Bayesian Cluster Tool

Generated by Doxygen 1.9.1

Thu Jul 13 2023 18:15:44

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 &a↩ ScanConfig, 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:

Chapter 3

File Index

3.1 File List

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

Chapter 4

Class Documentation

4.1 AutoRoI Struct Reference

A struct for storing the parameters for automatically extracting the RoIs.

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

4.1.1 Detailed Description

A struct for storing the parameters for automatically extracting the RoIs.

Definition at line 30 of file LocalizationFile.hpp.

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

- include/BayesianClustering/LocalizationFile.hpp

4.2 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.

4.2.1 Detailed Description

Class for storing the auxilliary configuration parameters.

Definition at line 284 of file Configuration.hpp.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 AuxConfiguration() [1/2]

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

Default constructor.

Constructor which parses the parameters when passed in as commandline arguments

Parameters

argc	The number of commandline arguments
argv	The commandline arguments

Definition at line 231 of file Configuration.cpp.

References FromVector().

4.2.2.2 AuxConfiguration() [2/2]

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

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

Parameters

aArgs	The commandline arguments

Definition at line 240 of file Configuration.cpp.

References FromVector().

4.2.3 Member Function Documentation

4.2.3.1 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 344 of file Configuration.hpp.

References mClusterR.

Referenced by main().

4.2.3.2 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 351 of file Configuration.hpp.

References mClusterT.

Referenced by main().

4.2.3.3 configFile()

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

Getter for the config file.

Returns

The name of the config file

Definition at line 337 of file Configuration.hpp.

References mConfigFile.

Referenced by main().

4.2.3.4 FromVector()

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

Parse the parameters when passed in as commandline arguments.

Parameters

aArgs	The commandline arguments

Definition at line 279 of file Configuration.cpp.

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

Referenced by AuxConfiguration().

4.2.3.5 inputFile()

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

Getter for the input file.

Returns

The name of the input RoI file

Definition at line 323 of file Configuration.hpp.

References mInputFile.

Referenced by main().

4.2.3.6 outputFile()

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

Getter for the output file.

Returns

The name of the output file

Definition at line 330 of file Configuration.hpp.

References mOutputFile.

Referenced by main().

4.2.3.7 SetConfigFile()

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

Setter for the config file.

Parameters

aFileName	The name of the file

Definition at line 267 of file Configuration.cpp.

References mConfigFile.

Referenced by FromVector().

4.2.3.8 SetInputFile()

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

Setter for the input file.

Parameters

aFileName	The name of the file

Definition at line 254 of file Configuration.cpp.

References mInputFile.

Referenced by FromVector().

4.2.3.9 SetOutputFile()

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

Setter for the output file.

Parameters

aFileName	The name of the file

Definition at line 260 of file Configuration.cpp.

References mOutputFile.

Referenced by FromVector().

4.2.3.10 SetValidate()

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

Set whether to validate clusterization.

Parameters

aValidate	Whether to validate clusterization

Definition at line 248 of file Configuration.cpp.

References mValidate.

Referenced by FromVector().

4.2.3.11 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 316 of file Configuration.hpp.

References mValidate.

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

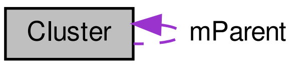
- include/BayesianClustering/Configuration.hpp
- src/BayesianClustering/Configuration.cpp

4.3 Cluster Class Reference

A class representing a cluster.

`#include <Cluster.hpp>`

Collaboration diagram for Cluster:



Classes

- struct Parameter

  A struct representing the cluster parameters.

Public Member Functions

- Cluster (const std::size_t &aParamSize)

  Default constructor.
- Cluster (const Data &aData, const std::vector< double > &aSigmabins2)

  Construct a cluster from a single data-point.
- Cluster (const Cluster &aOther)=delete

  Deleted copy constructor.
- Cluster & operator= (const Cluster &aOther)=delete

  Deleted assignment operator.
- Cluster (Cluster &&aOther)=default

  Default move constructor.
- Cluster & operator= (Cluster &&aOther)=default

  Default move-assignment constructor.
- ∼Cluster ()

  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.

4.3.1 Detailed Description

A class representing a cluster.

Definition at line 16 of file Cluster.hpp.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 Cluster() [1/4]

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

Default constructor.

Parameters

aParamSize	The number of sigma-bins

Definition at line 93 of file Cluster.cpp.

4.3.2.2 Cluster() [2/4]

```
Cluster::Cluster (
            const Data & aData,
            const std::vector< double > & aSigmabins2 )
```

Construct a cluster from a single data-point.

Parameters

aData	A data-point with which to initialize the cluster
aSigmabins2	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.

4.3.2.3 Cluster() [3/4]

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

Deleted copy constructor.

Parameters

aOther	Anonymous argument

4.3.2.4 Cluster() [4/4]

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

Default move constructor.

Parameters

aOther	Anonymous argument

4.3.3 Member Function Documentation

4.3.3.1 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().

4.3.3.2 operator+=()

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

Add another cluster to this one.

Parameters

aOther	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.

4.3.3.3 operator=() [1/2]

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

Default move-assignment constructor.

Returns

Reference to this, for chaining calls

Parameters

aOther	Anonymous argument

4.3.3.4 operator=() [2/2]

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

Deleted assignment operator.

Returns

Reference to this, for chaining calls

Parameters

aOther	Anonymous argument

4.3.3.5 UpdateLogScore()

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

Update log-probability after a scan.

Parameters

aScanConfig	The configuration parameters for the scan

Definition at line 124 of file Cluster.cpp.

References ScanConfiguration::log_probability_sigma(), mClusterScore, mClusterSize, mLastClusterSize, m←
Params, and ScanConfiguration::sigmabins().

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

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

4.4 ClusterWrapper Struct Reference

A struct for storing extracted parameters from a cluster.

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

Public Member Functions

- bool operator< (const ClusterWrapper &aOther)

  Comparison operator for sorting.
- 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.

4.4.1 Detailed Description

A struct for storing extracted parameters from a cluster.

Definition at line 47 of file API.hpp.

4.4.2 Member Function Documentation

4.4.2.1 operator<()

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

Comparison operator for sorting.

Returns

Whether we are smaller than the other

Parameters

aOther	Another ClusterWrapper to compare against

Definition at line 57 of file API.hpp.

References area, localizations, and perimeter.

4.4.2.2 operator==()

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

Equality operator required by boost python.

Returns

Whether we are equal to the other

Parameters

aOther	Another ClusterWrapper to compare against

Definition at line 67 of file API.hpp.

References centroid_x, and centroid_y.

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

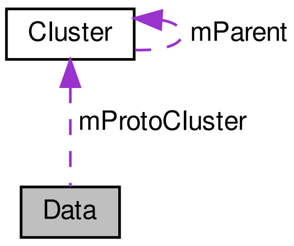
- include/BayesianClustering/API.hpp

4.5 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 locaalization scores, one per R-bin.
- std::vector< std::pair< PRECISION, std::size_t > > mNeighbours

  The list of neighbours as a pair of squared-distance and index into the list of points.
- Cluster ∗ mProtoCluster

  A cluster containing only this data-point.

4.5.1 Detailed Description

A class to store the raw data-points.

Definition at line 17 of file Data.hpp.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 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.

4.5.2.2 Data() [2/3]

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

Deleted copy constructor.

Parameters

aOther	Anonymous argument

4.5.2.3 Data() [3/3]

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

Default move constructor.

Parameters

aOther	Anonymous argument

4.5.3 Member Function Documentation

4.5.3.1 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

aData	?
R	?
aArea	The area of the window for normalizing the log score

Returns

The localization score

Definition at line 106 of file Data.cpp.

References mNeighbours.

4.5.3.2 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

aOther	A data-point to compare against

Definition at line 71 of file Data.hpp.

References phi.

4.5.3.3 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

aOther	A data-point to compare against

Definition at line 63 of file Data.hpp.

References dR2().

4.5.3.4 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

aOther	A data-point to compare against

Definition at line 54 of file Data.hpp.

References x, and y.

Referenced by dR().

4.5.3.5 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

aOther	A data-point to compare against

Definition at line 46 of file Data.hpp.

References r.

4.5.3.6 operator=() [1/2]

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

Deleted assignment operator.

Returns

Reference to this, for chaining calls

Parameters

aOther	Anonymous argument

4.5.3.7 operator=() [2/2]

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

Default move-assignment constructor.

Returns

Reference to this, for chaining calls

Parameters

aOther	Anonymous argument

4.5.3.8 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

aData	The collection of data-points
aIndex	The index of the current data-point
aMax2R	Twice the maximum radius out to which we will cluster
aMax2R2	Square of twice the maximum radius out to which we will cluster
aSigmabins2	The sigma-bins for initializing clusters
aProgressBar	The progress bar to update

Definition at line 28 of file Data.cpp.

4.5.3.9 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

aData	?
aScanConfig	The configuration parameters for the scan
aArea	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:

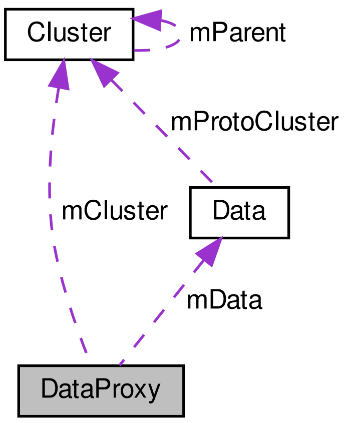
- include/BayesianClustering/Data.hpp
- src/BayesianClustering/Data.cpp

4.6 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, RoIproxy &aRoI)

  Entry point clusterization function - a new cluster will be created.
- void Clusterize (const PRECISION &a2R2, RoIproxy &aRoI, 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.

4.6.1 Detailed Description

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

Definition at line 18 of file DataProxy.hpp.

4.6.2 Constructor & Destructor Documentation

4.6.2.1 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.

4.6.2.2 DataProxy() [2/3]

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

Deleted copy constructor.

Parameters

aOther	Anonymous argument

4.6.2.3 DataProxy() [3/3]

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

Default move constructor.

Parameters

aOther	Anonymous argument

4.6.3 Member Function Documentation

4.6.3.1 Clusterize() [1/2]

```
void DataProxy::Clusterize (
             const PRECISION & a2R2,
             RoIproxy & aRoI )
```

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

Parameters

a2R2	The clusterization radius
aRoI	The RoI-proxy in which we are running

Definition at line 23 of file DataProxy.cpp.

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

Referenced by Clusterize().

4.6.3.2 Clusterize() [2/2]

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

Recursive clusterization function.

