specified <code>Roi</code> window
<i>aScanConfig</i>	The configuration for the scan
<i>aCallback</i>	The full callback to be applied

Definition at line 58 of file API.cpp.

References LocalizationFile::ExtractRols(), and Rol::ScanRT().

Referenced by ManualRoi\_Scan\_SimpleCallback().

#### 5.15.1.7 ManualRoi\_Scan\_SimpleCallback()

```
void ManualRoi_Scan_SimpleCallback (
    const std::string & aInFile,
    const ManualRoi & aManualRoi,
    const ScanConfiguration & aScanConfig,
    const std::function< void(const std::vector< ScanEntry > &) > & aCallback )
```

Manually extract Rol, run scan and apply a simple call-back.

##### Parameters

<i>aInFile</i>	The name of the localization file
<i>aManualRoi</i>	The manually-specified Rol window
<i>aScanConfig</i>	The configuration for the scan
<i>aCallback</i>	The simple callback to be applied

Definition at line 63 of file API.cpp.

References ManualRoi\_Scan\_FullCallback(), and Rolproxy::mLogP.

Referenced by ManualRoi\_Scan\_ToJson().

#### 5.15.1.8 ManualRoi\_Scan\_ToJson()

```
void ManualRoi_Scan_ToJson (
    const std::string & aInFile,
    const ManualRoi & aManualRoi,
    const ScanConfiguration & aScanConfig,
    const std::string & aOutFile )
```

Manually extract Rol, run scan and dump to JSON file.

##### Parameters

<i>aInFile</i>	The name of the localization file
<i>aManualRoi</i>	The manually-specified Rol window
<i>aScanConfig</i>	The configuration for the scan
<i>aOutFile</i>	The name of the output JSON file

Definition at line 72 of file API.cpp.

References `ManualRoi_Scan_SimpleCallback()`, and `ScanCallback_Json()`.

### 5.15.1.9 ScanCallback\_Json()

```
void ScanCallback_Json (
    const std::vector< ScanEntry > & aVector,
    const std::string & aOutFile )
```

A callback to dump a scan to a JSON file.

#### Parameters

<i>aVector</i>	A vector of scan results
<i>aOutFile</i>	The name of the output JSON file

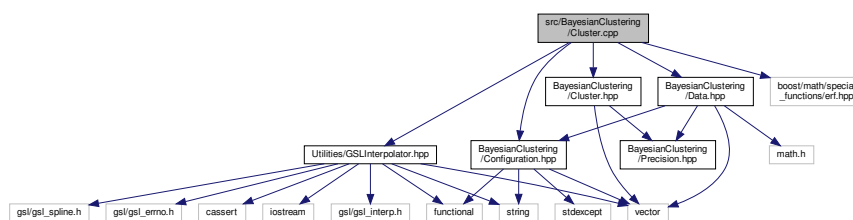
Definition at line 17 of file `API.cpp`.

Referenced by `AutoRoi_Scan_ToJson()`, and `ManualRoi_Scan_ToJson()`.

## 5.16 src/BayesianClustering/Cluster.cpp File Reference

```
#include "Utilities/GSLInterpolator.hpp"
#include <boost/math/special_functions/erf.hpp>
#include "BayesianClustering/Cluster.hpp"
#include "BayesianClustering/Data.hpp"
#include "BayesianClustering/Configuration.hpp"
```

Include dependency graph for `Cluster.cpp`:



## Functions

- double `normal_cdf` (const double &x, const double &sigma=1, const double &x0=0)

*Evaluate the Gaussian normal\_cdf at a given position Copied from the CERN ROOT implementaion, swap `ROOT::Math::erfc` and `ROOT::Math::erf` for the `boost::math` version.*

### 5.16.1 Function Documentation

## 5.16.1.1 normal\_cdf()

```
double normal_cdf (
    const double & x,
    const double & sigma = 1,
    const double & x0 = 0 ) [inline]
```

Evaluate the Gaussian normal\_cdf at a given position Copied from the CERN ROOT implementaion, swap ROOT↔::Math::erfc and ROOT::Math::erf for the boost::math version.

## Parameters

<i>x</i>	The position to evaluate the normal_cdf at
<i>sigma</i>	The standard-deviation of the Gaussian
<i>x0</i>	The mean of the Gaussian

## Returns

the value of the Gaussian normal\_cdf at x

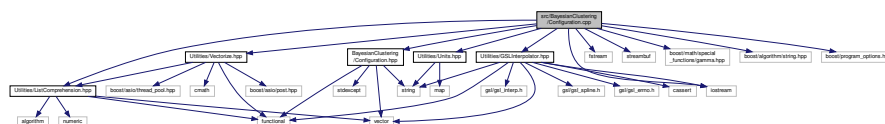
Definition at line 22 of file Cluster.cpp.

