

Bayesian Cluster Tool

Generated by Doxygen 1.9.1

Fri Jun 16 2023 11:41:44

1 Todo List	1
2 Class Index	3
2.1 Class List	3
3 File Index	5
3.1 File List	5
4 Class Documentation	7
4.1 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 ClusterWrapper Struct Reference	17
4.3.1 Detailed Description	18
4.3.2 Member Function Documentation	18
4.3.2.1 operator==()	18

4.4 Data Class Reference	19
4.4.1 Detailed Description	20
4.4.2 Constructor & Destructor Documentation	20
4.4.2.1 Data() [1/3]	20
4.4.2.2 Data() [2/3]	21
4.4.2.3 Data() [3/3]	21
4.4.3 Member Function Documentation	21
4.4.3.1 CalculateLocalizationScore()	21
4.4.3.2 dPhi()	22
4.4.3.3 dR()	22
4.4.3.4 dR2()	23
4.4.3.5 operator<()	23
4.4.3.6 operator=() [1/2]	23
4.4.3.7 operator=() [2/2]	24
4.4.3.8 Preprocess()	24
4.4.3.9 PreprocessLocalizationScores()	25
4.5 DataProxy Class Reference	25
4.5.1 Detailed Description	26
4.5.2 Constructor & Destructor Documentation	26
4.5.2.1 DataProxy() [1/3]	26
4.5.2.2 DataProxy() [2/3]	27
4.5.2.3 DataProxy() [3/3]	27
4.5.3 Member Function Documentation	27
4.5.3.1 Clusterize() [1/2]	27
4.5.3.2 Clusterize() [2/2]	28
4.5.3.3 GetCluster()	28
4.5.3.4 operator=() [1/2]	28
4.5.3.5 operator=() [2/2]	29
4.6 GSLInterpolator Class Reference	29
4.6.1 Detailed Description	30
4.6.2 Constructor & Destructor Documentation	30
4.6.2.1 GSLInterpolator() [1/4]	30
4.6.2.2 GSLInterpolator() [2/4]	31
4.6.2.3 GSLInterpolator() [3/4]	31
4.6.2.4 GSLInterpolator() [4/4]	31
4.6.3 Member Function Documentation	32
4.6.3.1 Deriv()	32
4.6.3.2 Deriv2()	32
4.6.3.3 Eval()	33
4.6.3.4 Evaluate()	33
4.6.3.5 Integ()	34
4.6.3.6 operator=() [1/2]	34

4.6.3.7 operator=() [2/2]	34
4.6.3.8 SetData() [1/2]	35
4.6.3.9 SetData() [2/2]	35
4.7 LocalizationFile Class Reference	36
4.7.1 Detailed Description	36
4.7.2 Constructor & Destructor Documentation	36
4.7.2.1 LocalizationFile() [1/3]	36
4.7.2.2 LocalizationFile() [2/3]	37
4.7.2.3 LocalizationFile() [3/3]	37
4.7.3 Member Function Documentation	37
4.7.3.1 ExtractRols() [1/2]	37
4.7.3.2 ExtractRols() [2/2]	38
4.7.3.3 operator=() [1/2]	38
4.7.3.4 operator=() [2/2]	39
4.8 ManualRol Struct Reference	39
4.8.1 Detailed Description	39
4.9 Cluster::Parameter Struct Reference	40
4.9.1 Detailed Description	40
4.9.2 Member Function Documentation	40
4.9.2.1 alt_log_score()	41
4.9.2.2 log_score()	41
4.9.2.3 operator+=()	41
4.10 ProgressBar Class Reference	42
4.10.1 Detailed Description	43
4.10.2 Constructor & Destructor Documentation	43
4.10.2.1 ProgressBar()	43
4.10.3 Member Function Documentation	43
4.10.3.1 operator++()	43
4.11 ProgressTimer Struct Reference	44
4.11.1 Detailed Description	44
4.11.2 Constructor & Destructor Documentation	44
4.11.2.1 ProgressTimer()	44
4.12 Rol Class Reference	45
4.12.1 Detailed Description	46
4.12.2 Constructor & Destructor Documentation	46
4.12.2.1 Rol() [1/3]	46
4.12.2.2 Rol() [2/3]	47
4.12.2.3 Rol() [3/3]	47
4.12.3 Member Function Documentation	47
4.12.3.1 Clusterize()	47
4.12.3.2 getArea()	48
4.12.3.3 getCentreX()	48

4.12.3.4 getCentreY()	48
4.12.3.5 getWidthX()	49
4.12.3.6 getWidthY()	49
4.12.3.7 operator=() [1/2]	49
4.12.3.8 operator=() [2/2]	50
4.12.3.9 Preprocess()	50
4.12.3.10 ScanRT()	50
4.12.3.11 SetCentre()	51
4.12.3.12 SetWidth()	51
4.13 Rolproxy Class Reference	52
4.13.1 Detailed Description	53
4.13.2 Constructor & Destructor Documentation	53
4.13.2.1 Rolproxy() [1/3]	53
4.13.2.2 Rolproxy() [2/3]	53
4.13.2.3 Rolproxy() [3/3]	54
4.13.3 Member Function Documentation	54
4.13.3.1 CheckClusterization()	54
4.13.3.2 Clusterize()	54
4.13.3.3 GetData()	55
4.13.3.4 operator=() [1/2]	55
4.13.3.5 operator=() [2/2]	56
4.13.3.6 ScanRT()	56
4.13.3.7 UpdateLogScore()	57
4.13.3.8 ValidateLogScore()	57
4.14 ScanConfiguration Class Reference	57
4.14.1 Detailed Description	60
4.14.2 Constructor & Destructor Documentation	60
4.14.2.1 ScanConfiguration() [1/4]	60
4.14.2.2 ScanConfiguration() [2/4]	61
4.14.2.3 ScanConfiguration() [3/4]	62
4.14.2.4 ScanConfiguration() [4/4]	62
4.14.3 Member Function Documentation	62
4.14.3.1 alpha()	62
4.14.3.2 FromVector()	62
4.14.3.3 log_probability_sigma() [1/2]	63
4.14.3.4 log_probability_sigma() [2/2]	63
4.14.3.5 logAlpha()	64
4.14.3.6 logGammaAlpha()	64
4.14.3.7 logPb()	64
4.14.3.8 logPbDagger()	65
4.14.3.9 operator=() [1/2]	65
4.14.3.10 operator=() [2/2]	65

4.14.3.11 probability_sigma() [1/2]	66
4.14.3.12 probability_sigma() [2/2]	66
4.14.3.13 Rbounds()	66
4.14.3.14 SetAlpha()	67
4.14.3.15 SetPb()	67
4.14.3.16 SetRBins()	67
4.14.3.17 SetSigmaParameters()	68
4.14.3.18 SetTBins()	68
4.14.3.19 sigmabins() [1/2]	69
4.14.3.20 sigmabins() [2/2]	69
4.14.3.21 sigmabins2() [1/2]	69
4.14.3.22 sigmabins2() [2/2]	70
4.14.3.23 sigmacount()	70
4.14.3.24 sigmaspacing()	71
4.14.3.25 Tbounds()	71
4.15 ScanEntry Struct Reference	71
4.15.1 Detailed Description	72
4.15.2 Member Function Documentation	72
4.15.2.1 operator<()	72
4.15.2.2 operator==(())	72
4.16 ScanConfiguration::tBounds Struct Reference	73
4.16.1 Detailed Description	73
5 File Documentation	75
5.1 include/BayesianClustering/API.hpp File Reference	75
5.1.1 Function Documentation	76
5.1.1.1 AutoRoi_Cluster_FullCallback()	77
5.1.1.2 AutoRoi_Cluster_SimpleCallback()	78
5.1.1.3 AutoRoi_Scan_FullCallback()	78
5.1.1.4 AutoRoi_Scan_SimpleCallback()	79
5.1.1.5 AutoRoi_Scan_ToJson()	79
5.1.1.6 ManualRoi_Cluster_FullCallback()	80
5.1.1.7 ManualRoi_Cluster_SimpleCallback()	80
5.1.1.8 ManualRoi_Scan_FullCallback()	81
5.1.1.9 ManualRoi_Scan_SimpleCallback()	81
5.1.1.10 ManualRoi_Scan_ToJson()	82
5.2 include/BayesianClustering/Cluster.hpp File Reference	82
5.3 include/BayesianClustering/Configuration.hpp File Reference	83
5.4 include/BayesianClustering/Data.hpp File Reference	84
5.5 include/BayesianClustering/DataProxy.hpp File Reference	85
5.6 include/BayesianClustering/LocalizationFile.hpp File Reference	86
5.7 include/BayesianClustering/Precision.hpp File Reference	87

5.8 include/BayesianClustering/Roi.hpp File Reference	87
5.9 include/BayesianClustering/RoiProxy.hpp File Reference	88
5.10 include/Utilities/GSLInterpolator.hpp File Reference	89
5.11 include/Utilities/ListComprehension.hpp File Reference	89
5.11.1 Function Documentation	90
5.11.1.1 operator" () [1/3]	91
5.11.1.2 operator" () [2/3]	91
5.11.1.3 operator" () [3/3]	92
5.11.1.4 range()	92
5.12 include/Utilities/MemoryMonitoring.hpp File Reference	93
5.12.1 Function Documentation	93
5.12.1.1 mem_usage()	93
5.13 include/Utilities/ProgressBar.hpp File Reference	94
5.14 include/Utilities/Units.hpp File Reference	95
5.14.1 Function Documentation	96
5.14.1.1 operator""_micrometer() [1/2]	96
5.14.1.2 operator""_micrometer() [2/2]	96
5.14.1.3 operator""_nanometer() [1/2]	97
5.14.1.4 operator""_nanometer() [2/2]	97
5.14.1.5 StrToDist()	98
5.15 include/Utilities/Vectorize.hpp File Reference	98
5.15.1 Function Documentation	99
5.15.1.1 operator&&()	99
5.15.1.2 operator" " ()	100
5.15.2 Variable Documentation	100
5.15.2.1 Nthreads	100
5.16 src/BayesianClustering/API.cpp File Reference	101
5.16.1 Function Documentation	102
5.16.1.1 _FullClusterToSimpleCluster_()	102
5.16.1.2 _FullScanToSimpleScan_()	102
5.16.1.3 _ScanCallback_Json_()	103
5.16.1.4 AutoRoi_Cluster_FullCallback()	103
5.16.1.5 AutoRoi_Cluster_SimpleCallback()	104
5.16.1.6 AutoRoi_Scan_FullCallback()	104
5.16.1.7 AutoRoi_Scan_SimpleCallback()	105
5.16.1.8 AutoRoi_Scan_ToJson()	105
5.16.1.9 ManualRoi_Cluster_FullCallback()	105
5.16.1.10 ManualRoi_Cluster_SimpleCallback()	106
5.16.1.11 ManualRoi_Scan_FullCallback()	106
5.16.1.12 ManualRoi_Scan_SimpleCallback()	107
5.16.1.13 ManualRoi_Scan_ToJson()	107
5.17 src/BayesianClustering/Cluster.cpp File Reference	108

5.17.1 Function Documentation	108
5.17.1.1 normal_cdf()	108
5.18 src/BayesianClustering/Configuration.cpp File Reference	109
5.19 src/BayesianClustering/Data.cpp File Reference	109
5.20 src/BayesianClustering/DataProxy.cpp File Reference	110
5.21 src/BayesianClustering/LocalizationFile.cpp File Reference	111
5.21.1 Function Documentation	111
5.21.1.1 __LoadCSV__()	111
5.21.1.2 __RecursiveSearch__()	112
5.22 src/BayesianClustering/Rol.cpp File Reference	112
5.23 src/BayesianClustering/Rolproxy.cpp File Reference	113
5.24 src/Cluster.cxx File Reference	113
5.24.1 Function Documentation	114
5.24.1.1 main()	114
5.24.1.2 ReportClusters()	115
5.25 src/PythonBindings/PythonBindings.cpp File Reference	115
5.25.1 Detailed Description	116
5.25.2 Macro Definition Documentation	117
5.25.2.1 ADAPT_CALLBACK_CONSTRUCTOR	117
5.25.2.2 ADAPTED_FN	117
5.25.2.3 EXPOSE_STRUCT	117
5.25.2.4 EXPOSE_VECTOR	118
5.25.2.5 FN	118
5.25.2.6 STRUCT_ARG	118
5.26 src/Utilities/GSLInterpolator.cpp File Reference	119
5.27 src/Utilities/ProgressBar.cpp File Reference	119
5.28 src/Utilities/Units.cpp File Reference	120
5.28.1 Function Documentation	120
5.28.1.1 StrToDist()	120
5.29 src/Utilities/Vectorize.cpp File Reference	121
5.29.1 Variable Documentation	121
5.29.1.1 Nthreads	121
Index	123