Parameters

a2R2	The clusterization radius
aRoI	The RoI-proxy in which we are running
aCluster	The cluster we are building
d	The recursion depth

Definition at line 34 of file DataProxy.cpp.

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

4.6.3.3 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().

4.6.3.4 operator=() [1/2]

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

Deleted assignment operator.

Returns

Reference to this, for chaining calls

Parameters

aOther	Anonymous argument

4.6.3.5 operator=() `[2/2]`

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

Default move-assignment constructor.

Returns

Reference to this, for chaining calls

Parameters

aOther	Anonymous argument

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

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

4.7 GSLInterpolator Class Reference

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

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

Public Member Functions

- GSLInterpolator (const gsl_interp_type ∗type, const unsigned int &ndata)

  Empty splice constructor.
- GSLInterpolator (const gsl_interp_type ∗type, const std::vector< double > &x, const std::vector< double > &y)

  Initialised splice constructor.
- GSLInterpolator (const gsl_interp_type ∗type, const std::map< double, double > &data)

  Initialised splice constructor.
- virtual ∼GSLInterpolator ()

  Destructor.
- GSLInterpolator (const GSLInterpolator &aOther)=delete

  Deleted copy constructor.
- GSLInterpolator & operator= (const GSLInterpolator &aOther)=delete

Deleted assignment operator.

- GSLInterpolator (GSLInterpolator &&aOther)=default

  Default move constructor.
- GSLInterpolator & operator= (GSLInterpolator &&aOther)=default

  Default move-assignment constructor.
- bool SetData (const std::vector< double > &x, const std::vector< double > &y)

  Set the spline data points.
- bool SetData (const unsigned int &ndata, const double ∗x, const double ∗y)

  Set the spline data points.
- double Evaluate (const std::function< int(double &) > &aFunction, const std::string &aName)

  Utility function that runs the GSL function that has been wrapped in a lambda below.
- double Eval (const double &x)

  Evaluate the spline at the given x.
- double Deriv (const double &x)

  The first derivative of the spline at the given x.
- double Deriv2 (const double &x)

  The second derivative of the spline at the given x.
- double Integ (const double &a, const double &b)

  The integral over the spline between two bounds.

Private Attributes

- unsigned int nErrors

  An error counter to suppress excess messages.
- gsl_interp_accel ∗ fAccel

  Underlying GSL machinery.
- gsl_spline ∗ fSpline

  Underlying GSL machinery for the spline itself.
- const gsl_interp_type ∗ fInterpType

  Underlying GSL machinery for the interpolation type.

4.7.1 Detailed Description

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

Definition at line 20 of file GSLInterpolator.hpp.

4.7.2 Constructor & Destructor Documentation

4.7.2.1 GSLInterpolator() [1/5]

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

Empty splice constructor.

Parameters

type	The spline type
ndata	The number of points that will be added to the spline

Definition at line 9 of file GSLInterpolator.cpp.

References fInterpType, and fSpline.

4.7.2.2 GSLInterpolator() `[2/5]`

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

Initialised splice constructor.

Parameters

type	The spline type
x	The points on the x-axis
y	The points on the y-axis

Definition at line 19 of file GSLInterpolator.cpp.

References fInterpType, fSpline, and SetData().

4.7.2.3 GSLInterpolator() `[3/5]`

```
GSLInterpolator::GSLInterpolator (
             const gsl_interp_type * type,
             const std::map< double, double > & data )
```

Initialised splice constructor.

Parameters

type	The spline type
data	Data points along the spline

Definition at line 32 of file GSLInterpolator.cpp.

References fInterpType, fSpline, and SetData().

4.7.2.4 GSLInterpolator() [4/5]

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

Deleted copy constructor.

Parameters

aOther	Anonymous argument

4.7.2.5 GSLInterpolator() [5/5]

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

Default move constructor.

Parameters

aOther	Anonymous argument

4.7.3 Member Function Documentation

4.7.3.1 Deriv()

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

The first derivative of the spline at the given x.

Parameters

x	The x-coordinate at which to evaluate the derivative

Returns

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

Definition at line 110 of file GSLInterpolator.hpp.

References Evaluate(), fAccel, and fSpline.

4.7.3.2 Deriv2()

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

The second derivative of the spline at the given x.

Parameters

x	The x-coordinate at which to evaluate the derivative

Returns

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

Definition at line 120 of file GSLInterpolator.hpp.

References Evaluate(), fAccel, and fSpline.

4.7.3.3 Eval()

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

Evaluate the spline at the given x.

Parameters

x	The x-coordinate at which to evaluate the spline

Returns

The value of the spline at the given x-coordinate

Definition at line 100 of file GSLInterpolator.hpp.

References Evaluate(), fAccel, and fSpline.

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

4.7.3.4 Evaluate()

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

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

Parameters

aFunction	A lambda that will be evaluated
aName	The operation name for the debugging messages

Returns

The interpolated value

Definition at line 81 of file GSLInterpolator.hpp.

References fAccel, and nErrors.

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

4.7.3.5 Integ()

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

The integral over the spline between two bounds.

Parameters

a	The lower bound of the integral
b	The upper bound of the integral

Returns

The integral over the spline between a and b

Definition at line 131 of file GSLInterpolator.hpp.

References Evaluate(), fAccel, and fSpline.

4.7.3.6 operator=() [1/2]

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

Deleted assignment operator.

Returns

Reference to this, for chaining calls

Parameters

aOther	Anonymous argument

4.7.3.7 operator=() [2/2]

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

Default move-assignment constructor.

Returns

Reference to this, for chaining calls

Parameters

aOther	Anonymous argument

4.7.3.8 SetData() [1/2]

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

Set the spline data points.

Parameters

x	The x-coordinates of the datapoints
y	The y-coordinates of the datapoints

Returns

success or fail

Definition at line 64 of file GSLInterpolator.hpp.

Referenced by GSLInterpolator().

4.7.3.9 SetData() [2/2]

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

Set the spline data points.

Parameters

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

Returns

success or fail

Definition at line 59 of file GSLInterpolator.cpp.

References fAccel, fInterpType, fSpline, and nErrors.

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

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

4.8 ImageJRoI Struct Reference

A struct for storing the parameters of an ImageJ RoI file.

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

Public Member Functions

- ImageJRoI (const std::string &aFilename, const double &aScale)

  Constructor

Public Attributes

- std::string filename

  The ImageJ zipped RoI file.
- double scale

  The pixel scale.

4.8.1 Detailed Description

A struct for storing the parameters of an ImageJ RoI file.

Definition at line 36 of file LocalizationFile.hpp.

4.8.2 Constructor & Destructor Documentation

4.8.2.1 ImageJRoI()

```
ImageJRoI::ImageJRoI (
             const std::string & aFilename,
             const double & aScale )  [inline]
```

Constructor

Parameters

aFilename	The ImageJ zipped RoI file
aScale	The pixel scale

Definition at line 38 of file LocalizationFile.hpp.

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

- include/BayesianClustering/LocalizationFile.hpp

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

  Manually extract an RoI.
- void ExtractRoIs (const AutoRoI &aRoI, const std::function< void(RoI &) > &aCallback) const

  Automatically extract the RoIs.
- void ExtractRoIs (const ImageJRoI &aRoI, const std::function< void(RoI &) > &aCallback) const

  Manually extract an RoI.
- const std::vector< Data > & data ()

  Accessor to the raw data.

Private Attributes

- std::string mFilename

  The localization file name.
- std::vector< Data > mData

  The localizations in the file.

4.9.1 Detailed Description

A class to store the raw data-points.

Definition at line 49 of file LocalizationFile.hpp.

4.9.2 Constructor & Destructor Documentation

4.9.2.1 LocalizationFile() [1/3]

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

Constructor.

Parameters

aFilename	The name of the localizations file

Definition at line 75 of file LocalizationFile.cpp.

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

4.9.2.2 LocalizationFile() [2/3]

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

Deleted copy constructor.

Parameters

aOther	Anonymous argument

4.9.2.3 LocalizationFile() [3/3]

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

Default move constructor.

Parameters

aOther	Anonymous argument

4.9.3 Member Function Documentation

4.9.3.1 data()

```
const std::vector< Data >& LocalizationFile::data ( )  [inline]
```

Accessor to the raw data.

Returns

Reference to the raw data

Definition at line 91 of file LocalizationFile.hpp.

References mData.

Referenced by CheckRoIs(), and GetLocalizations().

4.9.3.2 ExtractRoIs() [1/3]

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

Automatically extract the RoIs.

Parameters

aRoI	Parameters for automatically extracting RoIs
aCallback	A handler for each RoI found

Local record to store the size, the bounds and the datapoints

< The number of histogram cells in the RoI

< The mean X of the RoI

< The mean Y of the RoI

< The data points in the RoI

Definition at line 142 of file LocalizationFile.cpp.

References __RecursiveSearch__(), and mData.

4.9.3.3 ExtractRoIs() [2/3]

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

Manually extract an RoI.

Parameters

aRoI	Wrapper for an ImageJ RoI file
aCallback	A handler for each RoI found

Definition at line 321 of file LocalizationFile.cpp.

References ImageJRoI::filename, mData, OpenRoiZipfile(), and ImageJRoI::scale.

4.9.3.4 ExtractRoIs() [3/3]

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

Manually extract an RoI.

Parameters

aRoI	The manual RoI window
aCallback	A handler for each RoI found

Definition at line 107 of file LocalizationFile.cpp.

References ManualRoI::height, mData, ManualRoI::width, ManualRoI::x, and ManualRoI::y.

Referenced by CheckRoIs(), RunClustering(), and RunScan().

4.9.3.5 operator=() [1/2]

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

Deleted assignment operator.

Returns

Reference to this, for chaining calls

Parameters

aOther	Anonymous argument

4.9.3.6 operator=() [2/2]

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

Default move-assignment constructor.

Returns

Reference to this, for chaining calls

Parameters

aOther	Anonymous argument

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

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

4.10 ManualRoI Struct Reference

A struct for storing the parameters of a manual RoI.

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

Public Member Functions

- ManualRoI (const double &aX, const double &aY, const double &aWidth, const double &aHeight)

  Constructor.

Public Attributes

- double x

  The x-centre of the RoI.
- double y

  The y-centre of the RoI.
- double width

  The width of the RoI.
- double height

  The height of the RoI.

4.10.1 Detailed Description

A struct for storing the parameters of a manual RoI.

Definition at line 15 of file LocalizationFile.hpp.

4.10.2 Constructor & Destructor Documentation

4.10.2.1 ManualRoI()

```
ManualRoI::ManualRoI (
            const double & aX,
            const double & aY,
            const double & aWidth,
            const double & aHeight )  [inline]
```

Constructor.