Referenced by Cluster::Parameter::alt\_log\_score(), and Cluster::Parameter::log\_score().

## 5.17 src/BayesianClustering/Configuration.cpp File Reference

```
#include "Utilities/GSLInterpolator.hpp"
#include "Utilities/ListComprehension.hpp"
#include "Utilities/Vectorize.hpp"
#include "Utilities/Units.hpp"
#include "BayesianClustering/Configuration.hpp"
#include <iostream>
#include <fstream>
#include <streambuf>
#include <boost/math/special_functions/gamma.hpp>
#include "boost/algorithm/string.hpp"
#include "boost/program_options.hpp"
```

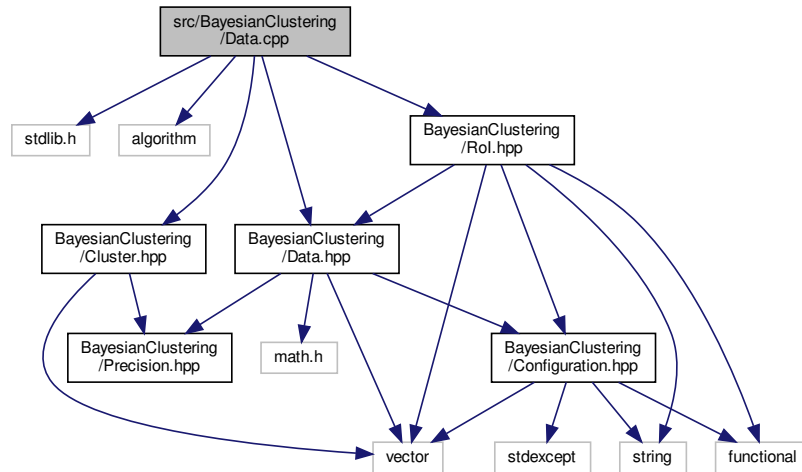
Include dependency graph for Configuration.cpp:



## 5.18 src/BayesianClustering/Data.cpp File Reference

```
#include <stdlib.h>
#include <algorithm>
```

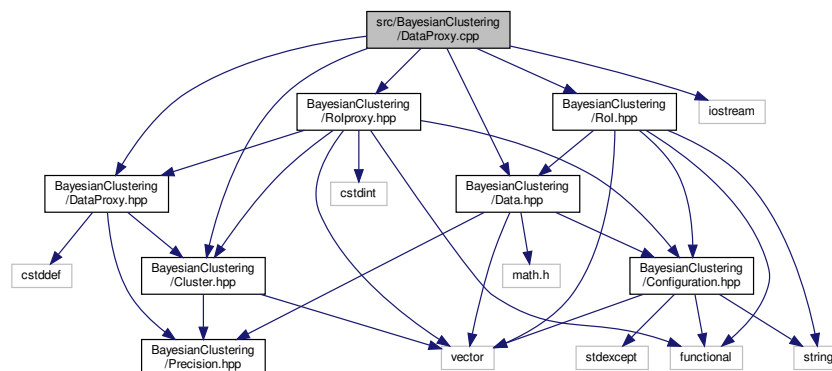
```
#include "BayesianClustering/Data.hpp"
#include "BayesianClustering/Cluster.hpp"
#include "BayesianClustering/RoI.hpp"
Include dependency graph for Data.cpp:
```



## 5.19 src/BayesianClustering/DataProxy.cpp File Reference

```
#include "BayesianClustering/DataProxy.hpp"
#include "BayesianClustering/Data.hpp"
#include "BayesianClustering/Cluster.hpp"
#include "BayesianClustering/RoIproxy.hpp"
#include "BayesianClustering/RoI.hpp"
#include <iostream>
```

Include dependency graph for DataProxy.cpp:



## Macros

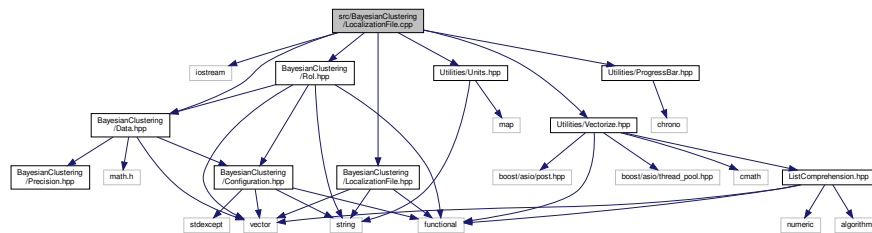
- `#define RECURSION_LIMIT 75000`  
The maximum depth for recursive clustering.