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:

AuxConfiguration	Class for storing the auxilliary configuration parameters	7
Cluster	A class representing a cluster	13
ClusterWrapper	A struct for storing a result of an individual scan configuration	17
Data	A class to store the raw data-points	19
DataProxy	A light-weight proxy for the raw data-points	25
GSLInterpolator	A utility wrapper around the GSL interpolator to give it a clean C++ interface	29
LocalizationFile	A class to store the raw data-points	36
ManualRol	A struct for storing the manual Rols	39
Cluster::Parameter	A struct representing the cluster parameters	40
ProgressBar	A utility progress-bar	42
ProgressTimer	A utility code timer	44
Rol	A class which holds the raw Rol data and global parameters	45
Rolproxy	A lightweight wrapper for the Rol to store clusters for a given scan	52
ScanConfiguration	Class for storing the scan configuration parameters	57
ScanEntry	A struct for storing a result of an individual scan configuration	71
ScanConfiguration::tBounds	A struct to store the bounds of a scan in either R or T	73

Chapter 3

File Index

3.1 File List

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

include/BayesianClustering/API.hpp	75
include/BayesianClustering/Cluster.hpp	82
include/BayesianClustering/Configuration.hpp	83
include/BayesianClustering/Data.hpp	84
include/BayesianClustering/DataProxy.hpp	85
include/BayesianClustering/LocalizationFile.hpp	86
include/BayesianClustering/Precision.hpp	87
include/BayesianClustering/Rol.hpp	87
include/BayesianClustering/Rolproxy.hpp	88
include/Utilities/GSLInterpolator.hpp	89
include/Utilities/ListComprehension.hpp	89
include/Utilities/MemoryMonitoring.hpp	93
include/Utilities/ProgressBar.hpp	94
include/Utilities/Units.hpp	95
include/Utilities/Vectorize.hpp	98
src/Cluster.cxx	113
src/Scan.cxx	??
src/BayesianClustering/API.cpp	101
src/BayesianClustering/Cluster.cpp	108
src/BayesianClustering/Configuration.cpp	109
src/BayesianClustering/Data.cpp	109
src/BayesianClustering/DataProxy.cpp	110
src/BayesianClustering/LocalizationFile.cpp	111
src/BayesianClustering/Rol.cpp	112
src/BayesianClustering/Rolproxy.cpp	113
src/PythonBindings/PythonBindings.cpp	
Self-contained sourcefile for producing python-bindings	115
src/Utilities/GSLInterpolator.cpp	119
src/Utilities/ProgressBar.cpp	119
src/Utilities/Units.cpp	120
src/Utilities/Vectorize.cpp	121

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 263 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 214 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 223 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 323 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 330 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 316 of file Configuration.hpp.

References mConfigFile.

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 262 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 302 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 309 of file Configuration.hpp.

References mOutputFile.

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 250 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 237 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 243 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 231 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 295 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 > &aSigmas2)
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](#) ()
Default destructor.
- [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>aSigmbins2</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 165 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 155 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 124 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 ClusterWrapper Struct Reference

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

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

Public Member Functions

- bool `operator==` (const ClusterWrapper &aOther)
Equality operator required by boost python.

Public Attributes

- `std::size_t` [localizations](#)
The number of localizations in the cluster.
- `long double` [area](#)
The area of the spanning convex hull.
- `long double` [perimeter](#)
The perimeter of the spanning convex hull.
- `double` [centroid_x](#)
The x-position of the centroid.
- `double` [centroid_y](#)
The y-position of the centroid.

4.3.1 Detailed Description

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

Definition at line 44 of file `API.hpp`.

4.3.2 Member Function Documentation

4.3.2.1 `operator==()`

```
bool ClusterWrapper::operator== (
    const ClusterWrapper & aOther ) [inline]
```

Equality operator required by boost python.

Returns

Whether we are equal to the other

Parameters

<i>aOther</i>	Another ClusterWrapper to compare against
---------------	---

Definition at line 54 of file `API.hpp`.

References `centroid_x`, and `centroid_y`.

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

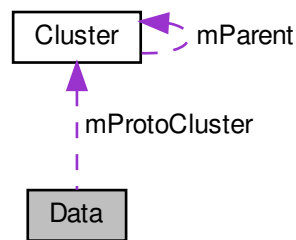
- `include/BayesianClustering/`[API.hpp](#)

4.4 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, [ProgressBar](#) &aProgressBar)
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.

4.4.1 Detailed Description

A class to store the raw data-points.

Definition at line 17 of file Data.hpp.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 Data() [1/3]

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

Constructor.

Parameters

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

Definition at line 14 of file Data.cpp.

4.4.2.2 Data() [2/3]

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

Deleted copy constructor.

Parameters

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

4.4.2.3 Data() [3/3]

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

Default move constructor.

Parameters

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

4.4.3 Member Function Documentation**4.4.3.1 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 106 of file Data.cpp.

References mNeighbours.

4.4.3.2 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 71 of file Data.hpp.

References phi.

4.4.3.3 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 63 of file Data.hpp.

References dR2().

4.4.3.4 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 54 of file Data.hpp.

References x, and y.

Referenced by dR().

4.4.3.5 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 46 of file Data.hpp.

References r.

4.4.3.6 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
---------------	--------------------

4.4.3.7 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
---------------	--------------------

4.4.3.8 Preprocess()

```
void Data::Preprocess (
    std::vector< Data > & aData,
    const std::size_t & aIndex,
    const double & aMax2R,
    const double & aMax2R2,
    const std::vector< double > & aSigmabins2,
    ProgressBar & aProgressBar )
```

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
<i>aProgressBar</i>	The progress bar to update

Definition at line 28 of file Data.cpp.

4.4.3.9 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 77 of file Data.cpp.

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

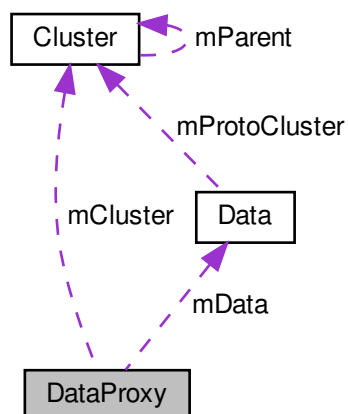
- include/BayesianClustering/Data.hpp
- src/BayesianClustering/Data.cpp

4.5 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.

4.5.1 Detailed Description

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

Definition at line 18 of file DataProxy.hpp.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 DataProxy() [1/3]

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

Default constructor.

Parameters

aData	The data-point for which this is the proxy
-----------------------	--

Definition at line 17 of file DataProxy.cpp.

4.5.2.2 DataProxy() [2/3]

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

Deleted copy constructor.

Parameters

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

4.5.2.3 DataProxy() [3/3]

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

Default move constructor.

Parameters

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

4.5.3 Member Function Documentation

4.5.3.1 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()`.

4.5.3.2 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`.

4.5.3.3 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()`.

4.5.3.4 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
---------------	--------------------

4.5.3.5 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

4.6 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.

4.6.1 Detailed Description

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

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

4.6.2 Constructor & Destructor Documentation

4.6.2.1 `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.

4.6.2.2 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().

4.6.2.3 GSLInterpolator() [3/4]

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

Deleted copy constructor.

Parameters

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

4.6.2.4 GSLInterpolator() [4/4]

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

Default move constructor.

Parameters

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

4.6.3 Member Function Documentation

4.6.3.1 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.

4.6.3.2 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.

4.6.3.3 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().

4.6.3.4 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().

4.6.3.5 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.

4.6.3.6 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
---------------	--------------------

4.6.3.7 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
---------------	--------------------

4.6.3.8 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().

4.6.3.9 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

4.7 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.

4.7.1 Detailed Description

A class to store the raw data-points.

Definition at line 25 of file LocalizationFile.hpp.

4.7.2 Constructor & Destructor Documentation

4.7.2.1 LocalizationFile() [1/3]

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

Constructor.

Parameters

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

Definition at line 68 of file LocalizationFile.cpp.

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

4.7.2.2 LocalizationFile() [2/3]

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

Deleted copy constructor.

Parameters

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

4.7.2.3 LocalizationFile() [3/3]

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

Default move constructor.

Parameters

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

4.7.3 Member Function Documentation

4.7.3.1 ExtractRols() [1/2]

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

Manually extract an [RoI](#).

Parameters

<i>aRoi</i>	The manual Roi window
<i>aCallback</i>	A handler for each Roi found

Definition at line 268 of file LocalizationFile.cpp.

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

4.7.3.2 ExtractRols() [2/2]

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

Automatically extract the Rols.

Parameters

<i>aCallback</i>	A handler for each Roi found
------------------	--

Definition at line 119 of file LocalizationFile.cpp.

References [__RecursiveSearch__\(\)](#), [mData](#), [Roi::SetCentre\(\)](#), and [Roi::SetWidth\(\)](#).

Referenced by [AutoRoi_Cluster_FullCallback\(\)](#), [AutoRoi_Scan_FullCallback\(\)](#), [AutoRoi_Scan_SimpleCallback\(\)](#), [ManualRoi_Cluster_FullCallback\(\)](#), [ManualRoi_Scan_FullCallback\(\)](#), and [ManualRoi_Scan_SimpleCallback\(\)](#).

4.7.3.3 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
---------------	--------------------

4.7.3.4 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](#)

4.8 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](#).

4.8.1 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](#)

4.9 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.

4.9.1 Detailed Description

A struct representing the cluster parameters.

Definition at line 21 of file Cluster.hpp.

4.9.2 Member Function Documentation

4.9.2.1 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()`.

4.9.2.2 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()`.

4.9.2.3 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](#)

4.10 ProgressBar Class 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.

Private Member Functions

- void [print](#) ()
Update the screen.

Private 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.
- std::mutex [mMutex](#)
A mutex for multi-threaded updates.
- std::string [mLabel](#)
The label for the start of the line.
- std::size_t [mPercent](#)
The current progress.

4.10.1 Detailed Description

A utility progress-bar.

Definition at line 8 of file ProgressBar.hpp.

4.10.2 Constructor & Destructor Documentation

4.10.2.1 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 8 of file ProgressBar.cpp.

References mLabel, and print().

4.10.3 Member Function Documentation

4.10.3.1 operator++()

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

Prefix increment.

Parameters

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

Definition at line 39 of file ProgressBar.cpp.

References operator++().

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

- [include/Utilities/ProgressBar.hpp](#)
- [src/Utilities/ProgressBar.cpp](#)

4.11 ProgressTimer Struct Reference

A utility code timer.

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

Public Member Functions

- [ProgressTimer](#) (const std::string &aLabel)
Constructor.
- virtual [~ProgressTimer](#) ()
Destructor.

Public Attributes

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

4.11.1 Detailed Description

A utility code timer.

Definition at line 49 of file `ProgressBar.hpp`.

4.11.2 Constructor & Destructor Documentation

4.11.2.1 ProgressTimer()

```
ProgressTimer::ProgressTimer (
    const std::string & aLabel )
```

Constructor.

Parameters

<i>aLabel</i>	A description of the task being timed
---------------	---------------------------------------

Definition at line 55 of file `ProgressBar.cpp`.

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

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

4.12 RoI Class Reference

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

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

Public Member Functions

- [RoI](#) (std::vector< [Data](#) > &&aData, const std::size_t &ald=0)
Default Constructor.
- [RoI](#) (const [RoI](#) &aOther)=delete
Deleted copy constructor.
- [RoI](#) & operator= (const [RoI](#) &aOther)=delete
Deleted assignment operator.
- [RoI](#) ([RoI](#) &&aOther)=default
Default move constructor.
- [~RoI](#) ()
Default destructor.
- [RoI](#) & operator= ([RoI](#) &&aOther)=default
Default move-assignment constructor.
- void [Preprocess](#) (const double &aMaxR, const std::vector< double > &aSigmabins2)
All the necessary pre-processing to get the [RoI](#) ready for an RT-scan.
- void [ScanRT](#) (const [ScanConfiguration](#) &aScanConfig, const std::function< void([RoIproxy](#) &, const double &, const double &) > &aCallback)
Run the scan.
- void [Clusterize](#) (const double &R, const double &T, const std::function< void([RoIproxy](#) &) > &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 [RoI](#) 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::size_t mId`
A unique identifier.
- `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.