Parameters

aX	The x-centre of the RoI
aY	The x-centre of the RoI
aWidth	The width of the RoI
aHeight	The height of the RoI

Definition at line 17 of file LocalizationFile.hpp.

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

- include/BayesianClustering/LocalizationFile.hpp

4.11 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.

4.11.1 Detailed Description

A struct representing the cluster parameters.

Definition at line 21 of file Cluster.hpp.

4.11.2 Member Function Documentation

4.11.2.1 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().

4.11.2.2 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().

4.11.2.3 operator+=()

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

Add another set of parameters to this set.

Parameters

aOther	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

4.12 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.

4.12.1 Detailed Description

A utility progress-bar.

Definition at line 8 of file ProgressBar.hpp.

4.12.2 Constructor & Destructor Documentation

4.12.2.1 ProgressBar()

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

Constructor.

Parameters

aLabel	A description of the task being timed
aMax	The number of calls equalling 100%

Definition at line 8 of file ProgressBar.cpp.

References mLabel, and print().

4.12.3 Member Function Documentation

4.12.3.1 operator++()

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

Prefix increment.

Parameters

aDummy	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

4.13 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.

4.13.1 Detailed Description

A utility code timer.

Definition at line 49 of file ProgressBar.hpp.

4.13.2 Constructor & Destructor Documentation

4.13.2.1 ProgressTimer()

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

Constructor.

Parameters

aLabel	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

4.14 RoI Class Reference

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

`#include <RoI.hpp>`

Public Member Functions

- RoI (const std::string &aId, std::vector< Data > &&aData, const double &aPhysicalCentreX, const double &aPhysicalCentreY, const double &aArea)

    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.
- const double & getCentreX () const

    Getter for the x-coordinate of the physical centre.
- const double & getCentreY () const

    Getter for the y-coordinate of the physical centre.
- const double & getArea () const

    Getter for the height of the ROI window.
- const std::vector< Data > & data () const

    Accessor to the raw data.
- const std::string & id () const

    Accessor to the RoI ID.

Private Attributes

- std::string mId

    The ID of the ROI.
- std::vector< Data > mData

    The collection of raw data points.
- 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 mArea

    The area of the window in physical units.

Friends

- class **RoIproxy**

4.14.1 Detailed Description

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

Definition at line 17 of file RoI.hpp.

4.14.2 Constructor & Destructor Documentation

4.14.2.1 RoI() [1/3]

```
RoI::RoI (
            const std::string & aId,
            std::vector< Data > && aData,
            const double & aPhysicalCentreX,
            const double & aPhysicalCentreY,
            const double & aArea )
```

Default Constructor.

Parameters

aId	The ID of the RoI
aData	The set of data-points in the RoI
aPhysicalCentreX	The x-coordinate of the centre of the window in physical units (becomes 0 in algorithm units)
aPhysicalCentreY	The y-coordinate of the centre of the window in physical units (becomes 0 in algorithm units)
aArea	The area of the RoI in physical units

Definition at line 17 of file RoI.cpp.

References mData.

4.14.2.2 RoI() [2/3]

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

Deleted copy constructor.

Parameters

aOther	Anonymous argument

4.14.2.3 RoI() [3/3]

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

Default move constructor.

Parameters

aOther	Anonymous argument

4.14.3 Member Function Documentation

4.14.3.1 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

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

Definition at line 57 of file RoI.cpp.

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

Referenced by RunClustering().

4.14.3.2 data()

```
const std::vector< Data >& RoI::data ( ) const  [inline]
```

Accessor to the raw data.

Returns

Reference to the raw data

Definition at line 89 of file RoI.hpp.

References mData.

4.14.3.3 getArea()

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

Getter for the height of the ROI window.

Returns

The height of the ROI window

Definition at line 82 of file RoI.hpp.

References mArea.

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

4.14.3.4 getCentreX()

```
const 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 68 of file RoI.hpp.

References mPhysicalCentreX.

4.14.3.5 getCentreY()

```
const 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 75 of file RoI.hpp.

References mPhysicalCentreY.

4.14.3.6 id()

```
const std::string& RoI::id ( ) const  [inline]
```

Accessor to the RoI ID.

Returns

Reference to the RoI ID

Definition at line 96 of file RoI.hpp.

References mId.

Referenced by _FullClusterToSimpleCluster_(), and _FullScanToSimpleScan_().

4.14.3.7 operator=() [1/2]

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

Deleted assignment operator.

Returns

Reference to this, for chaining calls

Parameters

aOther	Anonymous argument

4.14.3.8 operator=() [2/2]

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

Default move-assignment constructor.

Returns

Reference to this, for chaining calls

Parameters

aOther	Anonymous argument

4.14.3.9 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

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

Definition at line 34 of file RoI.cpp.

References mData, and range().

Referenced by Clusterize(), and ScanRT().

4.14.3.10 ScanRT()

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

Run the scan.

Parameters

aScanConfig	The configuration parameters for the scan
aCallback	A callback for each RT-scan result

Definition at line 43 of file RoI.cpp.

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

Referenced by _FullScanToSimpleScan_(), and RunScan().

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

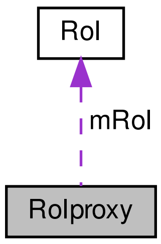
- include/BayesianClustering/RoI.hpp
- src/BayesianClustering/RoI.cpp

4.15 RoIproxy Class Reference

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

```
#include <RoIproxy.hpp>
```

Collaboration diagram for RoIproxy:



Public Member Functions

- RoIproxy (RoI &aRoI)

  Default constructor.
- RoIproxy (const RoIproxy &aOther)=delete

  Deleted copy constructor.
- RoIproxy & operator= (const RoIproxy &aOther)=delete

  Deleted assignment operator.
- RoIproxy (RoIproxy &&aOther)=default

  Default move constructor.
- RoIproxy & operator= (RoIproxy &&aOther)=default

  Default move-assignment constructor.
- ∼RoIproxy ()

  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(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.
- 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 UpdateLogScore (const ScanConfiguration &aScanConfig)

  Update log-probability after a scan.
- void ValidateLogScore (const ScanConfiguration &aScanConfig)

  Sean's validation code for testing when the running log-score fails.
- DataProxy & GetData (const std::size_t &aIndex)

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

Public Attributes

- std::vector< DataProxy > mData

  The collection of lightweight data-point wrappers used by this RoI wrapper.
- std::vector< Cluster > mClusters

  The collection of clusters found by this scan.
- std::size_t mClusteredCount

  The number of clustered data-points.
- std::size_t mBackgroundCount

  The number of background data-points.
- std::size_t mClusterCount

  The number of non-Null clusters.
- double mLogP

  The log-probability density associated with the last scan.
- const RoI & mRoI

  The underlying RoI this is a proxy to.

4.15.1 Detailed Description

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

Definition at line 19 of file RoIproxy.hpp.

4.15.2 Constructor & Destructor Documentation

4.15.2.1 RoIproxy() [1/3]

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

Default constructor.

Parameters

aRoI	An RoI for which this is a lightweight proxy

Definition at line 17 of file RoIproxy.cpp.

References mClusters, RoI::mData, and mData.

4.15.2.2 RoIproxy() [2/3]

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

Deleted copy constructor.

Parameters

aOther	Anonymous argument

4.15.2.3 RoIproxy() [3/3]

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

Default move constructor.

Parameters

aOther	Anonymous argument

4.15.3 Member Function Documentation

4.15.3.1 CheckClusterization()

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

Run validation tests on the clusters.

Parameters

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

Definition at line 34 of file RoIproxy.cpp.

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

4.15.3.2 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

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

Definition at line 139 of file RoIproxy.cpp.

References RoI::getArea(), mClusters, RoI::mData, mData, and mRoI.

Referenced by RoI::Clusterize().

4.15.3.3 GetData()

```
DataProxy& RoIproxy::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

aIndex	The index of the neighbour we are looking for

Definition at line 74 of file RoIproxy.hpp.

References mData.

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

4.15.3.4 operator=() [1/2]

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

Deleted assignment operator.

Returns

Reference to this, for chaining calls

Parameters

aOther	Anonymous argument

4.15.3.5 operator=() [2/2]

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

Default move-assignment constructor.

Returns

Reference to this, for chaining calls

Parameters

aOther	Anonymous argument

4.15.3.6 ScanRT()

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

Run an RT-scan.

Parameters

aScanConfig	The configuration parameters for the scan
aCallback	A callback for each RT-scan result
aProgressBar	The progress bar to update
aParallelization	The stride with which we will iterate across RT parameters
aOffset	The starting point for the strides as we iterate across RT parameters
aValidate	Run validation of the score calculation

Definition at line 104 of file RoIproxy.cpp.

4.15.3.7 UpdateLogScore()

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

Update log-probability after a scan.

Parameters

aScanConfig	The configuration parameters for the scan

Definition at line 219 of file RoIproxy.cpp.

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

4.15.3.8 ValidateLogScore()

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

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

Parameters

aScanConfig	The configuration parameters for the scan

Definition at line 160 of file RoIproxy.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:

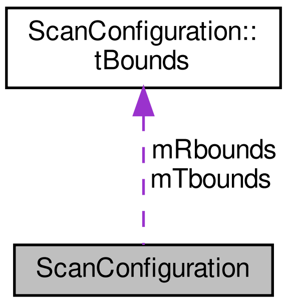
- include/BayesianClustering/RoIproxy.hpp
- src/BayesianClustering/RoIproxy.cpp

4.16 ScanConfiguration Class Reference

A 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 &aSigmaBins, const double &aSigmaMin, const double &aSigmaMax, const std::function< double(const double &) > &aInterpolator, const std::size_t &aRbins, const double &a↩ MinScanR, 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 std::size_t &aSigmaBins, const double &aSigmaMin, const double &aSigmaMax, const std::map< double, double > &aInterpolator, const std::size_t &aRbins, const double &aMinScanR, const double &aMaxScanR, const std::size_t &aTbins, const double &aMinScanT, const double &aMax↩ ScanT, 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 &aSigmaBins, const double &aSigmaMin, const double &a↩ SigmaMax, 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 gievn 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.

4.16.1 Detailed Description

A class for storing the scan configuration parameters.

Definition at line 12 of file Configuration.hpp.

4.16.2 Constructor & Destructor Documentation

4.16.2.1 ScanConfiguration() [1/5]

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

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

Parameters

aCfgFile	A Scan-parameter config file name

Definition at line 41 of file Configuration.cpp.

References FromVector().

4.16.2.2 ScanConfiguration() [2/5]

```
ScanConfiguration::ScanConfiguration (
            const std::size_t & aSigmaBins,
            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

aSigmaBins	The number of sigma bins
aSigmaMin	The lowest sigma bin
aSigmaMax	The highest sigma bin
aInterpolator	Function-object to generate the probability of any given sigma
aRbins	The number of R bins to scan over
aMinScanR	The lowest value of R to scan
aMaxScanR	The largest value of R to scan
aTbins	The number of T bins to scan over
aMinScanT	The lowest value of T to scan
aMaxScanT	The largest value of T to scan
aPB	The P_b parameter
aAlpha	The alpha parameter

Definition at line 55 of file Configuration.cpp.