## 5.20 src/BayesianClustering/LocalizationFile.cpp File Reference

```
#include <iostream>
#include "BayesianClustering/LocalizationFile.hpp"
#include "BayesianClustering/Data.hpp"
#include "BayesianClustering/RoI.hpp"
#include "Utilities/ProgressBar.hpp"
#include "Utilities/Vectorize.hpp"
#include "Utilities/Units.hpp"
```

Include dependency graph for LocalizationFile.cpp:



### Typedefs

- typedef std::array< std::array< int, 512 >, 512 > **tArray**

*Typedef an array for histogramming a Localization File.*

### Functions

- void **\_\_LoadCSV\_\_** (const std::string &aFilename, std::vector< **Data** > &aData, const std::size\_t &aOffset, int aCount)  
*Multithreading handler for loading a chunk of data from CSV file.*
- void **\_\_RecursiveSearch\_\_** (**tArray** &aHist, const int &aRolid, const int &i, const int &j)  
*Recursively search histogram for continuously connected regions over threshold.*

### 5.20.1 Function Documentation

#### 5.20.1.1 \_\_LoadCSV\_\_()

```
void __LoadCSV__ (
    const std::string & aFilename,
    std::vector< Data > & aData,
    const std::size_t & aOffset,
    int aCount )
```

Multithreading handler for loading a chunk of data from CSV file.

## Parameters

<i>aFilename</i>	The name of the file to open
<i>aData</i>	A vector into which to fill data
<i>aOffset</i>	The offset into the file
<i>aCount</i>	The (approximate) number of bytes to be handled by this handler

Definition at line 22 of file LocalizationFile.cpp.

References nanometer.

Referenced by LocalizationFile::LocalizationFile().

### 5.20.1.2 \_\_RecursiveSearch\_\_()

```
void __RecursiveSearch__ (
    tArray & aHist,
    const int & aRoId,
    const int & i,
    const int & j )
```

Recursively search histogram for continuously connected regions over threshold.

## Parameters

<i>aHist</i>	The histogram being searched
<i>aRoId</i>	The id of the region being allocated
<i>i</i>	The horizontal index of the current cell in the histogram
<i>j</i>	The vertical index of the current cell in the histogram

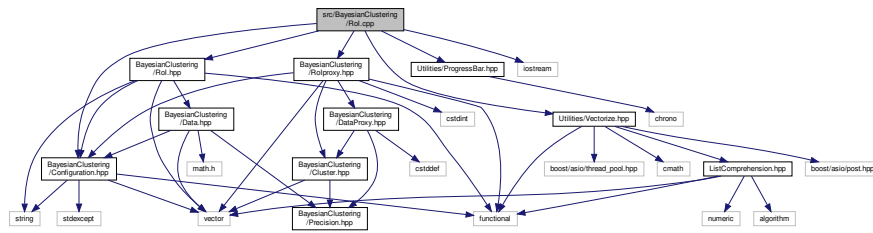
Definition at line 109 of file LocalizationFile.cpp.

Referenced by LocalizationFile::ExtractRols().

## 5.21 src/BayesianClustering/Rol.cpp File Reference

```
#include "BayesianClustering/RoI.hpp"
#include "BayesianClustering/RoIproxy.hpp"
#include "BayesianClustering/Configuration.hpp"
#include "Utilities/ProgressBar.hpp"
#include "Utilities/Vectorize.hpp"
#include <iostream>
```

Include dependency graph for Rol.cpp:



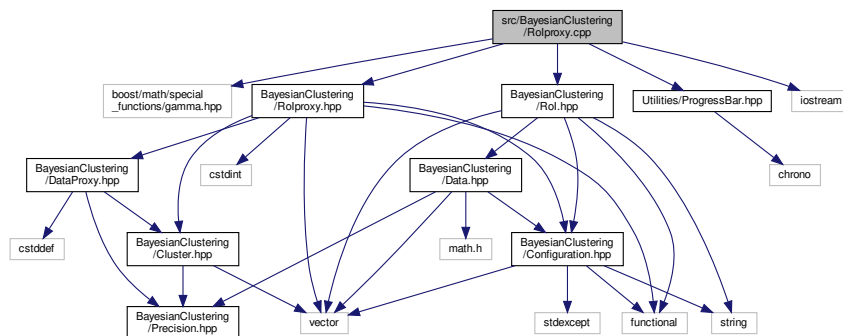
## 5.22 src/BayesianClustering/Rolproxy.cpp File Reference

```

#include <boost/math/special_functions/gamma.hpp>
#include "BayesianClustering/RoIproxy.hpp"
#include "BayesianClustering/RoI.hpp"
#include "Utilities/ProgressBar.hpp"
#include <iostream>