4.12.1 Detailed Description

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

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

4.12.2 Constructor & Destructor Documentation

4.12.2.1 `RoI()` [1/3]

```
RoI::RoI (
    std::vector< Data > && aData,
    const std::size_t & aId = 0 )
```

Default Constructor.

Parameters

<code>aData</code>	The set of data-points in the RoI
<code>aId</code>	A unique identifier

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

References `mData`.

4.12.2.2 RoI() [2/3]

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

Deleted copy constructor.

Parameters

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

4.12.2.3 RoI() [3/3]

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

Default move constructor.

Parameters

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

4.12.3 Member Function Documentation

4.12.3.1 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 58 of file RoI.cpp.

References `RoIproxy::Clusterize()`, and `Preprocess()`.

Referenced by `AutoRoi_Cluster_FullCallback()`, and `ManualRoi_Cluster_FullCallback()`.

4.12.3.2 `getArea()`

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

Getter for the height of the ROI window.

Returns

The height of the ROI window

Definition at line 102 of file `Rol.hpp`.

References `mArea`.

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

4.12.3.3 `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 75 of file `Rol.hpp`.

References `mPhysicalCentreX`.

4.12.3.4 `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 81 of file `Rol.hpp`.

References `mPhysicalCentreY`.

4.12.3.5 getWidthX()

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

Getter for the width of the ROI window.

Returns

The width of the ROI window

Definition at line 88 of file RoI.hpp.

References `mWidthX`.

4.12.3.6 getWidthY()

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

Getter for the height of the ROI window.

Returns

The height of the ROI window

Definition at line 95 of file RoI.hpp.

References `mWidthY`.

4.12.3.7 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
---------------	--------------------

4.12.3.8 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
---------------	--------------------

4.12.3.9 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 35 of file RoI.cpp.

References mData, and range().

Referenced by Clusterize(), and ScanRT().

4.12.3.10 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 44 of file RoI.cpp.

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

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

4.12.3.11 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 70 of file RoI.cpp.

References `mPhysicalCentreX`, and `mPhysicalCentreY`.

Referenced by `LocalizationFile::ExtractRols()`.

4.12.3.12 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 77 of file RoI.cpp.

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

Referenced by `LocalizationFile::ExtractRols()`.

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

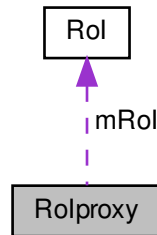
- `include/BayesianClustering/RoI.hpp`
- `src/BayesianClustering/RoI.cpp`

4.13 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.
- [~Rolproxy](#) ()
Default destructor.
- 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, [ProgressBar](#) &aProgressBar, 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 [Rol](#) 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 Rol & mRol`
The underlying [Rol](#) this is a proxy to.

4.13.1 Detailed Description

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

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

4.13.2 Constructor & Destructor Documentation

4.13.2.1 Rolproxy() [1/3]

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

Default constructor.

Parameters

<i>aRol</i>	An Rol for which this is a lightweight proxy
-------------	--

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

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

4.13.2.2 Rolproxy() [2/3]

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

Deleted copy constructor.

Parameters

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

4.13.2.3 Rolproxy() [3/3]

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

Default move constructor.

Parameters

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

4.13.3 Member Function Documentation**4.13.3.1 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 34 of file Rolproxy.cpp.

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

4.13.3.2 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 139 of file Rolproxy.cpp.

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

Referenced by Rol::Clusterize().

4.13.3.3 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 74 of file Rolproxy.hpp.

References mData.

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

4.13.3.4 operator=() [1/2]

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

Deleted assignment operator.

Returns

Reference to this, for chaining calls

Parameters

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

4.13.3.5 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
---------------	--------------------

4.13.3.6 ScanRT()

```
void RoIproxy::ScanRT (
    const ScanConfiguration & aScanConfig,
    const std::function< void(RoIproxy &, const double &, const double &) > & aCallback,
    ProgressBar & aProgressBar,
    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>aProgressBar</i>	The progress bar to update
<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 104 of file RoIproxy.cpp.

4.13.3.7 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 219 of file Rolproxy.cpp.

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

4.13.3.8 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 160 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:

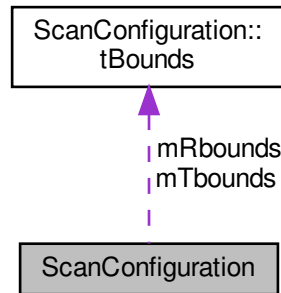
- `include/BayesianClustering/Rolproxy.hpp`
- `src/BayesianClustering/Rolproxy.cpp`

4.14 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](#) (const std::string &aCfgFile)
Constructor which parses the parameters when passed in as commandline arguments.
- [ScanConfiguration](#) (const std::size_t &aSigmaCount, const double &aSigmaMin, const double &aSigmaMax, const std::function< double(const double &) > &aInterpolator, const std::size_t &aRbins, const double &aMinScanR, const double &aMaxScanR, const std::size_t &aTbins, const double &aMinScanT, const double &aMaxScanT, const double &aPB, const double &aAlpha)
Constructor which take the parameters directly.
- [ScanConfiguration](#) (const [ScanConfiguration](#) &aOther)=delete
Deleted copy constructor.
- [ScanConfiguration](#) & operator= (const [ScanConfiguration](#) &aOther)=delete
Deleted assignment operator.
- [ScanConfiguration](#) ([ScanConfiguration](#) &&aOther)=default
Default move constructor.
- [ScanConfiguration](#) & operator= ([ScanConfiguration](#) &&aOther)=default
Default move-assignment constructor.
- [~ScanConfiguration](#) ()=default
Default destructor.
- 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.

4.14.1 Detailed Description

Class for storing the scan configuration parameters.

Definition at line 11 of file Configuration.hpp.

4.14.2 Constructor & Destructor Documentation

4.14.2.1 ScanConfiguration() [1/4]

```
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().

4.14.2.2 ScanConfiguration() [2/4]

```
ScanConfiguration::ScanConfiguration (
    const std::size_t & aSigmaCount,
    const double & aSigmaMin,
    const double & aSigmaMax,
    const std::function< double(const double &) > & aInterpolator,
    const std::size_t & aRbins,
    const double & aMinScanR,
    const double & aMaxScanR,
    const std::size_t & aTbins,
    const double & aMinScanT,
    const double & aMaxScanT,
    const double & aPB,
    const double & aAlpha )
```

Constructor which take the parameters directly.

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
<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
<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
<i>aPB</i>	The P_b parameter
<i>aAlpha</i>	The alpha parameter

Definition at line 55 of file Configuration.cpp.

References SetAlpha(), SetPb(), SetRBins(), SetSigmaParameters(), and SetTBins().

4.14.2.3 ScanConfiguration() [3/4]

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

Deleted copy constructor.

Parameters

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

4.14.2.4 ScanConfiguration() [4/4]

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

Default move constructor.

Parameters

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

4.14.3 Member Function Documentation

4.14.3.1 alpha()

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

Getter for the alpha parameter.

Returns

The alpha parameter

Definition at line 211 of file Configuration.hpp.

References mAlpha.

Referenced by Rolproxy::UpdateLogScore().

4.14.3.2 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 162 of file Configuration.cpp.

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

Referenced by `ScanConfiguration()`.

4.14.3.3 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 149 of file Configuration.hpp.

References `mLogProbabilitySigma`.

Referenced by `Cluster::UpdateLogScore()`.

4.14.3.4 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 178 of file Configuration.hpp.

References `mLogProbabilitySigma`.

4.14.3.5 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 217 of file Configuration.hpp.

References mLogAlpha.

Referenced by Rolproxy::UpdateLogScore().

4.14.3.6 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 223 of file Configuration.hpp.

References mLogGammaAlpha.

Referenced by Rolproxy::UpdateLogScore().

4.14.3.7 logPb()

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

Logarithm of the P_b parameter.

Returns

Logarithm of the P_b parameter

Definition at line 198 of file Configuration.hpp.

References mLogPb.

Referenced by Rolproxy::UpdateLogScore().

4.14.3.8 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 204 of file Configuration.hpp.

References mLogPbDagger.

Referenced by Rolproxy::UpdateLogScore().

4.14.3.9 operator=() [1/2]

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

Deleted assignment operator.

Returns

Reference to this, for chaining calls

Parameters

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

4.14.3.10 operator=() [2/2]

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

Default move-assignment constructor.

Returns

Reference to this, for chaining calls

Parameters

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

4.14.3.11 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 143 of file Configuration.hpp.

References mProbabilitySigma.

4.14.3.12 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 171 of file Configuration.hpp.

References mProbabilitySigma.

4.14.3.13 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 185 of file Configuration.hpp.

References mRbounds.

Referenced by Rol::ScanRT().

4.14.3.14 SetAlpha()

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

Setter for the alpha parameter.

Parameters

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

Definition at line 124 of file Configuration.cpp.

References `mAlpha`, `mLogAlpha`, and `mLogGammaAlpha`.

Referenced by `FromVector()`, and `ScanConfiguration()`.

4.14.3.15 SetPb()

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

Setter for the P_b parameter.

Parameters

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

Definition at line 117 of file Configuration.cpp.

References `mLogPb`, and `mLogPbDagger`.

Referenced by `FromVector()`, and `ScanConfiguration()`.

4.14.3.16 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 97 of file Configuration.cpp.

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

Referenced by FromVector(), and ScanConfiguration().

4.14.3.17 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 69 of file Configuration.cpp.

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

Referenced by FromVector(), and ScanConfiguration().

4.14.3.18 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 107 of file Configuration.cpp.

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

Referenced by FromVector(), and ScanConfiguration().

4.14.3.19 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 131 of file Configuration.hpp.

References mSigmabins.

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

4.14.3.20 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 157 of file Configuration.hpp.

References mSigmabins.

4.14.3.21 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 137 of file Configuration.hpp.

References mSigmabins2.

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

4.14.3.22 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 164 of file Configuration.hpp.

References mSigmabins2.

4.14.3.23 sigmacount()

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

Getter for the sigma count.

Returns

The sigma count

Definition at line 117 of file Configuration.hpp.

References mSigmacount.

Referenced by Rolproxy::ValidateLogScore().

4.14.3.24 sigmaspacing()

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

Getter for the sigma spacing.

Returns

The sigma spacing

Definition at line 124 of file Configuration.hpp.

References mSigmaspacing.

4.14.3.25 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 191 of file Configuration.hpp.

References mTbounds.

Referenced by Rol::ScanRT().

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

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

4.15 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.
- bool [operator==](#) (const [ScanEntry](#) &aOther)
Equality operator required by boost python.

Public Attributes

- double `r`
The R parameter
- double `t`
The T parameter.
- double `score`
The score.

4.15.1 Detailed Description

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

Definition at line 16 of file API.hpp.

4.15.2 Member Function Documentation

4.15.2.1 `operator<()`

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

Comparison operator for sorting.

Returns

Whether we are smaller than the other

Parameters

<i>aOther</i>	Another <code>ScanEntry</code> to compare against
---------------	---

Definition at line 24 of file API.hpp.

References `r`, and `t`.

4.15.2.2 `operator==()`

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

Equality operator required by boost python.

Returns

Whether we are equal to the other

Parameters

<i>aOther</i>	Another ScanEntry to compare against
---------------	--

Definition at line 33 of file API.hpp.

References [r](#), [score](#), and [t](#).

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

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

4.16 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.