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

4.16.2.3 ScanConfiguration() [3/5]

```
ScanConfiguration::ScanConfiguration (
            const std::size_t & aSigmaBins,
            const double & aSigmaMin,
            const double & aSigmaMax,
            const std::map< double, 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

aSigmaBins	The number of sigma bins
aSigmaMin	The lowest sigma bin
aSigmaMax	The highest sigma bin
aInterpolator	A set of points from which to create an interpolator
aRbins	The number of R bins to scan over
aMinScanR	The lowest value of R to scan
aMaxScanR	The largest value of R to scan
aTbins	The number of T bins to scan over
aMinScanT	The lowest value of T to scan
aMaxScanT	The largest value of T to scan
aPB	The P_b parameter
aAlpha	The alpha parameter

Definition at line 69 of file Configuration.cpp.

References GSLInterpolator::Eval(), SetAlpha(), SetPb(), SetRBins(), SetSigmaParameters(), and SetTBins().

4.16.2.4 ScanConfiguration() [4/5]

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

Deleted copy constructor.

Parameters

aOther	Anonymous argument

4.16.2.5 ScanConfiguration() [5/5]

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

Default move constructor.

Parameters

aOther	Anonymous argument

4.16.3 Member Function Documentation

4.16.3.1 alpha()

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

Getter for the alpha parameter.

Returns

The alpha parameter

Definition at line 232 of file Configuration.hpp.

References mAlpha.

Referenced by RoIproxy::UpdateLogScore().

4.16.3.2 FromVector()

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

Parse the parameters when passed in as commandline arguments.

Parameters

aArgs	The commandline arguments

Definition at line 179 of file Configuration.cpp.

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

Referenced by ScanConfiguration().

4.16.3.3 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 170 of file Configuration.hpp.

References mLogProbabilitySigma.

Referenced by Cluster::UpdateLogScore().

4.16.3.4 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	The index of the value of sigma to get the log-probability for

Returns

The log-probability of sigma_i

Definition at line 199 of file Configuration.hpp.

References mLogProbabilitySigma.

4.16.3.5 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 238 of file Configuration.hpp.

References mLogAlpha.

Referenced by RoIproxy::UpdateLogScore().

4.16.3.6 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 244 of file Configuration.hpp.

References mLogGammaAlpha.

Referenced by RoIproxy::UpdateLogScore().

4.16.3.7 logPb()

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

Logarithm of the P_b parameter.

Returns

Logarithm of the P_b parameter

Definition at line 219 of file Configuration.hpp.

References mLogPb.

Referenced by RoIproxy::UpdateLogScore().

4.16.3.8 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 225 of file Configuration.hpp.

References mLogPbDagger.

Referenced by RoIproxy::UpdateLogScore().

4.16.3.9 operator=() [1/2]

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

Deleted assignment operator.

Returns

Reference to this, for chaining calls

Parameters

aOther	Anonymous argument

4.16.3.10 operator=() [2/2]

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

Default move-assignment constructor.

Returns

Reference to this, for chaining calls

Parameters

aOther	Anonymous argument

4.16.3.11 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 164 of file Configuration.hpp.

References mProbabilitySigma.

4.16.3.12 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	The index of the value of sigma to get the probability for

Returns

The probability of sigma_i

Definition at line 192 of file Configuration.hpp.

References mProbabilitySigma.

4.16.3.13 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 206 of file Configuration.hpp.

References mRbounds.

Referenced by RoI::ScanRT().

4.16.3.14 SetAlpha()

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

Setter for the alpha parameter.

Parameters

aAlpha	The alpha parameter

Definition at line 141 of file Configuration.cpp.

References mAlpha, mLogAlpha, and mLogGammaAlpha.

Referenced by FromVector(), and ScanConfiguration().

4.16.3.15 SetPb()

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

Setter for the P_b parameter.

Parameters

aPB	The P_b parameter

Definition at line 134 of file Configuration.cpp.

References mLogPb, and mLogPbDagger.

Referenced by FromVector(), and ScanConfiguration().

4.16.3.16 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

aRbins	The number of R bins to scan over
aMinScanR	The lowest value of R to scan
aMaxScanR	The largest value of R to scan

Definition at line 114 of file Configuration.cpp.

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

Referenced by FromVector(), and ScanConfiguration().

4.16.3.17 SetSigmaParameters()

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

Setter for the sigma-bins to be integrated over.

Parameters

aSigmaBins	The number of sigma bins
aSigmaMin	The lowest sigma bin
aSigmaMax	The highest sigma bin
aInterpolator	Function-object to generate the probability of any given sigma

Definition at line 86 of file Configuration.cpp.

References mLogProbabilitySigma, mProbabilitySigma, mSigmabins, mSigmabins2, mSigmacount, m↩
Sigmaspacing, and range().

Referenced by FromVector(), and ScanConfiguration().

4.16.3.18 SetTBins()

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

Parameters

aTbins	The number of T bins to scan over
aMinScanT	The lowest value of T to scan
aMaxScanT	The largest value of T to scan

Definition at line 124 of file Configuration.cpp.

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

Referenced by FromVector(), and ScanConfiguration().

4.16.3.19 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 152 of file Configuration.hpp.

References mSigmabins.

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

4.16.3.20 sigmabins() [2/2]

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

Getter for the i'th value of sigma.

Parameters

i	The index of the value of sigma to get

Returns

The value of sigma_i

Definition at line 178 of file Configuration.hpp.

References mSigmabins.

4.16.3.21 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 158 of file Configuration.hpp.

References mSigmabins2.

Referenced by RoI::ScanRT(), and RoIproxy::ValidateLogScore().

4.16.3.22 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	The index of the value of sigma squared to get

Returns

The value of sigma_i squared

Definition at line 185 of file Configuration.hpp.

References mSigmabins2.

4.16.3.23 sigmacount()

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

Getter for the sigma count.

Returns

The sigma count

Definition at line 138 of file Configuration.hpp.

References mSigmacount.

Referenced by RoIproxy::ValidateLogScore().

4.16.3.24 sigmaspacing()

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

Getter for the sigma spacing.

Returns

The sigma spacing

Definition at line 145 of file Configuration.hpp.

References mSigmaspacing.

4.16.3.25 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 212 of file Configuration.hpp.

References mTbounds.

Referenced by RoI::ScanRT().

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

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

4.17 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.

4.17.1 Detailed Description

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

Definition at line 19 of file API.hpp.

4.17.2 Member Function Documentation

4.17.2.1 operator<()

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

Comparison operator for sorting.

Returns

Whether we are smaller than the other

Parameters

aOther	Another ScanEntry to compare against

Definition at line 27 of file API.hpp.

References r, and t.

4.17.2.2 operator==()

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

Equality operator required by boost python.

Returns

Whether we are equal to the other

Parameters

aOther	Another ScanEntry to compare against

Definition at line 36 of file API.hpp.

References r, score, and t.

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

- include/BayesianClustering/API.hpp

4.18 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.

4.18.1 Detailed Description

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

Definition at line 17 of file Configuration.hpp.

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

- include/BayesianClustering/Configuration.hpp

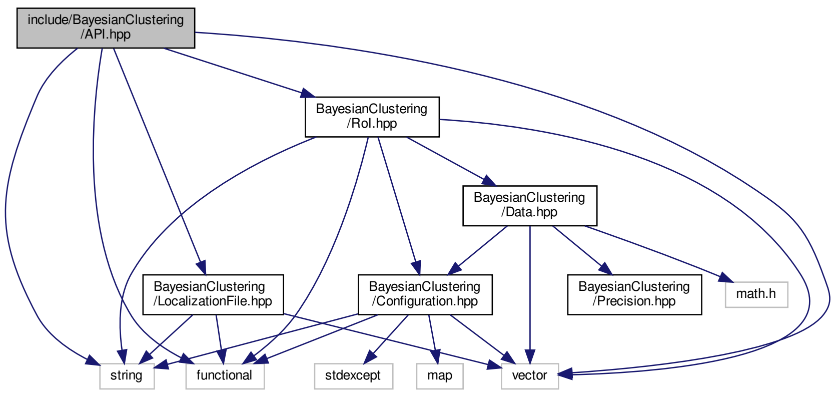
Chapter 5

File Documentation

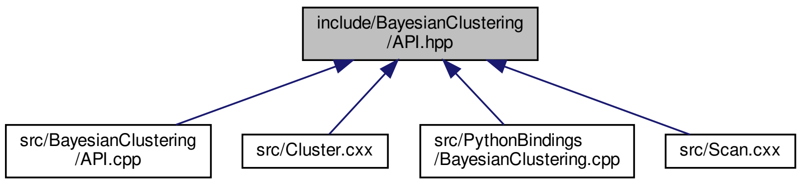
5.1 include/BayesianClustering/API.hpp File Reference

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

Include dependency graph for API.hpp:


include/BayesianClustering/API.hpp
BayesianClustering/RoI.hpp
BayesianClustering/Data.hpp
BayesianClustering/LocalizationFile.hpp
BayesianClustering/Configuration.hpp
BayesianClustering/Precision.hpp
math.h
string
functional
stdexcept
map
vector

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


include/BayesianClustering/API.hpp
src/BayesianClustering/API.cpp
src/Cluster.cxx
src/PythonBindings/BayesianClustering.cpp
src/Scan.cxx

Classes

- struct ScanEntry

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

  A struct for storing extracted parameters from a cluster.

Typedefs

- typedef std::function< void(RoIproxy &, const double &, const double &) > tFullScanCallback

  Typedef the full scan callback for clarity.
- typedef std::function< void(const std::string &, const std::vector< ScanEntry > &) > tSimpleScanCallback

  Typedef the simplified scan callback for clarity.
- typedef std::function< void(RoIproxy &) > tFullClusterCallback

  Typedef the full clustering callback for clarity.
- typedef std::function< void(const std::string &, const std::vector< ClusterWrapper > &) > tSimpleClusterCallback

  Typedef the simplified clustering callback for clarity.

Functions

- void _ScanCallback_Json_ (const std::string &aRoiId, const std::vector< ScanEntry > &aVector, const std↩ ::string &aInFile, const std::string &aOutputPattern)

  A callback to dump a scan to a JSON file.
- void _FullScanToSimpleScan_ (RoI &aRoI, const ScanConfiguration &aScanConfig, const tSimpleScanCallback &aCallback)

  A callback to neatly package the scan results for easy consumption.
- void _ClusterCallback_Json_ (const std::string &aRoiId, const std::vector< ClusterWrapper > &aVector, const std::string &aInFile, const std::string &aOutputPattern)

  A callback to dump a clustering run to a JSON file.
- void _FullClusterToSimpleCluster_ (RoIproxy &aRoIproxy, const tSimpleClusterCallback &aCallback)

  A callback to neatly package the scan results for easy consumption.
- template<typename RoIConfig >
  void RunScan (const std::string &aInFile, const RoIConfig &aRoIConfig, const ScanConfiguration &aScan↩ Config, const tFullScanCallback &aCallback)

  Automatically extract RoI, run scan and apply a full call-back.
- template<typename RoIConfig >
  void RunScan (const std::string &aInFile, const RoIConfig &aRoIConfig, const ScanConfiguration &aScan↩ Config, const tSimpleScanCallback &aCallback)

  Automatically extract RoI, run scan and apply a simple call-back.
- template<typename RoIConfig >
  void RunScan (const std::string &aInFile, const RoIConfig &aRoIConfig, const ScanConfiguration &aScan↩ Config, const std::string &aOutputPattern)

  Automatically extract RoI, run scan and dump to JSON file.
- template<typename RoIConfig >
  void RunClustering (const std::string &aInFile, const RoIConfig &aRoIConfig, const double &aR, const double &aT, const tFullClusterCallback &aCallback)

  Automatically extract RoI, clusterize and apply a full call-back.
- template<typename RoIConfig >
  void RunClustering (const std::string &aInFile, const RoIConfig &aRoIConfig, const double &aR, const double &aT, const tSimpleClusterCallback &aCallback)

  Automatically extract RoI, clusterize and apply a full call-back.
- template<typename RoIConfig >
  void RunClustering (const std::string &aInFile, const RoIConfig &aRoIConfig, const double &aR, const double &aT, const std::string &aOutputPattern)

  Automatically specify RoI, clusterize and apply a full call-back.

5.1.1 Function Documentation

5.1.1.1 _ClusterCallback_Json_()