```

Include dependency graph for Rolproxy.cpp:



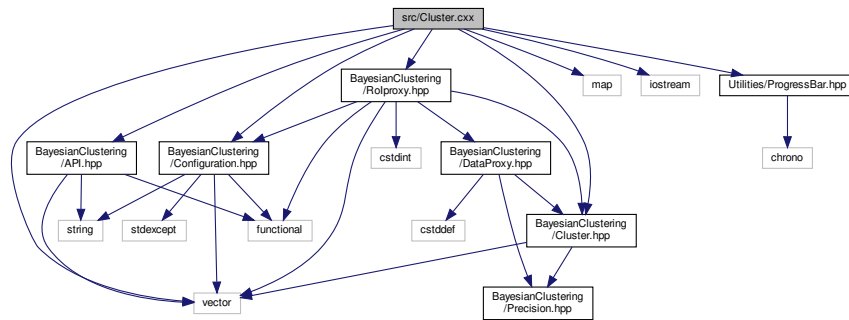
## 5.23 src/Cluster.cxx File Reference

```

#include "BayesianClustering/API.hpp"
#include "BayesianClustering/Cluster.hpp"
#include "BayesianClustering/RoIproxy.hpp"
#include "BayesianClustering/Configuration.hpp"
#include <map>
#include <vector>
#include <iostream>
#include "Utilities/ProgressBar.hpp"

```

Include dependency graph for Cluster.cxx:



## Functions

- void [ReportClusters](#) ([Rolproxy](#) &aProxy)  
*Callback to report clusters.*
- int [main](#) (int argc, char \*\*argv)  
*The main function.*

### 5.23.1 Function Documentation

#### 5.23.1.1 main()

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

The main function.

#### Parameters

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

#### Returns

The exit code

Definition at line 41 of file Cluster.cxx.

References [AutoRoi\\_Cluster\\_Callback\(\)](#), [AuxConfiguration::ClusterR\(\)](#), [AuxConfiguration::ClusterT\(\)](#), [AuxConfiguration::inputFile\(\)](#), and [ReportClusters\(\)](#).

### 5.23.1.2 ReportClusters()

```
void ReportClusters (
    RoIproxy & aProxy )
```

Callback to report clusters.

#### Parameters

<i>aProxy</i>	The <a href="#">RoI</a> to report
---------------	-----------------------------------

Definition at line 20 of file Cluster.cxx.

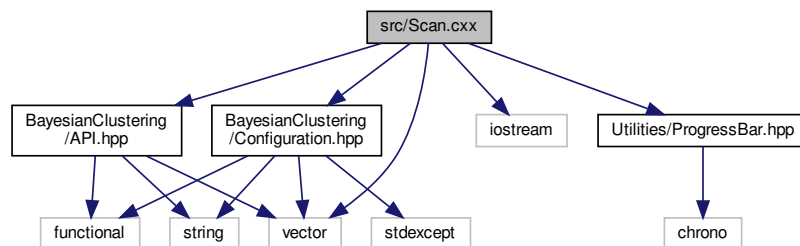
References [RoIproxy::mData](#).

Referenced by [main\(\)](#).

## 5.24 src/PythonBindings/PythonBindings.cpp File Reference

## 5.25 src/Scan.cxx File Reference

```
#include "BayesianClustering/API.hpp"
#include "BayesianClustering/Configuration.hpp"
#include <vector>
#include <iostream>
#include "Utilities/ProgressBar.hpp"
Include dependency graph for Scan.cxx:
```



## Functions

- [int main](#) (int argc, char \*\*argv)  
The main function.

### 5.25.1 Function Documentation

### 5.25.1.1 main()

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

The main function.

#### Parameters

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

#### Returns

The exit code

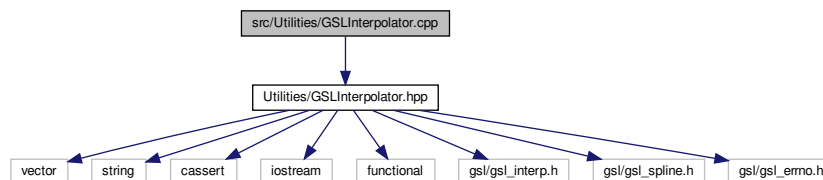
Definition at line 134 of file Scan.cxx.