4.16.1 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](#)

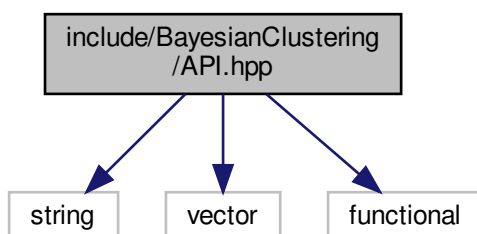
Chapter 5

File Documentation

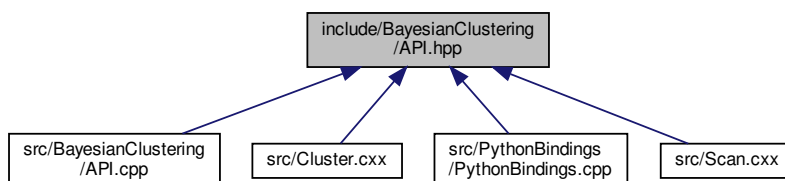
5.1 include/BayesianClustering/API.hpp File Reference

```
#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.
- struct [ClusterWrapper](#)
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_FullCallback](#) (const std::string &InFile, const double &aR, const double &aT, const std::function< void([RoiProxy](#) &) > &aCallback)
Automatically extract [Roi](#), clusterize and apply a full call-back.
- void [AutoRoi_Cluster_SimpleCallback](#) (const std::string &InFile, const double &aR, const double &aT, const std::function< void(const std::vector< [ClusterWrapper](#) > &) > &aCallback)
Automatically extract [Roi](#), clusterize and apply a full 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_FullCallback](#) (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 full call-back.
- void [ManualRoi_Cluster_SimpleCallback](#) (const std::string &InFile, const [ManualRoi](#) &aManualRoi, const double &aR, const double &aT, const std::function< void(const std::vector< [ClusterWrapper](#) > &) > &aCallback)
Manually extract [Roi](#), clusterize and apply a full call-back.

5.1.1 Function Documentation

5.1.1.1 AutoRoi_Cluster_FullCallback()

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

Automatically extract [RoI](#), clusterize and apply a full 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 104 of file API.cpp.

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

Referenced by `AutoRoi_Cluster_SimpleCallback()`.

5.1.1.2 AutoRoi_Cluster_SimpleCallback()

```
void AutoRoi_Cluster_SimpleCallback (
    const std::string & aInFile,
    const double & aR,
    const double & aT,
    const std::function< void(const std::vector< ClusterWrapper > &) > & aCallback )
```

Automatically extract [Rol](#), clusterize and apply a full 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 110 of file API.cpp.

References `_FullClusterToSimpleCluster_()`, and `AutoRoi_Cluster_FullCallback()`.

5.1.1.3 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 [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 84 of file API.cpp.

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

5.1.1.4 `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 90 of file API.cpp.

References `_FullScanToSimpleScan_()`, and `LocalizationFile::ExtractRols()`.

Referenced by `AutoRoi_Scan_ToJson()`.

5.1.1.5 `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 96 of file API.cpp.

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

Referenced by `BOOST_PYTHON_MODULE()`.

5.1.1.6 ManualRoi_Cluster_FullCallback()

```
void ManualRoi_Cluster_FullCallback (
    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 full call-back.

Parameters

<i>aInFile</i>	The name of the localization file
<i>aManualRoi</i>	The manually-specified Roi 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 138 of file API.cpp.

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

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

5.1.1.7 ManualRoi_Cluster_SimpleCallback()

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

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

Parameters

<i>aInFile</i>	The name of the localization file
<i>aManualRoi</i>	The manually-specified Roi 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 144 of file API.cpp.

References [_FullClusterToSimpleCluster_\(\)](#), and [ManualRoi_Cluster_FullCallback\(\)](#).

5.1.1.8 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 Roi window
<i>aScanConfig</i>	The configuration for the scan
<i>aCallback</i>	The full callback to be applied

Definition at line 118 of file API.cpp.

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

5.1.1.9 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 Roi window
<i>aScanConfig</i>	The configuration for the scan
<i>aCallback</i>	The simple callback to be applied

Definition at line 124 of file API.cpp.

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

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

5.1.1.10 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 Roi window
<i>aScanConfig</i>	The configuration for the scan
<i>aOutFile</i>	The name of the output JSON file

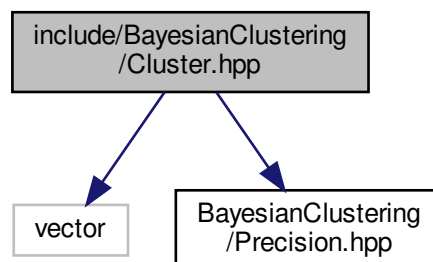
Definition at line 130 of file API.cpp.

References [_ScanCallback_Json\(\)](#), and [ManualRoi_Scan_SimpleCallback\(\)](#).

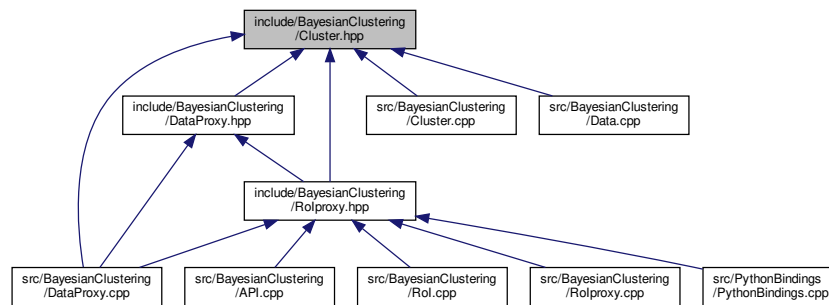
Referenced by [BOOST_PYTHON_MODULE\(\)](#).

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:



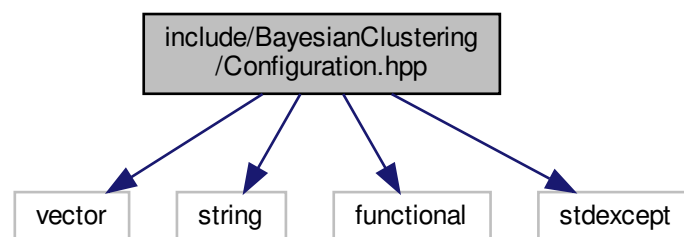
Classes

- class [Cluster](#)
A class representing a cluster.
- struct [Cluster::Parameter](#)
A struct representing the cluster parameters.

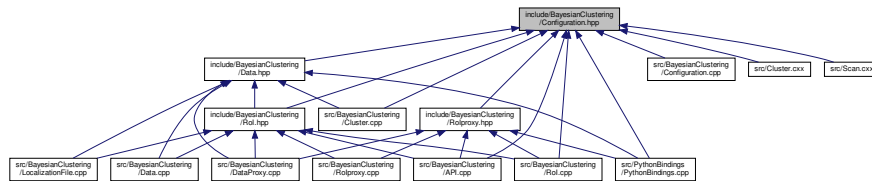
5.3 include/BayesianClustering/Configuration.hpp File Reference

```
#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:



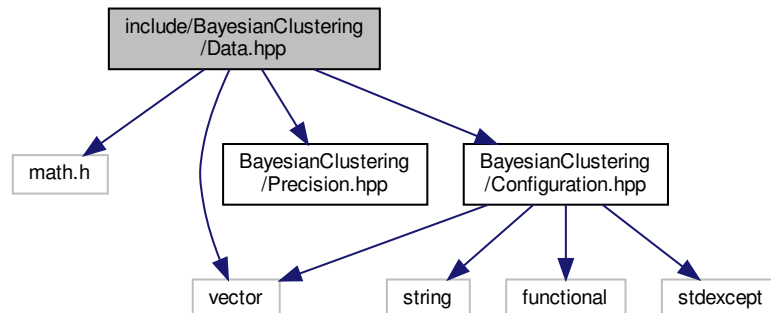
Classes

- 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.

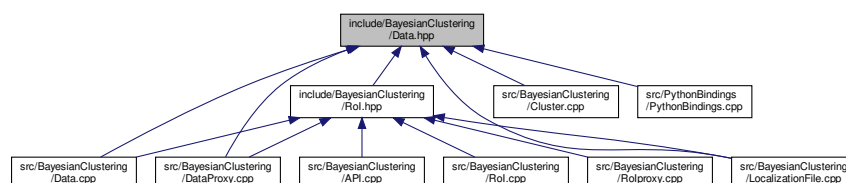
5.4 include/BayesianClustering/Data.hpp File Reference

```
#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:



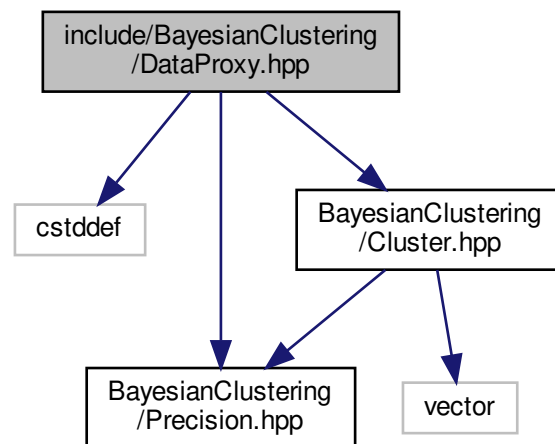
Classes

- class [Data](#)

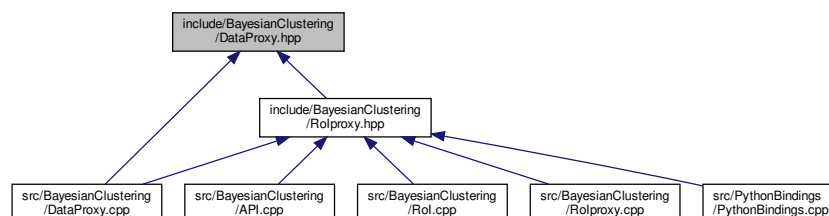
A class to store the raw data-points.

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

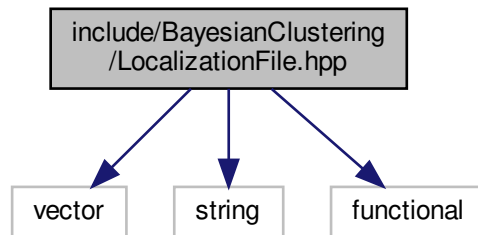
- class [DataProxy](#)

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

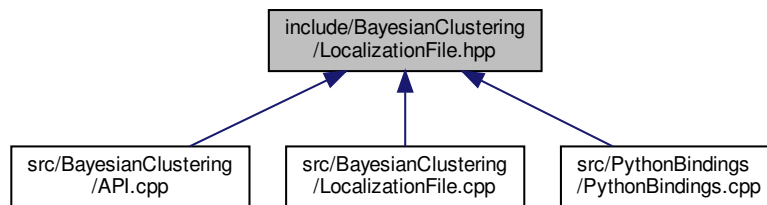
5.6 include/BayesianClustering/LocalizationFile.hpp File Reference

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

Include dependency graph for LocalizationFile.hpp:



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

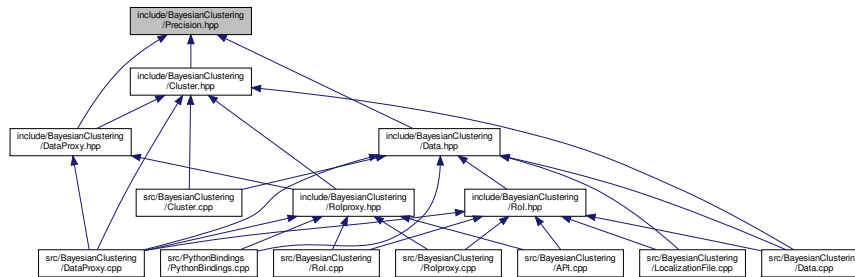


Classes

- 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:



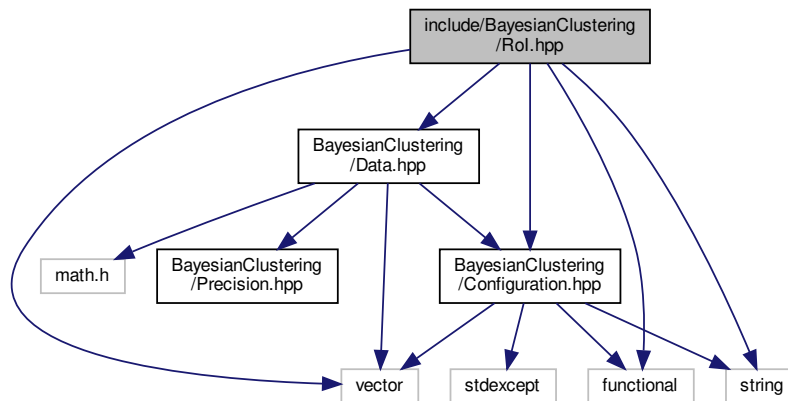
5.8 include/BayesianClustering/Rol.hpp File Reference

```

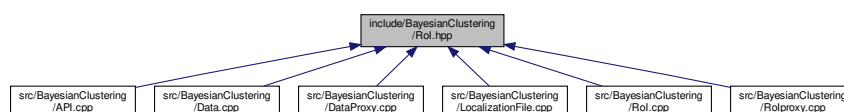
#include <vector>
#include <functional>
#include <string>
#include "BayesianClustering/Data.hpp"
#include "BayesianClustering/Configuration.hpp"