```
void _ClusterCallback_Json_ (
            const std::string & aRoiId,
            const std::vector< ClusterWrapper > & aVector,
            const std::string & aInFile,
            const std::string & aOutputPattern )
```

A callback to dump a clustering run to a JSON file.

Parameters

aRoiId	The RoI ID
aVector	A vector of cluster-wrappers
aInFile	The name of the localization file
aOutputPattern	The name of the output JSON file

Definition at line 87 of file API.cpp.

Referenced by RunClustering().

5.1.1.2 _FullClusterToSimpleCluster_()

```
void _FullClusterToSimpleCluster_ (
            RoIproxy & aRoIproxy,
            const tSimpleClusterCallback & aCallback )
```

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

Parameters

aRoIproxy	The region-proxy containing the clusters
aCallback	The simple callback to be applied

Definition at line 106 of file API.cpp.

References RoI::id(), RoIproxy::mData, and RoIproxy::mRoI.

Referenced by RunClustering().

5.1.1.3 _FullScanToSimpleScan_()

```
void _FullScanToSimpleScan_ (
             RoI & aRoI,
             const ScanConfiguration & aScanConfig,
             const tSimpleScanCallback & aCallback )
```

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

Parameters

aRoI	The region of interest
aScanConfig	The configuration for the scan
aCallback	The simple callback to be applied

Definition at line 72 of file API.cpp.

References RoI::id(), and RoI::ScanRT().

Referenced by RunScan().

5.1.1.4 _ScanCallback_Json_()

```
void _ScanCallback_Json_ (
             const std::string & aRoiId,
             const std::vector< ScanEntry > & aVector,
             const std::string & aInFile,
             const std::string & aOutputPattern )
```

A callback to dump a scan to a JSON file.

Parameters

aRoiId	The RoI ID
aVector	A vector of scan results
aInFile	The name of the localization file
aOutputPattern	The name of the output JSON file

Definition at line 26 of file API.cpp.

Referenced by RunScan().

5.1.1.5 RunClustering() [1/3]

```
template<typename RoIConfig >
void RunClustering (
             const std::string & aInFile,
             const RoIConfig & aRoIConfig,
             const double & aR,
             const double & aT,
             const std::string & aOutputPattern )
```

Automatically specify RoI, clusterize and apply a full call-back.

Parameters

aInFile	The name of the localization file
aRoIConfig	Specify the mechanism used to extract the RoIs
aR	The R value of the clusterizer
aT	The T value of the clusterizer
aOutputPattern	A formattable-string specifying the name of the output JSON files. Substitutable fields are {input} (giving the stem of the input file name) and {roi} (giving the RoI id).

Definition at line 181 of file API.hpp.

References _ClusterCallback_Json_(), and RunClustering().

5.1.1.6 RunClustering() [2/3]

```
template<typename RoIConfig >
void RunClustering (
                const std::string & aInFile,
                const RoIConfig & aRoIConfig,
                const double & aR,
                const double & aT,
                const tFullClusterCallback & aCallback )
```

Automatically extract RoI, clusterize and apply a full call-back.

Parameters

aInFile	The name of the localization file
aRoIConfig	Specify the mechanism used to extract the RoIs
aR	The R value of the clusterizer
aT	The T value of the clusterizer
aCallback	The callback to be applied

Definition at line 157 of file API.hpp.

References RoI::Clusterize(), and LocalizationFile::ExtractRoIs().

Referenced by main(), and RunClustering().

5.1.1.7 RunClustering() [3/3]

```
template<typename RoIConfig >
void RunClustering (
             const std::string & aInFile,
             const RoIConfig & aRoIConfig,
             const double & aR,
             const double & aT,
             const tSimpleClusterCallback & aCallback )
```

Automatically extract RoI, clusterize and apply a full call-back.

Parameters

aInFile	The name of the localization file
aRoIConfig	Specify the mechanism used to extract the RoIs
aR	The R value of the clusterizer
aT	The T value of the clusterizer
aCallback	The callback to be applied

Definition at line 169 of file API.hpp.

References _FullClusterToSimpleCluster_(), and RunClustering().

5.1.1.8 RunScan() [1/3]

```
template<typename RoIConfig >
void RunScan (
             const std::string & aInFile,
             const RoIConfig & aRoIConfig,
             const ScanConfiguration & aScanConfig,
             const std::string & aOutputPattern )
```

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

Parameters

aInFile	The name of the localization file
aRoIConfig	Specify the mechanism used to extract the RoIs
aScanConfig	The configuration for the scan
aOutputPattern	A formattable-string specifying the name of the output JSON files. Substitutable fields are {input} (giving the stem of the input file name) and {roi} (giving the RoI id).

Definition at line 142 of file API.hpp.

References _ScanCallback_Json_(), and RunScan().

5.1.1.9 RunScan() [2/3]

```
template<typename RoIConfig >
void RunScan (
             const std::string & aInFile,
             const RoIConfig & aRoIConfig,
             const ScanConfiguration & aScanConfig,
             const tFullScanCallback & aCallback )
```

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

Parameters

aInFile	The name of the localization file
aRoIConfig	Specify the mechanism used to extract the RoIs
aScanConfig	The configuration for the scan
aCallback	The full callback to be applied

Definition at line 120 of file API.hpp.

References LocalizationFile::ExtractRoIs(), and RoI::ScanRT().

Referenced by main(), and RunScan().

5.1.1.10 RunScan() [3/3]

```
template<typename RoIConfig >
void RunScan (
             const std::string & aInFile,
             const RoIConfig & aRoIConfig,
             const ScanConfiguration & aScanConfig,
             const tSimpleScanCallback & aCallback )
```

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

Parameters

aInFile	The name of the localization file
aRoIConfig	Specify the mechanism used to extract the RoIs
aScanConfig	The configuration for the scan
aCallback	The simple callback to be applied

Definition at line 131 of file API.hpp.

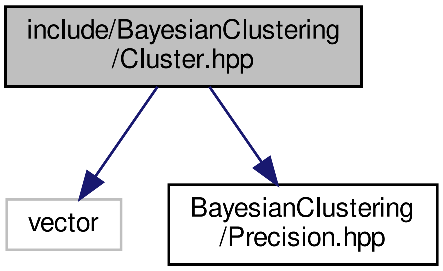
References _FullScanToSimpleScan_(), and LocalizationFile::ExtractRoIs().

5.2 include/BayesianClustering/Cluster.hpp File Reference

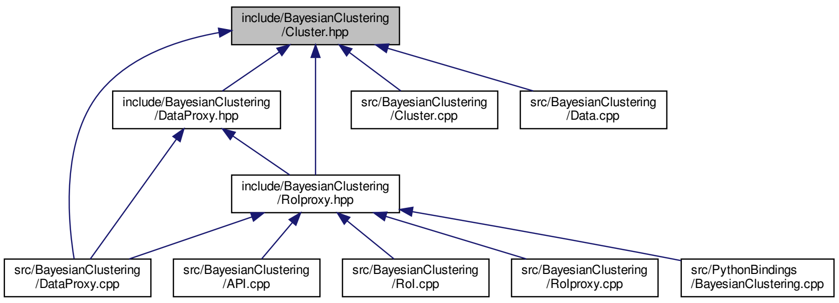
```
#include <vector>
```

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

Include dependency graph for Cluster.hpp:



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



Classes

- class Cluster

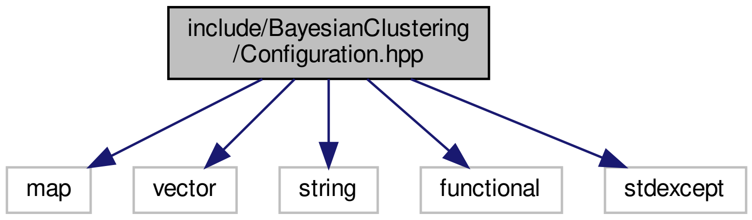
  A class representing a cluster.
- struct Cluster::Parameter

  A struct representing the cluster parameters.

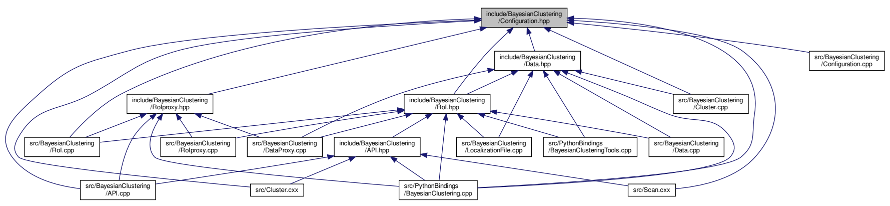
5.3 include/BayesianClustering/Configuration.hpp File Reference

```
#include <map>
#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

  A 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

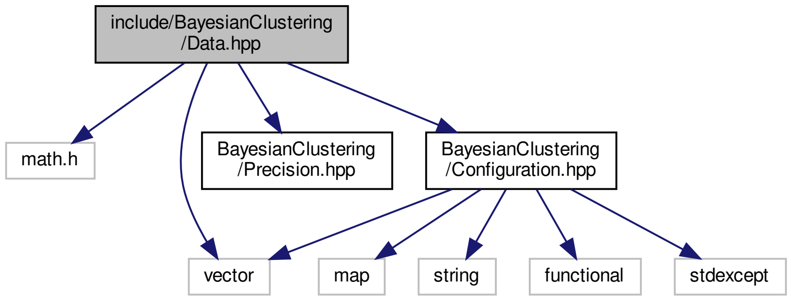
  Class for storing the auxilliary configuration parameters.