References `AutoRoi_Scan_ToJson()`, `AuxConfiguration::configFile()`, `AuxConfiguration::inputFile()`, and `AuxConfiguration::outputFile()`.

## 5.26 src/Utilities/GSLInterpolator.cpp File Reference

```
#include "Utilities/GSLInterpolator.hpp"
```

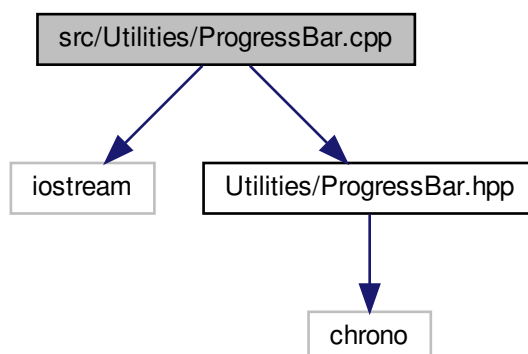
Include dependency graph for `GSLInterpolator.cpp`:



## 5.27 src/Utilities/ProgressBar.cpp File Reference

```
#include <iostream>
#include "Utilities/ProgressBar.hpp"
```

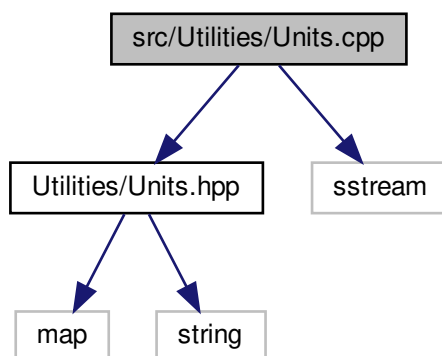
Include dependency graph for ProgressBar.cpp:



## 5.28 src/Utilities/Units.cpp File Reference

```
#include "Utilities/Units.hpp"  
#include <sstream>
```

Include dependency graph for Units.cpp:



## Functions

- long double [StrToDist](#) (const std::string &aStr)  
*Convert a string representation to a distance.*

## Variables

- `const std::map< std::string, double > UnitMap { {"nm",nanometer}, {"um",micrometer}, {"mm",millimeter}, {"m",meter} }`

*A map for converting string representations of SI units to scaling factors.*

## 5.28.1 Function Documentation

### 5.28.1.1 StrToDist()

```
long double StrToDist (
    const std::string & aStr )
```

Convert a string representation to a distance.

#### Parameters

<code>aStr</code>	A string representation of a distance
-------------------	---------------------------------------

#### Returns

The literal value

Definition at line 12 of file Units.cpp.

References UnitMap.

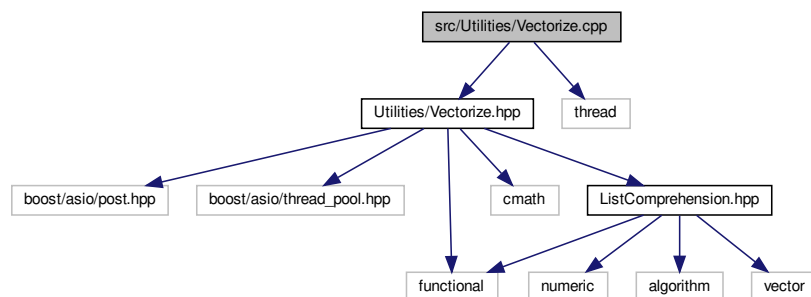
Referenced by ScanConfiguration::FromVector(), and AuxConfiguration::FromVector().

## 5.29 src/Utilities/Vectorize.cpp File Reference

```
#include "Utilities/Vectorize.hpp"
```

```
#include <thread>
```

Include dependency graph for Vectorize.cpp:





## Variables

- `std::size_t Nthreads = std::thread::hardware_concurrency()`

*The number of threads used, initialized to the number of hardware threads.*

### 5.29.1 Variable Documentation

#### 5.29.1.1 Nthreads

```
std::size_t Nthreads = std::thread::hardware_concurrency()
```

The number of threads used, initialized to the number of hardware threads.

Utility variable for the concurrency.

Definition at line 8 of file Vectorize.cpp.