```

Include dependency graph for Rol.hpp:



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



Classes

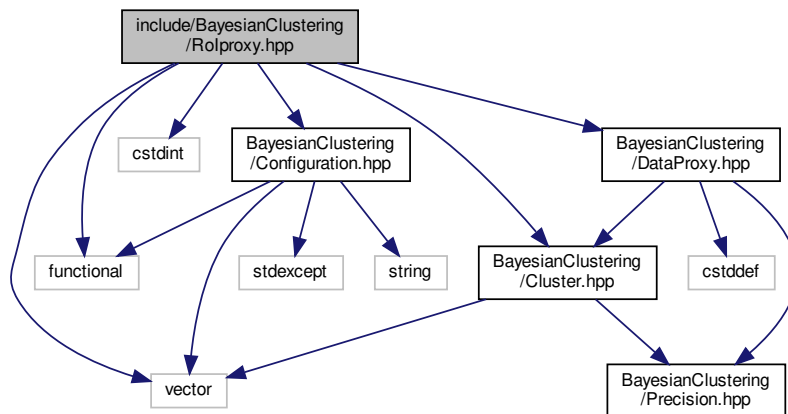
- class [Rol](#)

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

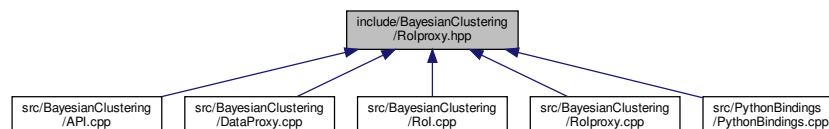
5.9 include/BayesianClustering/Rolproxy.hpp File Reference

```
#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

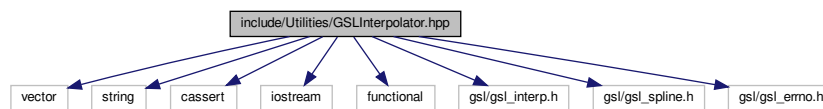
- class [Rolproxy](#)

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

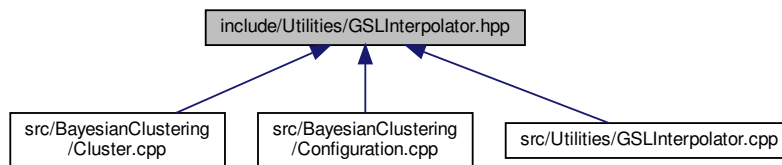
5.10 include/Utilities/GSLInterpolator.hpp File Reference

```
#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](#)

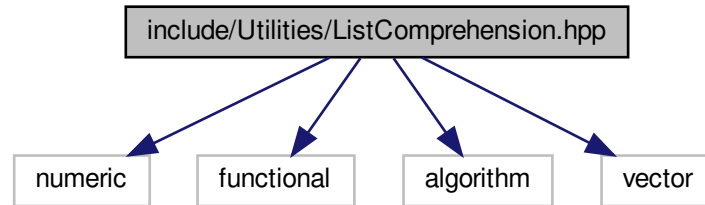
A utility wrapper around the GSL interpolator to give it a clean C++ interface.

5.11 include/Utilities/ListComprehension.hpp File Reference

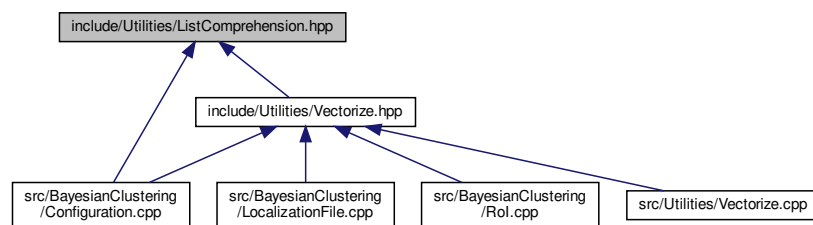
```
#include <numeric>
#include <functional>
#include <algorithm>
```

```
#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<t↵
Container>::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

N	The number of elements
-----	------------------------

Returns

A vector of ints

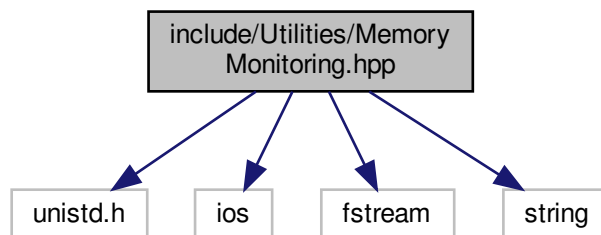
Definition at line 70 of file ListComprehension.hpp.

Referenced by LocalizationFile::LocalizationFile(), Rol::Preprocess(), Rol::ScanRT(), and ScanConfiguration::SetSigmaParameters().

5.12 include/Utilities/MemoryMonitoring.hpp File Reference

```
#include <unistd.h>
#include <ios>
#include <fstream>
#include <string>
```

Include dependency graph for MemoryMonitoring.hpp:



Functions

- void `mem_usage` (double &vm_usage, double &resident_set)
Utility to get Virtual Memory and Resident Set usage.

5.12.1 Function Documentation

5.12.1.1 mem_usage()

```
void mem_usage (
    double & vm_usage,
    double & resident_set )
```

Utility to get Virtual Memory and Resident Set usage.

Parameters

<i>vm_usage</i>	Return the Virtual Memory usage
<i>resident_set</i>	Return the Resident Set usage

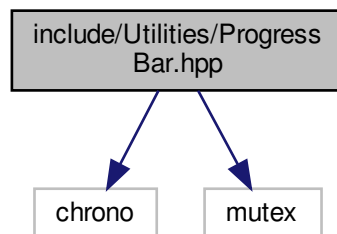
Definition at line 12 of file MemoryMonitoring.hpp.

5.13 include/Utilities/ProgressBar.hpp File Reference

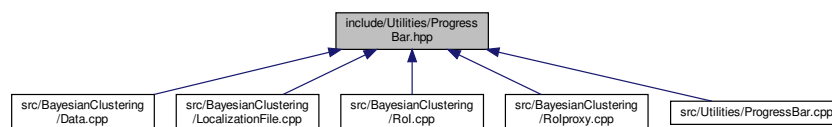
```
#include <chrono>
```

```
#include <mutex>
```

Include dependency graph for ProgressBar.hpp:



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



Classes

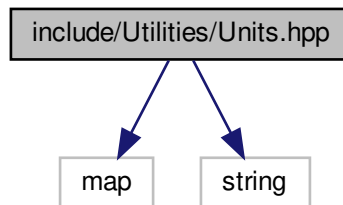
- class [ProgressBar](#)
A utility progress-bar.
- struct [ProgressTimer](#)
A utility code timer.

5.14 include/Utilities/Units.hpp File Reference

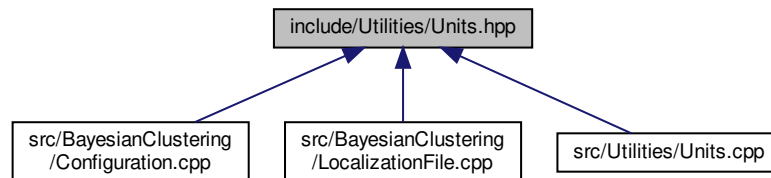
```
#include <map>
```

```
#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.

5.14.1 Function Documentation

5.14.1.1 `operator""_micrometer()` [1/2]

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

User-defined literals fot micrometer quantities.

Parameters

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

Returns

The literal value

Definition at line 39 of file Units.hpp.

References micrometer.

5.14.1.2 `operator""_micrometer()` [2/2]

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

User-defined literals fot micrometer quantities.

Parameters

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

Returns

The literal value

Definition at line 47 of file Units.hpp.

References micrometer.

5.14.1.3 operator""_nanometer() [1/2]

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

User-defined literals fot nanometer quantities.

Parameters

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

Returns

The literal value

Definition at line 23 of file Units.hpp.

References nanometer.

5.14.1.4 operator""_nanometer() [2/2]

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

User-defined literals fot nanometer quantities.

Parameters

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

Returns

The literal value

Definition at line 31 of file Units.hpp.

References nanometer.

5.14.1.5 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

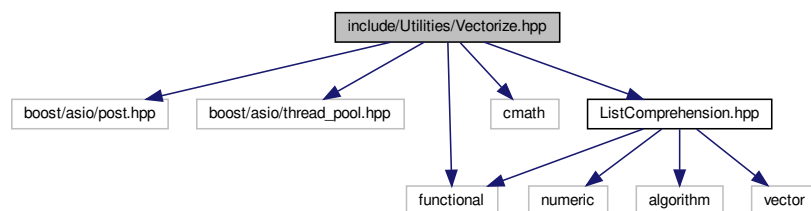
Definition at line 12 of file Units.cpp.

References UnitMap.

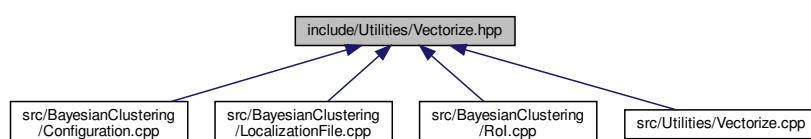
Referenced by ScanConfiguration::FromVector(), and AuxConfiguration::FromVector().

5.15 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.

5.15.1 Function Documentation

5.15.1.1 [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](#).

5.15.1.2 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.

5.15.2 Variable Documentation

5.15.2.1 Nthreads

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

Utility variable for the concurrency.

Utility variable for the concurrency.

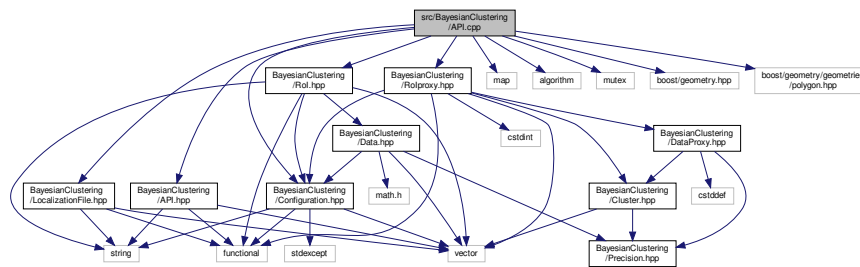
Definition at line 8 of file Vectorize.cpp.

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

5.16 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 <map>
#include <algorithm>
#include <mutex>
#include <boost/geometry.hpp>
#include <boost/geometry/geometries/polygon.hpp>
```

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 [_FullScanToSimpleScan_](#) (RoI &aRoI, const [ScanConfiguration](#) &aScanConfig, const std::function< void(const std::vector< [ScanEntry](#) > &) > &aCallback)
A callback to neatly package the scan results for easy consumption.
- void [_FullClusterToSimpleCluster_](#) (RoIproxy &aRoIproxy, const std::function< void(const std::vector< [ClusterWrapper](#) > &) > &aCallback)
A callback to neatly package the scan results for easy consumption.
- void [AutoRoI_Scan_FullCallback](#) (const std::string &alnFile, 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 &alnFile, 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 &alnFile, const [ScanConfiguration](#) &aScanConfig, const std::function< void(const std::string &aOutFile) > &aCallback)
Automatically extract [RoI](#), run scan and dump to JSON file.
- void [AutoRoI_Cluster_FullCallback](#) (const std::string &alnFile, const double &aR, const double &aT, const std::function< void([RoIproxy](#) &) > &aCallback)
Automatically extract [RoI](#), clusterize and apply a full call-back.
- void [AutoRoI_Cluster_SimpleCallback](#) (const std::string &alnFile, const double &aR, const double &aT, const std::function< void(const std::vector< [ClusterWrapper](#) > &) > &aCallback)
Automatically extract [RoI](#), clusterize and apply a full call-back.
- void [ManualRoI_Scan_FullCallback](#) (const std::string &alnFile, 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 &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.

- 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.

- void [ManualRoi_Cluster_FullCallback](#) (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 full call-back.

- void [ManualRoi_Cluster_SimpleCallback](#) (const std::string &aInFile, const [ManualRoi](#) &aManualRoi, const double &aR, const double &aT, const std::function< void(const std::vector< [ClusterWrapper](#) > &) > &aCallback)

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

5.16.1 Function Documentation

5.16.1.1 [_FullClusterToSimpleCluster_\(\)](#)