5.4 include/BayesianClustering/Data.hpp File Reference

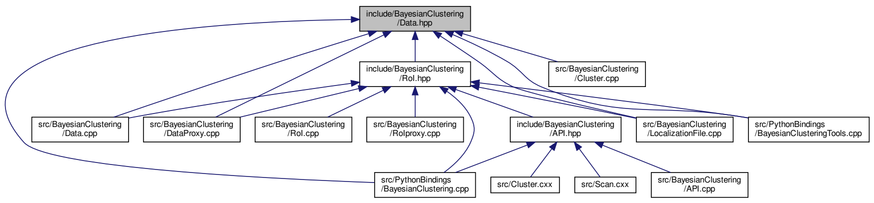
```
#include <math.h>
#include <vector>
#include "BayesianClustering/Precision.hpp"
```

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

Include dependency graph for Data.hpp:



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



Classes

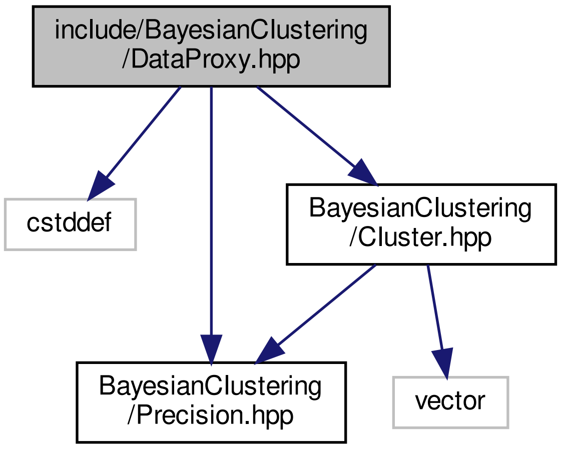
- class Data

  A class to store the raw data-points.

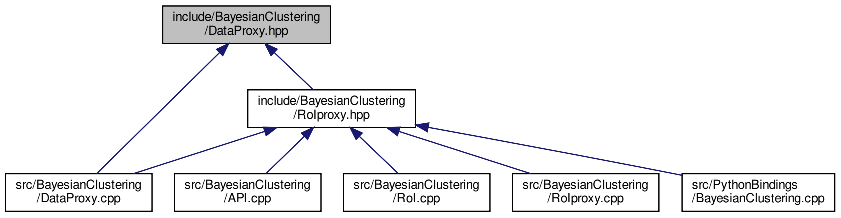
5.5 include/BayesianClustering/DataProxy.hpp File Reference

```
#include <cstddef>
#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

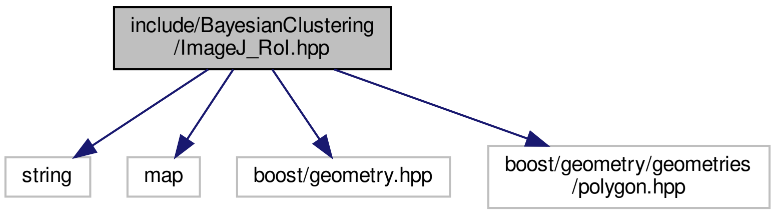
    A light-weight proxy for the raw data-points.

5.6 include/BayesianClustering/ImageJ_RoI.hpp File Reference

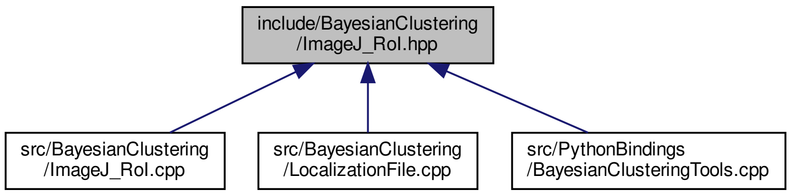
```
#include <string>
#include <map>
#include <boost/geometry.hpp>
```

`#include <boost/geometry/geometries/polygon.hpp>`

Include dependency graph for ImageJ_RoI.hpp:



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



Typedefs

- typedef boost::geometry::model::point< uint16_t, 2, boost::geometry::cs::cartesian > roi_point

  Typedef a boost::geometry type representing an ImageJ Roi point for simplicity.
- typedef boost::geometry::model::ring< roi_point > roi_polygon

  Typedef a boost::geometry type representing an ImageJ Roi polygon for simplicity.

Functions

- roi_polygon DecodeBinaryRoI (const uint8_t ∗const aData)

  Decode ImageJ binary RoI data to boost::geometry polygon Reverse engineered from `https://github.↵com/imagej/ImageJ/blob/master/ij/io/RoiDecoder.java`.
- std::map< std::string, roi_polygon > OpenRoiZipfile (const std::string &aZipFileName)

  Decode ImageJ zipped binary RoI data to a map of named boost::geometry polygons.

5.6.1 Function Documentation

5.6.1.1 DecodeBinaryRoI()

```
roi_polygon DecodeBinaryRoI (
            const uint8_t *const aData )
```

Decode ImageJ binary RoI data to boost::geometry polygon Reverse engineered from `https://github.↩ com/imagej/ImageJ/blob/master/ij/io/RoiDecoder.java`.

Parameters

aData	A C-array containing the binary RoI data

Returns

A boost::geometry polygon containing the RoI information

Definition at line 17 of file ImageJ_RoI.cpp.

Referenced by OpenRoiZipfile().

5.6.1.2 OpenRoiZipfile()

```
std::map< std::string , roi_polygon > OpenRoiZipfile (
            const std::string & aZipFileName )
```

Decode ImageJ zipped binary RoI data to a map of named boost::geometry polygons.

Parameters

aZipFileName	The name of the zip-file to be opened

Returns

A map of named boost::geometry polygons containing the RoI information

Definition at line 31 of file ImageJ_RoI.cpp.

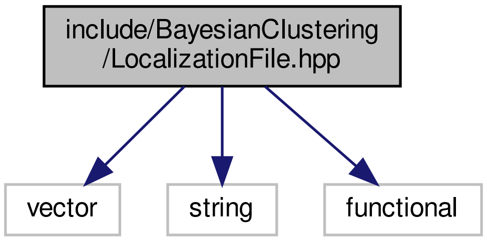
References DecodeBinaryRoI().

Referenced by LocalizationFile::ExtractRoIs(), and GetRoIs().

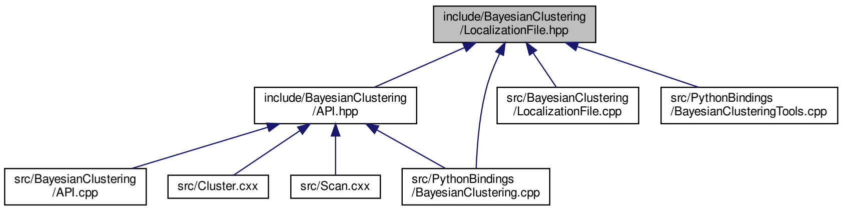
5.7 include/BayesianClustering/LocalizationFile.hpp File Reference

```
#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 ManualRoI

    A struct for storing the parameters of a manual RoI.
- struct AutoRoI

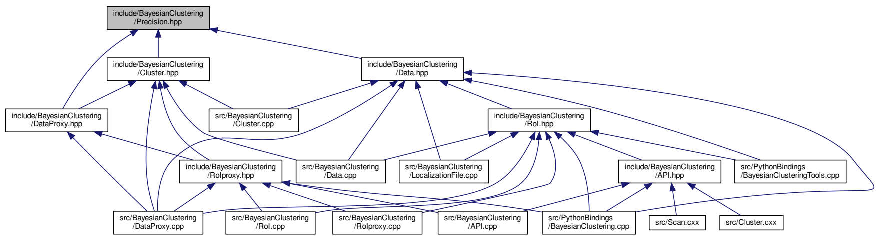
    A struct for storing the parameters for automatically extracting the RoIs.
- struct ImageJRoI

    A struct for storing the parameters of an ImageJ RoI file.
- class LocalizationFile

    A class to store the raw data-points.