Referenced by `AuxConfiguration::FromVector()`, `LocalizationFile::LocalizationFile()`, `operator&&()`, `operator||()`, and `Rol::ScanRT()`.



# Index

- [\\_\\_LoadCSV\\_\\_](#)
  - [LocalizationFile.cpp](#), [103](#)
  - [\\_\\_RecursiveSearch\\_\\_](#)
  - [LocalizationFile.cpp](#), [104](#)
- [alpha](#)
- [ScanConfiguration](#), [60](#)
- [alt\\_log\\_score](#)
- [Cluster::Parameter](#), [39](#)
- [API.cpp](#)
- [AutoRoi\\_Cluster\\_Callback](#), [95](#)
  - [AutoRoi\\_Scan\\_FullCallback](#), [95](#)
  - [AutoRoi\\_Scan\\_SimpleCallback](#), [97](#)
  - [AutoRoi\\_Scan\\_ToJson](#), [97](#)
  - [ManualRoi\\_Cluster\\_Callback](#), [98](#)
  - [ManualRoi\\_Scan\\_FullCallback](#), [98](#)
  - [ManualRoi\\_Scan\\_SimpleCallback](#), [99](#)
  - [ManualRoi\\_Scan\\_ToJson](#), [99](#)
  - [ScanCallback\\_Json](#), [100](#)
- [API.hpp](#)
- [AutoRoi\\_Cluster\\_Callback](#), [72](#)
  - [AutoRoi\\_Scan\\_FullCallback](#), [73](#)
  - [AutoRoi\\_Scan\\_SimpleCallback](#), [73](#)
  - [AutoRoi\\_Scan\\_ToJson](#), [74](#)
  - [ManualRoi\\_Cluster\\_Callback](#), [74](#)
  - [ManualRoi\\_Scan\\_FullCallback](#), [74](#)
  - [ManualRoi\\_Scan\\_SimpleCallback](#), [75](#)
  - [ManualRoi\\_Scan\\_ToJson](#), [75](#)
- [AutoRoi\\_Cluster\\_Callback](#)
- [API.cpp](#), [95](#)
  - [API.hpp](#), [72](#)
- [AutoRoi\\_Scan\\_FullCallback](#)
- [API.cpp](#), [95](#)
  - [API.hpp](#), [73](#)
- [AutoRoi\\_Scan\\_SimpleCallback](#)
- [API.cpp](#), [97](#)
  - [API.hpp](#), [73](#)
- [AutoRoi\\_Scan\\_ToJson](#)
- [API.cpp](#), [97](#)
  - [API.hpp](#), [74](#)
- [AuxConfiguration](#), [7](#)
- [AuxConfiguration](#), [8](#), [9](#)
  - [ClusterR](#), [9](#)
  - [ClusterT](#), [9](#)
  - [configFile](#), [9](#)
  - [FromVector](#), [10](#)
  - [inputFile](#), [10](#)
  - [outputFile](#), [10](#)
  - [SetConfigFile](#), [11](#)
  - [SetInputFile](#), [11](#)
  - [SetOutputFile](#), [12](#)
  - [SetValidate](#), [12](#)
  - [validate](#), [12](#)
- [CalculateLocalizationScore](#)
- [Data](#), [20](#)
- [CheckClusterization](#)
- [Rolproxy](#), [53](#)
- [Cluster](#), [13](#)
- [Cluster](#), [14](#), [15](#)
  - [GetParent](#), [15](#)
  - [operator+=](#), [16](#)
  - [operator=](#), [16](#)
  - [UpdateLogScore](#), [17](#)
- [Cluster.cpp](#)
- [normal\\_cdf](#), [100](#)
- [Cluster.cxx](#)
- [main](#), [106](#)
  - [ReportClusters](#), [106](#)
- [Cluster::Parameter](#), [39](#)
- [alt\\_log\\_score](#), [39](#)
  - [log\\_score](#), [40](#)
  - [operator+=](#), [40](#)
- [Clusterize](#)
- [DataProxy](#), [26](#), [27](#)
  - [Rol](#), [46](#)
  - [Rolproxy](#), [53](#)
- [ClusterR](#)
- [AuxConfiguration](#), [9](#)
- [ClusterT](#)
- [AuxConfiguration](#), [9](#)
- [configFile](#)
- [AuxConfiguration](#), [9](#)
- [Data](#), [17](#)
- [CalculateLocalizationScore](#), [20](#)
  - [Data](#), [19](#), [20](#)
  - [dPhi](#), [21](#)
  - [dR](#), [21](#)
  - [dR2](#), [21](#)
  - [operator<](#), [22](#)
  - [operator=](#), [22](#), [23](#)
  - [Preprocess](#), [23](#)
  - [PreprocessLocalizationScores](#), [23](#)
- [DataProxy](#), [24](#)
- [Clusterize](#), [26](#), [27](#)
  - [DataProxy](#), [25](#), [26](#)
  - [GetCluster](#), [27](#)
  - [operator=](#), [27](#), [28](#)
- [Deriv](#)