```
void _FullClusterToSimpleCluster_ (
    RoIproxy & aRoIproxy,
    const std::function< void(const std::vector< ClusterWrapper > &) > & aCallback )
```

A callback to neatly package the scan results for easy consumption.

Parameters

<i>aRoIproxy</i>	The region-proxy containing the clusters
<i>aCallback</i>	The simple callback to be applied

Definition at line 54 of file API.cpp.

References [RoIproxy::mData](#).

Referenced by [AutoRoi_Cluster_SimpleCallback\(\)](#), and [ManualRoi_Cluster_SimpleCallback\(\)](#).

5.16.1.2 [_FullScanToSimpleScan_\(\)](#)

```
void _FullScanToSimpleScan_ (
    RoI & aRoI,
    const ScanConfiguration & aScanConfig,
    const std::function< void(const std::vector< ScanEntry > &) > & aCallback )
```

A callback to neatly package the scan results for easy consumption.

Parameters

<i>aRoi</i>	The region of interest
<i>aScanConfig</i>	The configuration for the scan
<i>aCallback</i>	The simple callback to be applied

Definition at line 41 of file API.cpp.

References `RoiProxy::mLogP`, and `Roi::ScanRT()`.

Referenced by `AutoRoi_Scan_SimpleCallback()`, and `ManualRoi_Scan_SimpleCallback()`.

5.16.1.3 _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 23 of file API.cpp.

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

5.16.1.4 AutoRoi_Cluster_FullCallback()

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

Automatically extract `Roi`, clusterize and apply a full 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 104 of file API.cpp.

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

Referenced by AutoRoi_Cluster_SimpleCallback().

5.16.1.5 AutoRoi_Cluster_SimpleCallback()

```
void AutoRoi_Cluster_SimpleCallback (
    const std::string & aInFile,
    const double & aR,
    const double & aT,
    const std::function< void(const std::vector< ClusterWrapper > &) > & aCallback )
```

Automatically extract Rol, clusterize and apply a full 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 110 of file API.cpp.

References _FullClusterToSimpleCluster_(), and AutoRoi_Cluster_FullCallback().

5.16.1.6 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 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 84 of file API.cpp.

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

5.16.1.7 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 [Roi](#), 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 90 of file API.cpp.

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

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

5.16.1.8 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 96 of file API.cpp.

References [_ScanCallback_Json_\(\)](#), and [AutoRoi_Scan_SimpleCallback\(\)](#).

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

5.16.1.9 ManualRoi_Cluster_FullCallback()

```
void ManualRoi_Cluster_FullCallback (
    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 full call-back.

Parameters

<i>aInFile</i>	The name of the localization file
<i>aManualRoI</i>	The manually-specified RoI 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 138 of file API.cpp.

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

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

5.16.1.10 ManualRoi_Cluster_SimpleCallback()

```

void ManualRoi_Cluster_SimpleCallback (
    const std::string & aInFile,
    const ManualRoI & aManualRoI,
    const double & aR,
    const double & aT,
    const std::function< void(const std::vector< ClusterWrapper > &) > & aCallback )

```

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

Parameters

<i>aInFile</i>	The name of the localization file
<i>aManualRoI</i>	The manually-specified RoI 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 144 of file API.cpp.

References [_FullClusterToSimpleCluster_\(\)](#), and [ManualRoi_Cluster_FullCallback\(\)](#).

5.16.1.11 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 RoI window
<i>aScanConfig</i>	The configuration for the scan
<i>aCallback</i>	The full callback to be applied

Definition at line 118 of file API.cpp.

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

5.16.1.12 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 RoI window
<i>aScanConfig</i>	The configuration for the scan
<i>aCallback</i>	The simple callback to be applied

Definition at line 124 of file API.cpp.

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

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

5.16.1.13 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 Roi window
<i>aScanConfig</i>	The configuration for the scan
<i>aOutFile</i>	The name of the output JSON file

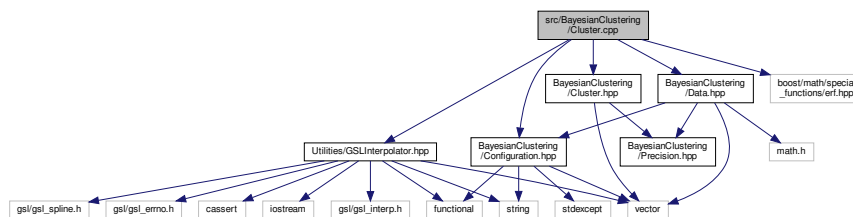
Definition at line 130 of file API.cpp.

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

Referenced by `BOOST_PYTHON_MODULE()`.

5.17 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.

5.17.1 Function Documentation

5.17.1.1 `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

x	The position to evaluate the normal_cdf at
σ	The standard-deviation of the Gaussian
$x0$	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().

5.18 src/BayesianClustering/Configuration.cpp File Reference

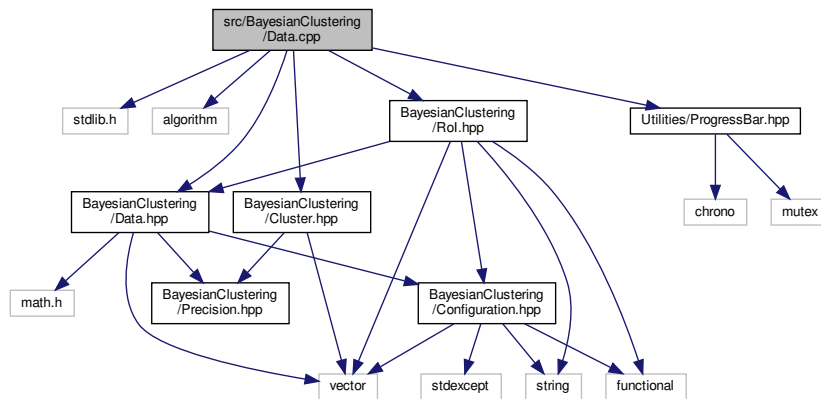
```
#include "Utilities/GSLInterpolator.hpp"
#include "Utilities/ListComprehension.hpp"
#include "Utilities/Vectorize.hpp"
#include "Utilities/Units.hpp"
#include "BayesianClustering/Configuration.hpp"
#include <iostream>
#include <fstream>
#include <streambuf>
#include <boost/math/special_functions/gamma.hpp>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



5.19 src/BayesianClustering/Data.cpp File Reference

```
#include <stdlib.h>
#include <algorithm>
#include "BayesianClustering/Data.hpp"
#include "BayesianClustering/Cluster.hpp"
#include "BayesianClustering/RoI.hpp"
```

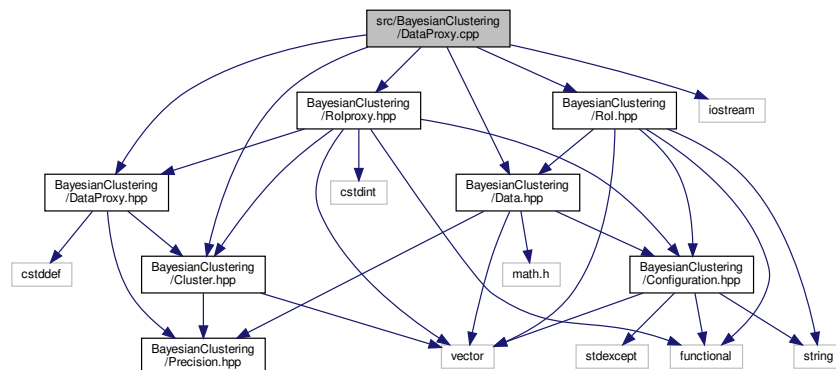
```
#include "Utilities/ProgressBar.hpp"
Include dependency graph for Data.cpp:
```



5.20 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:



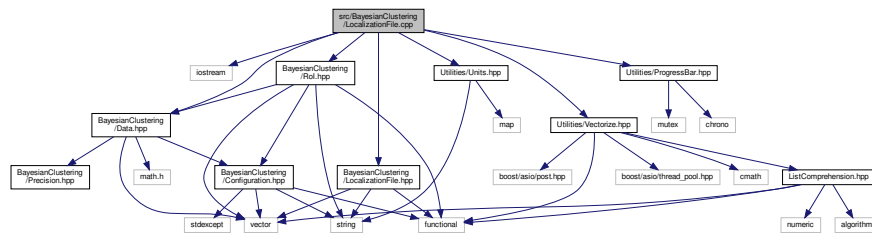
Macros

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

5.21 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.

5.21.1 Function Documentation

5.21.1.1 __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 23 of file LocalizationFile.cpp.

References nanometer.

Referenced by LocalizationFile::LocalizationFile().

5.21.1.2 __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

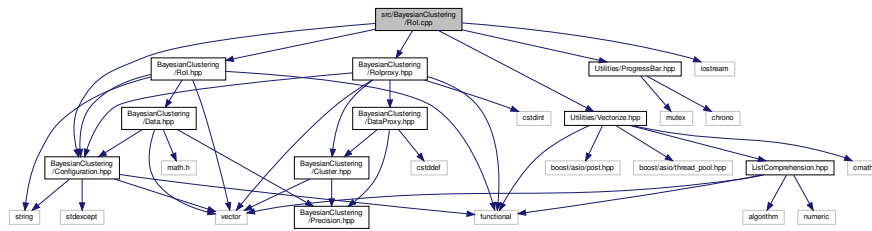
Definition at line 109 of file LocalizationFile.cpp.

Referenced by LocalizationFile::ExtractRols().

5.22 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:



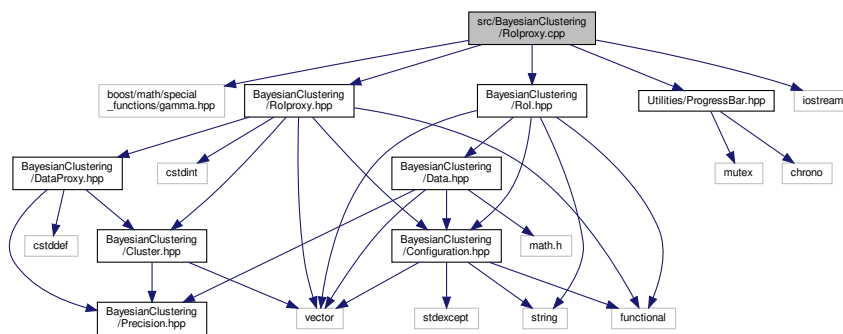
5.23 src/BayesianClustering/Rolproxy.cpp File Reference

```

#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:



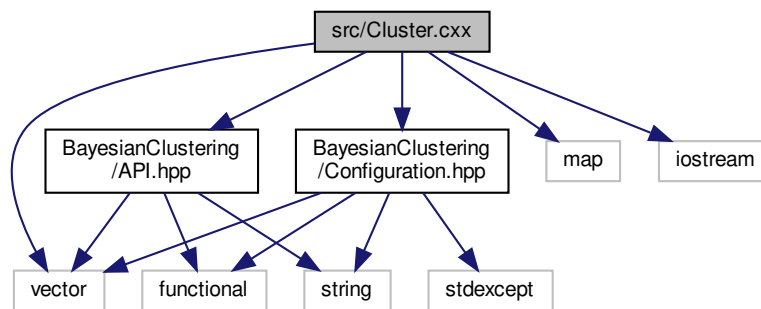
5.24 src/Cluster.cxx File Reference

```

#include "BayesianClustering/API.hpp"
#include "BayesianClustering/Configuration.hpp"
#include <map>
#include <vector>
#include <iostream>

```

Include dependency graph for Cluster.cxx:



Functions

- void [ReportClusters](#) (const std::vector< [ClusterWrapper](#) > &aClusters)
Callback to report clusters.
- int [main](#) (int argc, char **argv)
The main function.

5.24.1 Function Documentation

5.24.1.1 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 28 of file Cluster.cxx.

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

5.24.1.2 ReportClusters()

```
void ReportClusters (
    const std::vector< ClusterWrapper > & aClusters )
```

Callback to report clusters.

Parameters

<code>aClusters</code>	A vector of clusters
------------------------	----------------------

Definition at line 14 of file Cluster.cxx.