5.8 include/BayesianClustering/Precision.hpp File Reference

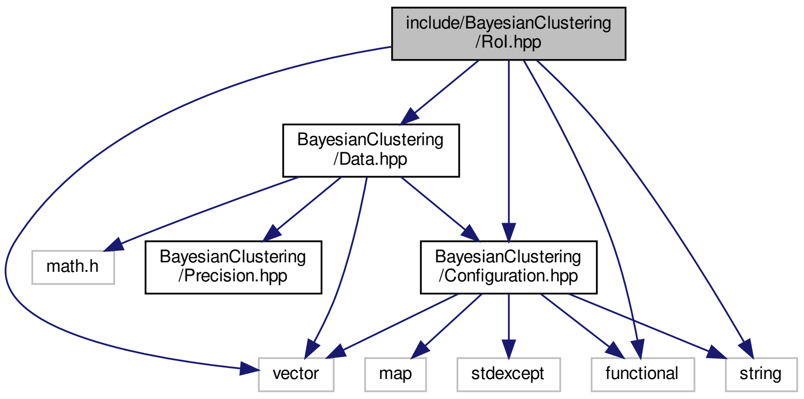
This graph shows which files directly or indirectly include this file:


include/BayesianClustering/Precision.hpp
include/BayesianClustering/Cluster.hpp
include/BayesianClustering/Data.hpp
include/BayesianClustering/DataProxy.hpp
src/BayesianClustering/Cluster.cpp
include/BayesianClustering/RoI.hpp
include/BayesianClustering/RoIproxy.hpp
src/BayesianClustering/Data.cpp
src/BayesianClustering/LocalizationFile.cpp
include/BayesianClustering/API.hpp
src/PythonBindings/BayesianClusteringTools.cpp
src/BayesianClustering/DataProxy.cpp
src/BayesianClustering/RoI.cpp
src/BayesianClustering/RoIproxy.cpp
src/BayesianClustering/API.cpp
src/PythonBindings/BayesianClustering.cpp
src/Scan.cxx
src/Cluster.cxx

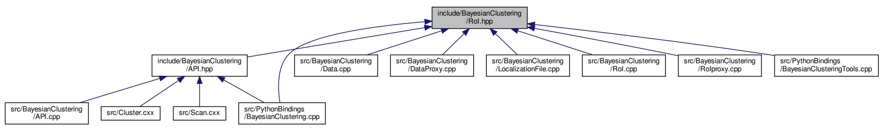
5.9 include/BayesianClustering/RoI.hpp File Reference

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

Include dependency graph for RoI.hpp:


include/BayesianClustering/RoI.hpp
BayesianClustering/Data.hpp
math.h
BayesianClustering/Precision.hpp
BayesianClustering/Configuration.hpp
vector
map
stdexcept
functional
string

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


include/BayesianClustering/RoI.hpp
include/BayesianClustering/API.hpp
src/BayesianClustering/Data.cpp
src/BayesianClustering/DataProxy.cpp
src/BayesianClustering/LocalizationFile.cpp
src/BayesianClustering/RoI.cpp
src/BayesianClustering/RoIproxy.cpp
src/PythonBindings/BayesianClusteringTools.cpp
src/BayesianClustering/API.cpp
src/Cluster.cxx
src/Scan.cxx
src/PythonBindings/BayesianClustering.cpp

Classes

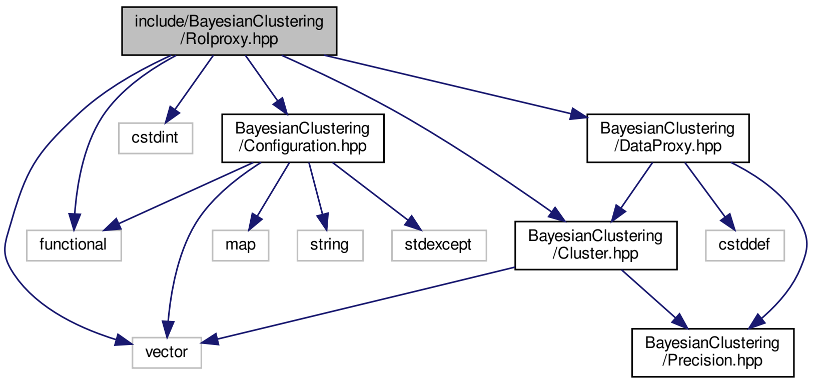
- class RoI

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

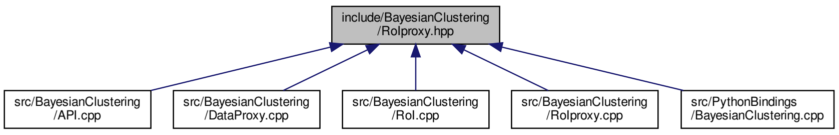
5.10 include/BayesianClustering/RoIproxy.hpp File Reference

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

Include dependency graph for RoIproxy.hpp:


include/BayesianClustering/RoIproxy.hpp
cstdint
BayesianClustering/Configuration.hpp
BayesianClustering/DataProxy.hpp
functional
map
string
stdexcept
BayesianClustering/Cluster.hpp
cstddef
vector
BayesianClustering/Precision.hpp

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


include/BayesianClustering/RoIproxy.hpp
src/BayesianClustering/API.cpp
src/BayesianClustering/DataProxy.cpp
src/BayesianClustering/RoI.cpp
src/BayesianClustering/RoIproxy.cpp
src/PythonBindings/BayesianClustering.cpp

Classes

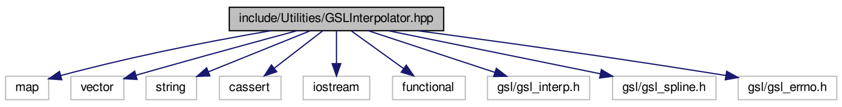
- class RoIproxy

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

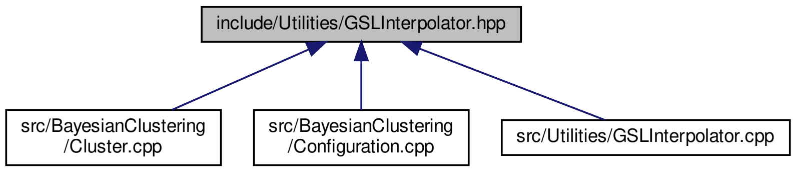
5.11 include/Utilities/GSLInterpolator.hpp File Reference

```
#include <map>
#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:


include/Utilities/GSLInterpolator.hpp
src/BayesianClustering /Cluster.cpp
src/BayesianClustering /Configuration.cpp
src/Utilities/GSLInterpolator.cpp

Classes

- class GSLInterpolator

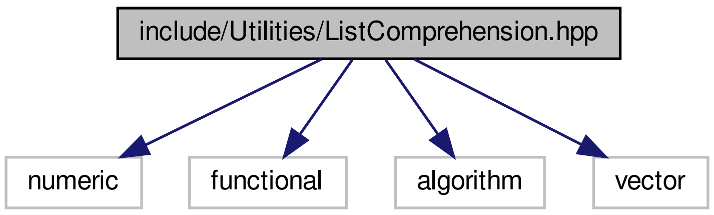
  A utility wrapper around the GSL interpolator to give it a clean C++ interface.

5.12 include/Utilities/ListComprehension.hpp File Reference

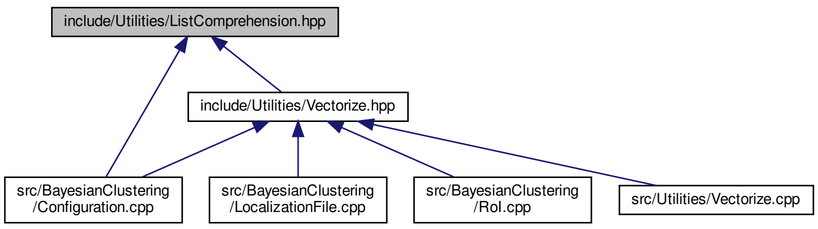
```
#include <numeric>
#include <functional>
#include <algorithm>
```

```
#include <vector>
```

Include dependency graph for ListComprehension.hpp:



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


include/Utilities/ListComprehension.hpp
include/Utilities/Vectorize.hpp
src/BayesianClustering /Configuration.cpp
src/BayesianClustering /LocalizationFile.cpp
src/BayesianClustering /RoI.cpp
src/Utilities/Vectorize.cpp

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 &&a↩
  Container)

  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.

5.12.1 Function Documentation

5.12.1.1 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

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

Parameters

aExpr	A function-call to be applied to each element of the container
aContainer	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.

5.12.1.2 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

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

Parameters

aExpr	A function-call to be applied to each element of the container
aContainer	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.

5.12.1.3 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

tType	A container type
tContainerType	Template magic to determine the type of the data in the container

Parameters

aPtr	A pointer-to-member-variable to be applied to each element of the container
aContainer	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.

5.12.1.4 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

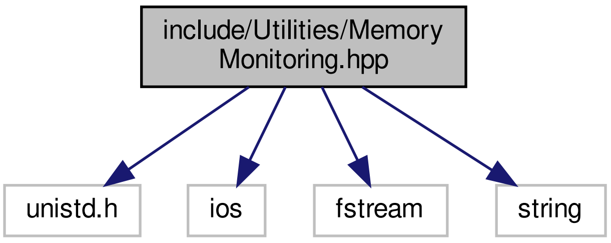
A vector of ints

Definition at line 70 of file ListComprehension.hpp.

Referenced by LocalizationFile::LocalizationFile(), RoI::Preprocess(), RoI::ScanRT(), and ScanConfiguration::Set↩
SigmaParameters().

5.13 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.

5.13.1 Function Documentation

5.13.1.1 mem_usage()

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

Utility to get Virtual Memory and Resident Set usage.

Parameters

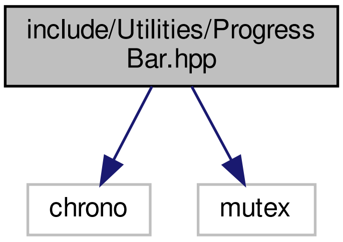
vm_usage	Return the Virtual Memory usage
resident_set	Return the Resident Set usage

Definition at line 12 of file MemoryMonitoring.hpp.

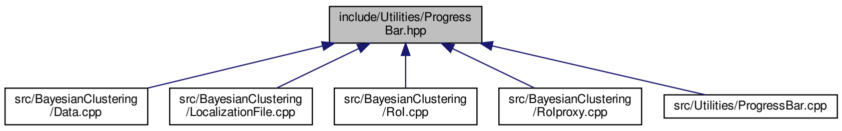
5.14 include/Utilities/ProgressBar.hpp File Reference