- GSLInterpolator, 31
- Deriv2
  - GSLInterpolator, 31
- dPhi
  - Data, 21
- dR
  - Data, 21
- dR2
  - Data, 21
- Eval
  - GSLInterpolator, 32
- Evaluate
  - GSLInterpolator, 32
- ExtractRols
  - LocalizationFile, 36, 37
- FromVector
  - AuxConfiguration, 10
  - ScanConfiguration, 60
- getArea
  - Rol, 46
- getCentreX
  - Rol, 47
- getCentreY
  - Rol, 47
- GetCluster
  - DataProxy, 27
- GetData
  - Rolproxy, 54
- GetParent
  - Cluster, 15
- getWidthX
  - Rol, 47
- getWidthY
  - Rol, 48
- GSLInterpolator, 28
  - Deriv, 31
  - Deriv2, 31
  - Eval, 32
  - Evaluate, 32
  - GSLInterpolator, 29, 30
  - Integ, 32
  - operator=, 33
  - SetData, 34
- include/BayesianClustering/API.hpp, 71
- include/BayesianClustering/Cluster.hpp, 76
- include/BayesianClustering/Configuration.hpp, 77
- include/BayesianClustering/Data.hpp, 78
- include/BayesianClustering/DataProxy.hpp, 79
- include/BayesianClustering/LocalizationFile.hpp, 80
- include/BayesianClustering/Precision.hpp, 81
- include/BayesianClustering/Rol.hpp, 81
- include/BayesianClustering/Rolproxy.hpp, 82
- include/Utilities/GSLInterpolator.hpp, 83
- include/Utilities/ListComprehension.hpp, 84
- include/Utilities/ProgressBar.hpp, 87
- include/Utilities/Units.hpp, 88
- include/Utilities/Vectorize.hpp, 91
- inputFile
  - AuxConfiguration, 10
- Integ
  - GSLInterpolator, 32
- ListComprehension.hpp
  - operator|, 85, 86
  - range, 87
- LocalizationFile, 35
  - ExtractRols, 36, 37
  - LocalizationFile, 35, 36
  - operator=, 37
- LocalizationFile.cpp
  - \_\_LoadCSV\_\_, 103
  - \_\_RecursiveSearch\_\_, 104
- log\_probability\_sigma
  - ScanConfiguration, 61
- log\_score
  - Cluster::Parameter, 40
- logAlpha
  - ScanConfiguration, 61
- logGammaAlpha
  - ScanConfiguration, 62
- logPb
  - ScanConfiguration, 62
- logPbDagger
  - ScanConfiguration, 62
- main
  - Cluster.cxx, 106
  - Scan.cxx, 107
- ManualRol, 38
- ManualRoi\_Cluster\_Callback
  - API.cpp, 98
  - API.hpp, 74
- ManualRoi\_Scan\_FullCallback
  - API.cpp, 98
  - API.hpp, 74
- ManualRoi\_Scan\_SimpleCallback
  - API.cpp, 99
  - API.hpp, 75
- ManualRoi\_Scan\_ToJson
  - API.cpp, 99
  - API.hpp, 75
- normal\_cdf
  - Cluster.cpp, 100
- Nthreads
  - Vectorize.cpp, 111
  - Vectorize.hpp, 93
- operator<
  - Data, 22
  - ScanEntry, 69
- operator++
  - ProgressBar, 42
  - ProgressBar2, 43