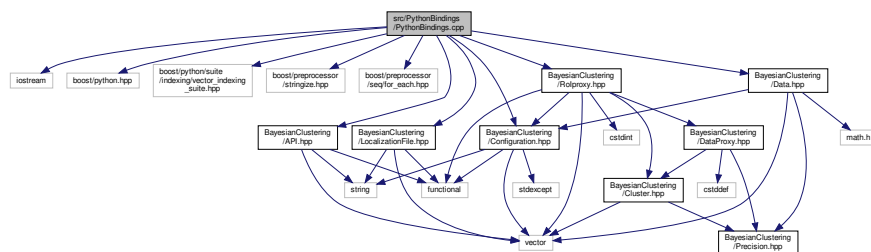
Referenced by main().

5.25 src/PythonBindings/PythonBindings.cpp File Reference

Self-contained sourcefile for producing python-bindings.

```
#include <iostream>
#include <boost/python.hpp>
#include <boost/python/suite/indexing/vector_indexing_suite.hpp>
#include <boost/preprocessor/stringize.hpp>
#include <boost/preprocessor/seq/for_each.hpp>
#include "BayesianClustering/API.hpp"
#include "BayesianClustering/LocalizationFile.hpp"
#include "BayesianClustering/Configuration.hpp"
#include "BayesianClustering/RoIproxy.hpp"
#include "BayesianClustering/Data.hpp"
#include "BayesianClustering/Precision.hpp"
```

Include dependency graph for PythonBindings.cpp:



Macros

- `#define ADAPT_CALLBACK_CONSTRUCTOR(SIGNATURE) template<> template<> SIGNATURE< boost::python::object& >(boost::python::object& aCallback) : function([&](auto&&... aArgs){ aCallback(aArgs...); }) {}`

Helper macro to define a constructor for a `std::function` of a given signature from a `boost::python` callback object. Using explicit specialization of the class to the specified type and explicit specialization of the constructor to take a `boost::python::object` (hence two `template<>`'s at the start), create a generic variadic lambda that captures the `boost::python` callback object and pass the lambda to a deferred constructor.

- `#define FN(X, ...) def(#X , &X , args(__VA_ARGS__))`

Helper Macro to simplify defining functions.

- #define [ADAPTED_FN](#)(X, ...) def(#X , +Adapted::X , args(__VA_ARGS__))

Helper Macro to simplify defining functions with python callbacks.

- #define [STRUCT_ARG](#)(r, CLASS, ARG) .def_readwrite(BOOST_PP_STRINGIZE(ARG) , &CLASS::ARG)

Helper Macro to deal with the boilerplate when dealing with structs.

- #define [EXPOSE_STRUCT](#)(CLASS, ARGS) class_< CLASS >(#CLASS) BOOST_PP_SEQ_FOR_EACH([STRUCT_ARG](#) , CLASS , ARGS)

Helper Macro to deal with the boilerplate when dealing with structs.

- #define [EXPOSE_VECTOR](#)(CLASS) class_< std::vector< CLASS> >("Vector##CLASS").def(vector_↵ indexing_suite< std::vector< CLASS > >());

Helper Macro to deal with the boilerplate when dealing with vectors of objects.

Functions

- [ADAPT_CALLBACK_CONSTRUCTOR](#) (std::function< void([Rolproxy](#) &, const double &, const double &) >)

Define a std::function constructor for full scan callback.

- [ADAPT_CALLBACK_CONSTRUCTOR](#) (std::function< void(const std::vector< [ScanEntry](#) > &) >)

Define a std::function constructor for simple scan callback.

- [ADAPT_CALLBACK_CONSTRUCTOR](#) (std::function< void([Rolproxy](#) &) >)

Define a std::function constructor for full clusterizer callback.

- [ADAPT_CALLBACK_CONSTRUCTOR](#) (std::function< void(const std::vector< [ClusterWrapper](#) > &) >)

Define a std::function constructor for simple clusterizer callback.

- [BOOST_PYTHON_MODULE](#) (BayesianClustering)

Boost Python Wrapper providing bindings for our C++ functions.

Variables

- auto [Adapted::AutoRoi_Scan_SimpleCallback](#) = [](const std::string& aInFile , const [ScanConfiguration](#)& aScanConfig , const boost::python::object& aCallback){ ::AutoRoi_Scan_SimpleCallback(aInFile , aScan↵ Config , aCallback); }

Lambda to automatically extract [Rol](#), run scan and apply a simple python callback.

- auto [Adapted::AutoRoi_Cluster_SimpleCallback](#) = [](const std::string& aInFile , const double& aR , const double& aT , const boost::python::object& aCallback){ ::AutoRoi_Cluster_SimpleCallback(aInFile , aR , aT , aCallback); }

Lambda to automatically extract [Rol](#), clusterize and apply a simple python callback.

- auto [Adapted::ManualRoi_Scan_SimpleCallback](#) = [](const std::string& aInFile , const [ManualRoi](#)& a↵ ManualRoi , const [ScanConfiguration](#)& aScanConfig , const boost::python::object& aCallback){ ::Manual↵ Roi_Scan_SimpleCallback(aInFile , aManualRoi , aScanConfig , aCallback); }

Lambda to manually extract [Rol](#), run scan and apply a simple python callback.

- auto [Adapted::ManualRoi_Cluster_SimpleCallback](#) = [](const std::string& aInFile , const [ManualRoi](#)& a↵ ManualRoi , const double& aR , const double& aT , const boost::python::object& aCallback){ ::ManualRoi↵ _Cluster_SimpleCallback(aInFile , aManualRoi , aR , aT , aCallback); }

Lambda to manually extract [Rol](#), clusterize and apply a simple python callback.

5.25.1 Detailed Description

Self-contained sourcefile for producing python-bindings.

5.25.2 Macro Definition Documentation

5.25.2.1 ADAPT_CALLBACK_CONSTRUCTOR

```
#define ADAPT_CALLBACK_CONSTRUCTOR(  
    SIGNATURE ) template<> template<> SIGNATURE::function< boost::python::object&  
>( boost::python::object& aCallback ) : function( [&]( auto&&... aArgs ){ aCallback( a←  
Args... ); } ) {}
```

Helper macro to define a constructor for a `std::function` of a given signature from a `boost::python` callback object. Using explicit specialization of the class to the specified type and explicit specialization of the constructor to take a `boost::python::object` (hence two `template<>`'s at the start), create a generic variadic lambda that captures the `boost::python` callback object and pass the lambda to a deferred constructor.

Gnarly!

Parameters

<i>SIGNATURE</i>	The signature of the <code>std::function</code> we are creating a constructor for
------------------	---

Definition at line 38 of file `PythonBindings.cpp`.

5.25.2.2 ADAPTED_FN

```
#define ADAPTED_FN(  
    X,  
    ... ) def( #X , +Adapted::X , args( __VA_ARGS__ ) )
```

Helper Macro to simplify defining functions with `python` callbacks.

Parameters

<i>X</i>	The function being defined
...	List of strings giving argument names

Definition at line 73 of file `PythonBindings.cpp`.

5.25.2.3 EXPOSE_STRUCT

```
#define EXPOSE_STRUCT(  
    CLASS,  
    ARGS ) class_< CLASS >( #CLASS ) BOOST_PP_SEQ_FOR_EACH( STRUCT_ARG , CLASS ,  
ARGS )
```

Helper Macro to deal with the boilerplate when dealing with structs.

Parameters

<i>CLASS</i>	The Class name
<i>ARGS</i>	One of the arguments

Definition at line 84 of file PythonBindings.cpp.

5.25.2.4 EXPOSE_VECTOR

```
#define EXPOSE_VECTOR(  
    CLASS ) class_< std::vector< CLASS> >( "Vector##CLASS" ).def( vector_indexing↵  
_suite< std::vector< CLASS > >() );
```

Helper Macro to deal with the boilerplate when dealing with vectors of objects.

Parameters

<i>CLASS</i>	The Class name
--------------	----------------

Definition at line 88 of file PythonBindings.cpp.

5.25.2.5 FN

```
#define FN(  
    X,  
    ... ) def( #X , &X , args( __VA_ARGS__ ) )
```

Helper Macro to simplify defining functions.

Parameters

<i>X</i>	The function being defined
<i>...</i>	List of strings giving argument names

Definition at line 68 of file PythonBindings.cpp.

5.25.2.6 STRUCT_ARG

```
#define STRUCT_ARG(  
    r,  
    CLASS,  
    ARG ) .def_readwrite( BOOST_PP_STRINGIZE( ARG ) , &CLASS::ARG )
```

Helper Macro to deal with the boilerplate when dealing with structs.

Parameters

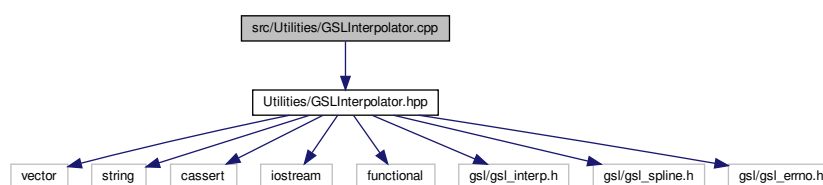
<i>r</i>	BOOST PP internal
<i>CLASS</i>	The Class name
<i>ARG</i>	One of the arguments

Definition at line 79 of file PythonBindings.cpp.

5.26 src/Utilities/GSLInterpolator.cpp File Reference

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

Include dependency graph for GSLInterpolator.cpp:



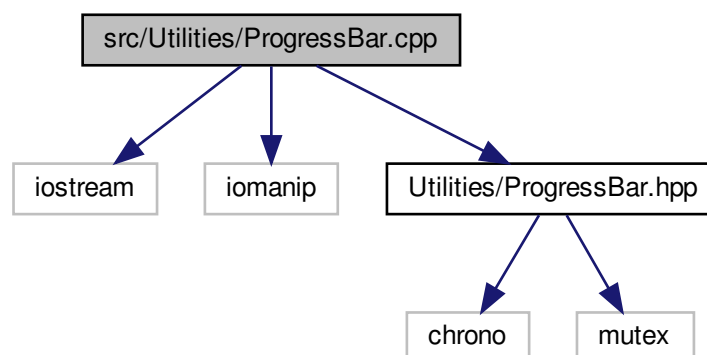
5.27 src/Utilities/ProgressBar.cpp File Reference

```
#include <iostream>
```

```
#include <iomanip>
```

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

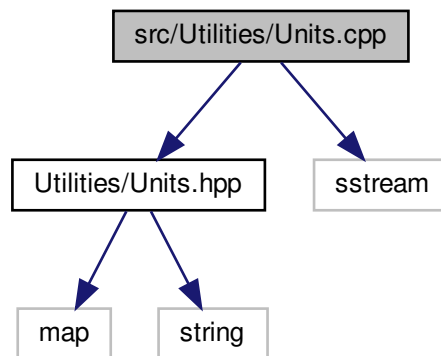
Include dependency graph for ProgressBar.cpp:



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.

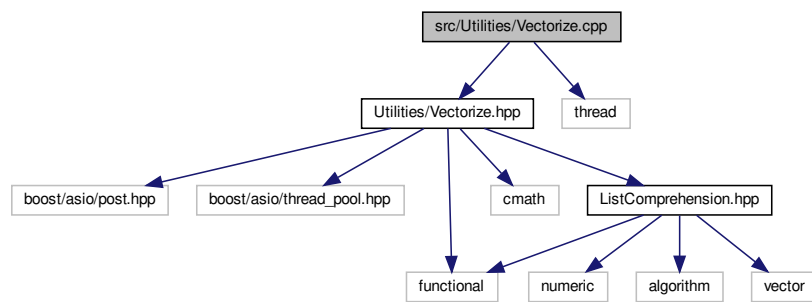
Referenced by ScanConfiguration::FromVector(), and AuxConfiguration::FromVector().