- operator+=
  - Cluster, 16
  - Cluster::Parameter, 40
- operator=
  - Cluster, 16
  - Data, 22, 23
  - DataProxy, 27, 28
  - GSLInterpolator, 33
  - LocalizationFile, 37
  - Rol, 48
  - Rolproxy, 54, 55
- operator&&
  - Vectorize.hpp, 92
- operator""\_micrometer
  - Units.hpp, 89
- operator""\_nanometer
  - Units.hpp, 90
- operator |
  - ListComprehension.hpp, 85, 86
- operator | |
  - Vectorize.hpp, 93
- outputFile
  - AuxConfiguration, 10
- Preprocess
  - Data, 23
  - Rol, 49
- PreprocessLocalizationScores
  - Data, 23
- probability\_sigma
  - ScanConfiguration, 63
- ProgressBar, 41
  - operator++, 42
  - ProgressBar, 41
- ProgressBar2, 42
  - operator++, 43
  - ProgressBar2, 43
- range
  - ListComprehension.hpp, 87
- Rbounds
  - ScanConfiguration, 64
- ReportClusters
  - Cluster.cxx, 106
- Rol, 44
  - Clusterize, 46
  - getArea, 46
  - getCentreX, 47
  - getCentreY, 47
  - getWidthX, 47
  - getWidthY, 48
  - operator=, 48
  - Preprocess, 49
  - Rol, 45, 46
  - ScanRT, 49
  - SetCentre, 50
  - SetWidth, 50
- Rolproxy, 51
  - CheckClusterization, 53
  - Clusterize, 53
  - GetData, 54
  - operator=, 54, 55
  - Rolproxy, 52, 53
  - ScanRT, 55
  - UpdateLogScore, 55
  - ValidateLogScore, 56
- Scan.cxx
  - main, 107
- ScanCallback\_Json
  - API.cpp, 100
- ScanConfiguration, 56
  - alpha, 60
  - FromVector, 60
  - log\_probability\_sigma, 61
  - logAlpha, 61
  - logGammaAlpha, 62
  - logPb, 62
  - logPbDagger, 62
  - probability\_sigma, 63
  - Rbounds, 64
  - ScanConfiguration, 59, 60
  - SetAlpha, 64
  - SetPb, 64
  - SetRBins, 65
  - SetSigmaParameters, 65
  - SetTBins, 66
  - sigmabins, 66
  - sigmabins2, 67
  - sigmacount, 67
  - sigmaspacing, 68
  - Tbounds, 68
- ScanConfiguration::tBounds, 70
- ScanEntry, 69
  - operator<, 69
- ScanRT
  - Rol, 49
  - Rolproxy, 55
- SetAlpha
  - ScanConfiguration, 64
- SetCentre
  - Rol, 50
- SetConfigFile
  - AuxConfiguration, 11
- SetData
  - GSLInterpolator, 34
- SetInputFile
  - AuxConfiguration, 11
- SetOutputFile
  - AuxConfiguration, 12
- SetPb
  - ScanConfiguration, 64
- SetRBins
  - ScanConfiguration, 65
- SetSigmaParameters
  - ScanConfiguration, 65
- SetTBins
  - ScanConfiguration, 66

- SetValidate
  - AuxConfiguration, [12](#)
- SetWidth
  - RoI, [50](#)
- sigmabins
  - ScanConfiguration, [66](#)
- sigmabins2
  - ScanConfiguration, [67](#)
- sigmacount
  - ScanConfiguration, [67](#)
- sigmaspacing
  - ScanConfiguration, [68](#)
- src/BayesianClustering/API.cpp, [94](#)
- src/BayesianClustering/Cluster.cpp, [100](#)
- src/BayesianClustering/Configuration.cpp, [101](#)
- src/BayesianClustering/Data.cpp, [101](#)
- src/BayesianClustering/DataProxy.cpp, [102](#)
- src/BayesianClustering/LocalizationFile.cpp, [103](#)
- src/BayesianClustering/RoI.cpp, [104](#)
- src/BayesianClustering/RoIproxy.cpp, [105](#)
- src/Cluster.cxx, [105](#)
- src/PythonBindings/PythonBindings.cpp, [107](#)
- src/Scan.cxx, [107](#)
- src/Utilities/GSLInterpolator.cpp, [108](#)
- src/Utilities/ProgressBar.cpp, [108](#)
- src/Utilities/Units.cpp, [109](#)
- src/Utilities/Vectorize.cpp, [110](#)
- StrToDist
  - Units.cpp, [110](#)
  - Units.hpp, [91](#)
- Tbounds
  - ScanConfiguration, [68](#)
- Units.cpp
  - StrToDist, [110](#)
- Units.hpp
  - operator""\_micrometer, [89](#)
  - operator""\_nanometer, [90](#)
  - StrToDist, [91](#)
- UpdateLogScore
  - Cluster, [17](#)
  - RoIproxy, [55](#)
- validate
  - AuxConfiguration, [12](#)
- ValidateLogScore
  - RoIproxy, [56](#)
- Vectorize.cpp
  - Nthreads, [111](#)
- Vectorize.hpp
  - Nthreads, [93](#)
  - operator&&, [92](#)
  - operator | |, [93](#)