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

- [_FullClusterToSimpleCluster_](#)
 - [API.cpp, 102](#)
 - [_FullScanToSimpleScan_](#)
 - [API.cpp, 102](#)
 - [_ScanCallback_Json_](#)
 - [API.cpp, 103](#)
 - [__LoadCSV__](#)
 - [LocalizationFile.cpp, 111](#)
 - [__RecursiveSearch__](#)
 - [LocalizationFile.cpp, 112](#)
- [ADAPT_CALLBACK_CONSTRUCTOR](#)
 - [PythonBindings.cpp, 117](#)
- [ADAPTED_FN](#)
 - [PythonBindings.cpp, 117](#)
- [alpha](#)
 - [ScanConfiguration, 62](#)
- [alt_log_score](#)
 - [Cluster::Parameter, 40](#)
- [API.cpp](#)
 - [_FullClusterToSimpleCluster_, 102](#)
 - [_FullScanToSimpleScan_, 102](#)
 - [_ScanCallback_Json_, 103](#)
 - [AutoRoi_Cluster_FullCallback, 103](#)
 - [AutoRoi_Cluster_SimpleCallback, 104](#)
 - [AutoRoi_Scan_FullCallback, 104](#)
 - [AutoRoi_Scan_SimpleCallback, 104](#)
 - [AutoRoi_Scan_ToJson, 105](#)
 - [ManualRoi_Cluster_FullCallback, 105](#)
 - [ManualRoi_Cluster_SimpleCallback, 106](#)
 - [ManualRoi_Scan_FullCallback, 106](#)
 - [ManualRoi_Scan_SimpleCallback, 107](#)
 - [ManualRoi_Scan_ToJson, 107](#)
- [API.hpp](#)
 - [AutoRoi_Cluster_FullCallback, 76](#)
 - [AutoRoi_Cluster_SimpleCallback, 78](#)
 - [AutoRoi_Scan_FullCallback, 78](#)
 - [AutoRoi_Scan_SimpleCallback, 79](#)
 - [AutoRoi_Scan_ToJson, 79](#)
 - [ManualRoi_Cluster_FullCallback, 79](#)
 - [ManualRoi_Cluster_SimpleCallback, 80](#)
 - [ManualRoi_Scan_FullCallback, 80](#)
 - [ManualRoi_Scan_SimpleCallback, 81](#)
 - [ManualRoi_Scan_ToJson, 81](#)
- [AutoRoi_Cluster_FullCallback](#)
 - [API.cpp, 103](#)
 - [API.hpp, 76](#)
- [AutoRoi_Cluster_SimpleCallback](#)
 - [API.cpp, 104](#)
 - [API.hpp, 78](#)
- [AutoRoi_Scan_FullCallback](#)
 - [API.cpp, 104](#)
 - [API.hpp, 78](#)
- [AutoRoi_Scan_SimpleCallback](#)
 - [API.cpp, 104](#)
 - [API.hpp, 79](#)
- [AutoRoi_Scan_ToJson](#)
 - [API.cpp, 105](#)
 - [API.hpp, 79](#)
- [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, 21](#)
- [CheckClusterization](#)
 - [Rolproxy, 54](#)
- [Cluster, 13](#)
 - [Cluster, 14, 15](#)
 - [GetParent, 15](#)
 - [operator+=", 16](#)
 - [operator=, 16](#)
 - [UpdateLogScore, 17](#)
- [Cluster.cpp](#)
 - [normal_cdf, 108](#)
- [Cluster.cxx](#)
 - [main, 114](#)
 - [ReportClusters, 114](#)
- [Cluster::Parameter, 40](#)
 - [alt_log_score, 40](#)
 - [log_score, 41](#)
 - [operator+=", 41](#)
- [Clusterize](#)
 - [DataProxy, 27, 28](#)
 - [Rol, 47](#)
 - [Rolproxy, 54](#)
- [ClusterR](#)
 - [AuxConfiguration, 9](#)
- [ClusterT](#)
 - [AuxConfiguration, 9](#)

- ClusterWrapper, 17
 - operator==, 18
- configFile
 - AuxConfiguration, 9
- Data, 19
 - CalculateLocalizationScore, 21
 - Data, 20, 21
 - dPhi, 22
 - dR, 22
 - dR2, 22
 - operator<, 23
 - operator=, 23, 24
 - Preprocess, 24
 - PreprocessLocalizationScores, 25
- DataProxy, 25
 - Clusterize, 27, 28
 - DataProxy, 26, 27
 - GetCluster, 28
 - operator=, 28, 29
- Deriv
 - GSLInterpolator, 32
- Deriv2
 - GSLInterpolator, 32
- dPhi
 - Data, 22
- dR
 - Data, 22
- dR2
 - Data, 22
- Eval
 - GSLInterpolator, 33
- Evaluate
 - GSLInterpolator, 33
- EXPOSE_STRUCT
 - PythonBindings.cpp, 117
- EXPOSE_VECTOR
 - PythonBindings.cpp, 118
- ExtractRols
 - LocalizationFile, 37, 38
- FN
 - PythonBindings.cpp, 118
- FromVector
 - AuxConfiguration, 10
 - ScanConfiguration, 62
- getArea
 - RoI, 47
- getCentreX
 - RoI, 48
- getCentreY
 - RoI, 48
- GetCluster
 - DataProxy, 28
- GetData
 - RoIproxy, 55
- GetParent
 - Cluster, 15
- getWidthX
 - RoI, 48
- getWidthY
 - RoI, 49
- GSLInterpolator, 29
 - Deriv, 32
 - Deriv2, 32
 - Eval, 33
 - Evaluate, 33
 - GSLInterpolator, 30, 31
 - Integ, 33
 - operator=, 34
 - SetData, 35
- include/BayesianClustering/API.hpp, 75
- include/BayesianClustering/Cluster.hpp, 82
- include/BayesianClustering/Configuration.hpp, 83
- include/BayesianClustering/Data.hpp, 84
- include/BayesianClustering/DataProxy.hpp, 85
- include/BayesianClustering/LocalizationFile.hpp, 86
- include/BayesianClustering/Precision.hpp, 87
- include/BayesianClustering/RoI.hpp, 87
- include/BayesianClustering/RoIproxy.hpp, 88
- include/Utilities/GSLInterpolator.hpp, 89
- include/Utilities/ListComprehension.hpp, 89
- include/Utilities/MemoryMonitoring.hpp, 93
- include/Utilities/ProgressBar.hpp, 94
- include/Utilities/Units.hpp, 95
- include/Utilities/Vectorize.hpp, 98
- inputFile
 - AuxConfiguration, 10
- Integ
 - GSLInterpolator, 33
- ListComprehension.hpp
 - operator | , 90–92
 - range, 92
- LocalizationFile, 36
 - ExtractRols, 37, 38
 - LocalizationFile, 36, 37
 - operator=, 38
- LocalizationFile.cpp
 - __LoadCSV__, 111
 - __RecursiveSearch__, 112
- log_probability_sigma
 - ScanConfiguration, 63
- log_score
 - Cluster::Parameter, 41
- logAlpha
 - ScanConfiguration, 63
- logGammaAlpha
 - ScanConfiguration, 64
- logPb
 - ScanConfiguration, 64
- logPbDagger
 - ScanConfiguration, 64
- main

- Cluster.cxx, 114
- ManualRoi, 39
- ManualRoi_Cluster_FullCallback
 - API.cpp, 105
 - API.hpp, 79
- ManualRoi_Cluster_SimpleCallback
 - API.cpp, 106
 - API.hpp, 80
- ManualRoi_Scan_FullCallback
 - API.cpp, 106
 - API.hpp, 80
- ManualRoi_Scan_SimpleCallback
 - API.cpp, 107
 - API.hpp, 81
- ManualRoi_Scan_ToJson
 - API.cpp, 107
 - API.hpp, 81
- mem_usage
 - MemoryMonitoring.hpp, 93
- MemoryMonitoring.hpp
 - mem_usage, 93
- normal_cdf
 - Cluster.cpp, 108
- Nthreads
 - Vectorize.cpp, 121
 - Vectorize.hpp, 100
- operator<
 - Data, 23
 - ScanEntry, 72
- operator++
 - ProgressBar, 43
- operator+=
 - Cluster, 16
 - Cluster::Parameter, 41
- operator=
 - Cluster, 16
 - Data, 23, 24
 - DataProxy, 28, 29
 - GSLInterpolator, 34
 - LocalizationFile, 38
 - Roi, 49
 - RoiProxy, 55, 56
 - ScanConfiguration, 65
- operator==
 - ClusterWrapper, 18
 - ScanEntry, 72
- operator&&
 - Vectorize.hpp, 99
- operator""_micrometer
 - Units.hpp, 96
- operator""_nanometer
 - Units.hpp, 97
- operator |
 - ListComprehension.hpp, 90–92
- operator | |
 - Vectorize.hpp, 99
- outputFile
 - AuxConfiguration, 10
- Preprocess
 - Data, 24
 - Roi, 50
- PreprocessLocalizationScores
 - Data, 25
- probability_sigma
 - ScanConfiguration, 66
- ProgressBar, 42
 - operator++, 43
 - ProgressBar, 43
- ProgressTimer, 44
 - ProgressTimer, 44
- PythonBindings.cpp
 - ADAPT_CALLBACK_CONSTRUCTOR, 117
 - ADAPTED_FN, 117
 - EXPOSE_STRUCT, 117
 - EXPOSE_VECTOR, 118
 - FN, 118
 - STRUCT_ARG, 118
- range
 - ListComprehension.hpp, 92
- Rbounds
 - ScanConfiguration, 66
- ReportClusters
 - Cluster.cxx, 114
- Roi, 45
 - Clusterize, 47
 - getArea, 47
 - getCentreX, 48
 - getCentreY, 48
 - getWidthX, 48
 - getWidthY, 49
 - operator=, 49
 - Preprocess, 50
 - Roi, 46, 47
 - ScanRT, 50
 - SetCentre, 51
 - SetWidth, 51
- RoiProxy, 52
 - CheckClusterization, 54
 - Clusterize, 54
 - GetData, 55
 - operator=, 55, 56
 - RoiProxy, 53, 54
 - ScanRT, 56
 - UpdateLogScore, 56
 - ValidateLogScore, 57
- ScanConfiguration, 57
 - alpha, 62
 - FromVector, 62
 - log_probability_sigma, 63
 - logAlpha, 63
 - logGammaAlpha, 64
 - logPb, 64
 - logPbDagger, 64

- operator=, 65
- probability_sigma, 66
- Rbounds, 66
- ScanConfiguration, 60–62
- SetAlpha, 66
- SetPb, 67
- SetRBins, 67
- SetSigmaParameters, 68
- SetTBins, 68
- sigmabins, 69
- sigmabins2, 69, 70
- sigmacount, 70
- sigmaspacing, 70
- Tbounds, 71
- ScanConfiguration::tBounds, 73
- ScanEntry, 71
 - operator<, 72
 - operator==, 72
- ScanRT
 - Rol, 50
 - Rolproxy, 56
- SetAlpha
 - ScanConfiguration, 66
- SetCentre
 - Rol, 51
- SetConfigFile
 - AuxConfiguration, 11
- SetData
 - GSLInterpolator, 35
- SetInputFile
 - AuxConfiguration, 11
- SetOutputFile
 - AuxConfiguration, 12
- SetPb
 - ScanConfiguration, 67
- SetRBins
 - ScanConfiguration, 67
- SetSigmaParameters
 - ScanConfiguration, 68
- SetTBins
 - ScanConfiguration, 68
- SetValidate
 - AuxConfiguration, 12
- SetWidth
 - Rol, 51
- sigmabins
 - ScanConfiguration, 69
- sigmabins2
 - ScanConfiguration, 69, 70
- sigmacount
 - ScanConfiguration, 70
- sigmaspacing
 - ScanConfiguration, 70
- src/BayesianClustering/API.cpp, 101
- src/BayesianClustering/Cluster.cpp, 108
- src/BayesianClustering/Configuration.cpp, 109
- src/BayesianClustering/Data.cpp, 109
- src/BayesianClustering/DataProxy.cpp, 110
- src/BayesianClustering/LocalizationFile.cpp, 111
- src/BayesianClustering/Rol.cpp, 112
- src/BayesianClustering/Rolproxy.cpp, 113
- src/Cluster.cxx, 113
- src/PythonBindings/PythonBindings.cpp, 115
- src/Utilities/GSLInterpolator.cpp, 119
- src/Utilities/ProgressBar.cpp, 119
- src/Utilities/Units.cpp, 120
- src/Utilities/Vectorize.cpp, 121
- StrToDist
 - Units.cpp, 120
 - Units.hpp, 97
- STRUCT_ARG
 - PythonBindings.cpp, 118
- Tbounds
 - ScanConfiguration, 71
- Units.cpp
 - StrToDist, 120
- Units.hpp
 - operator""_micrometer, 96
 - operator""_nanometer, 97
 - StrToDist, 97
- UpdateLogScore
 - Cluster, 17
 - Rolproxy, 56
- validate
 - AuxConfiguration, 12
- ValidateLogScore
 - Rolproxy, 57
- Vectorize.cpp
 - Nthreads, 121
- Vectorize.hpp
 - Nthreads, 100
 - operator&&, 99
 - operator | |, 99