```
#include <chrono>
#include <mutex>
```

Include dependency graph for ProgressBar.hpp:



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



Classes

- class ProgressBar

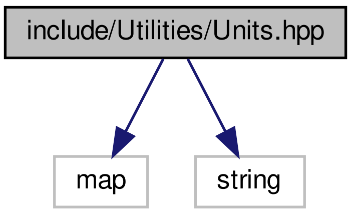
    A utility progress-bar.
- struct ProgressTimer

    A utility code timer.

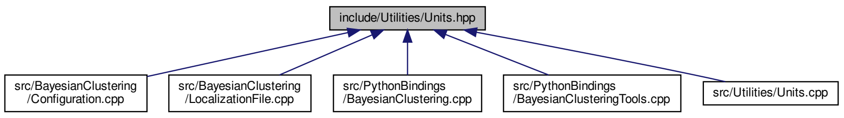
5.15 include/Utilities/Units.hpp File Reference

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

Include dependency graph for Units.hpp:



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


include/Utilities/Units.hpp
src/BayesianClustering /Configuration.cpp
src/BayesianClustering /LocalizationFile.cpp
src/PythonBindings /BayesianClustering.cpp
src/PythonBindings /BayesianClusteringTools.cpp
src/Utilities/Units.cpp

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.

5.15.1 Function Documentation

5.15.1.1 operator""""_micrometer() [1/2]

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

User-defined literals for micrometer quantities.

Parameters

aVal	The specified value

Returns

The literal value

Definition at line 39 of file Units.hpp.

References micrometer.

5.15.1.2 operator""""_micrometer() [2/2]

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

User-defined literals for micrometer quantities.

Parameters

aVal	The specified value

Returns

The literal value

Definition at line 47 of file Units.hpp.

References micrometer.

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

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

User-defined literals for nanometer quantities.

Parameters

aVal	The specified value

Returns

The literal value

Definition at line 23 of file Units.hpp.

References nanometer.

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

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

User-defined literals for nanometer quantities.

Parameters

aVal	The specified value

Returns

The literal value

Definition at line 31 of file Units.hpp.

References nanometer.

5.15.1.5 StrToDist()

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

Convert a string representation to a distance.

Parameters

aStr	A string representation of a distance

Returns

The literal value

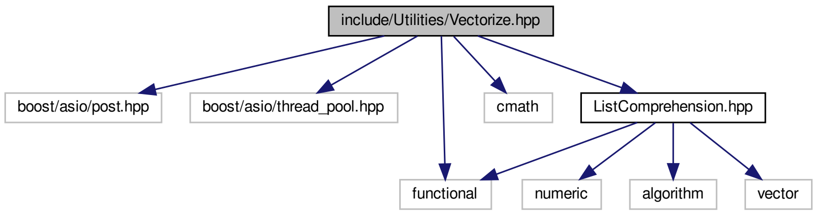
Definition at line 12 of file Units.cpp.

References UnitMap.

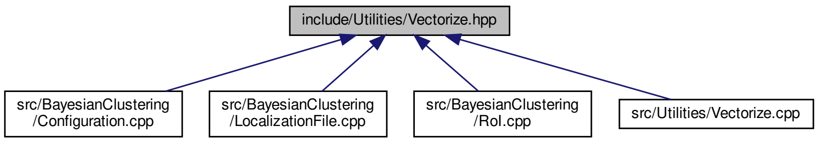
Referenced by ScanConfiguration::FromVector(), and AuxConfiguration::FromVector().

5.16 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:


include/Utilities/Vectorize.hpp
src/BayesianClustering /Configuration.cpp
src/BayesianClustering /LocalizationFile.cpp
src/BayesianClustering /RoI.cpp
src/Utilities/Vectorize.cpp

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.

5.16.1 Function Documentation

5.16.1.1 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

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

Parameters

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

Definition at line 38 of file Vectorize.hpp.

References Nthreads.

5.16.1.2 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

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

Parameters

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

Definition at line 22 of file Vectorize.hpp.

References Nthreads.

5.16.2 Variable Documentation

5.16.2.1 Nthreads

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

Utility variable for the concurrency.

Utility variable for the concurrency.

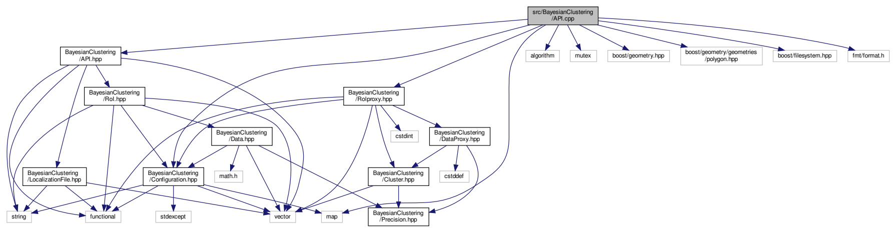
Definition at line 8 of file Vectorize.cpp.

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

5.17 src/BayesianClustering/API.cpp File Reference

```
#include "BayesianClustering/API.hpp"
#include "BayesianClustering/Configuration.hpp"
#include "BayesianClustering/RoIproxy.hpp"
#include <map>
#include <algorithm>
#include <mutex>
#include <boost/geometry.hpp>
#include <boost/geometry/geometries/polygon.hpp>
#include <boost/filesystem.hpp>
#include <fmt/format.h>
```

Include dependency graph for API.cpp:


src/BayesianClustering/API.cpp
BayesianClustering/API.hpp
algorithm
mutex
boost/geometry.hpp
boost/geometry/geometries/polygon.hpp
boost/filesystem.hpp
fmt/format.h
BayesianClustering/RoI.hpp
BayesianClustering/RoIproxy.hpp
BayesianClustering/Data.hpp
cstdint
BayesianClustering/DataProxy.hpp
BayesianClustering/LocalizationFile.hpp
BayesianClustering/Configuration.hpp
math.h
BayesianClustering/Cluster.hpp
cstddef
string
functional
stdexcept
vector
map
BayesianClustering/Precision.hpp

Functions

- void _ScanCallback_Json_ (const std::string &aRoiId, const std::vector< ScanEntry > &aVector, const std↩ ::string &aInFile, const std::string &aOutputPattern)

  A callback to dump a scan to a JSON file.
- void _FullScanToSimpleScan_ (RoI &aRoI, const ScanConfiguration &aScanConfig, const tSimpleScanCallback &aCallback)

  A callback to neatly package the scan results for easy consumption.
- void _ClusterCallback_Json_ (const std::string &aRoiId, const std::vector< ClusterWrapper > &aVector, const std::string &aInFile, const std::string &aOutputPattern)

  A callback to dump a clustering run to a JSON file.
- void _FullClusterToSimpleCluster_ (RoIproxy &aRoIproxy, const tSimpleClusterCallback &aCallback)

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

5.17.1 Function Documentation

5.17.1.1 _ClusterCallback_Json_()

```
void _ClusterCallback_Json_ (
            const std::string & aRoiId,
            const std::vector< ClusterWrapper > & aVector,
            const std::string & aInFile,
            const std::string & aOutputPattern )
```

A callback to dump a clustering run to a JSON file.

Parameters

aRoiId	The RoI ID
aVector	A vector of cluster-wrappers
aInFile	The name of the localization file
aOutputPattern	The name of the output JSON file

Definition at line 87 of file API.cpp.

Referenced by RunClustering().

5.17.1.2 _FullClusterToSimpleCluster_()

```
void _FullClusterToSimpleCluster_ (
            RoIproxy & aRoIproxy,
            const tSimpleClusterCallback & aCallback )
```

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

Parameters

aRoIproxy	The region-proxy containing the clusters
aCallback	The simple callback to be applied

Definition at line 106 of file API.cpp.

References RoI::id(), RoIproxy::mData, and RoIproxy::mRoI.

Referenced by RunClustering().

5.17.1.3 _FullScanToSimpleScan_()

```
void _FullScanToSimpleScan_ (
            RoI & aRoI,
            const ScanConfiguration & aScanConfig,
            const tSimpleScanCallback & aCallback )
```

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

Parameters

aRoI	The region of interest
aScanConfig	The configuration for the scan
aCallback	The simple callback to be applied

Definition at line 72 of file API.cpp.

References RoI::id(), and RoI::ScanRT().

Referenced by RunScan().

5.17.1.4 _ScanCallback_Json_()

```
void _ScanCallback_Json_ (
            const std::string & aRoiId,
            const std::vector< ScanEntry > & aVector,
            const std::string & aInFile,
            const std::string & aOutputPattern )
```

A callback to dump a scan to a JSON file.

Parameters

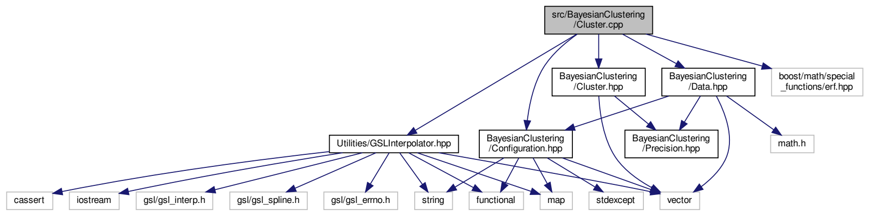
aRoiId	The RoI ID
aVector	A vector of scan results
aInFile	The name of the localization file
aOutputPattern	The name of the output JSON file

Definition at line 26 of file API.cpp.

Referenced by RunScan().

5.18 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:


src/BayesianClustering/Cluster.cpp
BayesianClustering/Cluster.hpp
BayesianClustering/Data.hpp
boost/math/special_functions/erf.hpp
Utilities/GSLInterpolator.hpp
BayesianClustering/Configuration.hpp
BayesianClustering/Precision.hpp
math.h
cassert
iostream
gsl/gsl_interp.h
gsl/gsl_spline.h
gsl/gsl_errno.h
string
functional
map
stdexcept
vector

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.

5.18.1 Function Documentation

5.18.1.1 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
sigma	The standard-deviation of the Gaussian
x0	The mean of the Gaussian

Returns

the value of the Gaussian normal_cdf at x

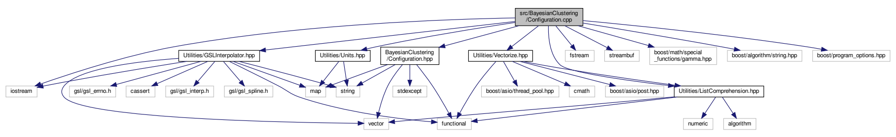
Definition at line 22 of file Cluster.cpp.

Referenced by Cluster::Parameter::alt_log_score(), and Cluster::Parameter::log_score().

5.19 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"
```

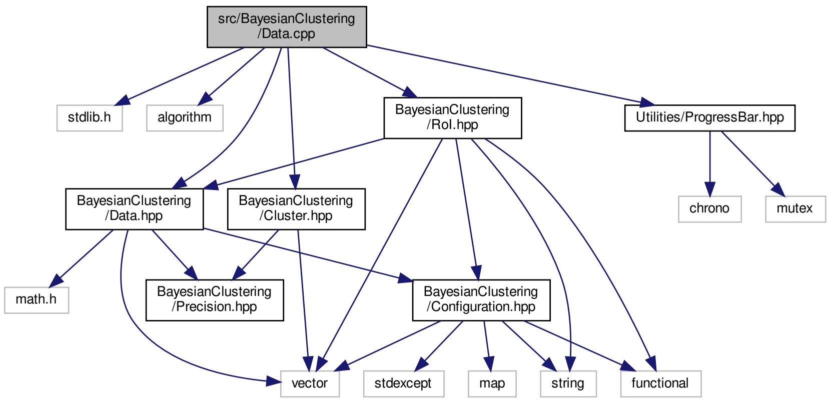
Include dependency graph for Configuration.cpp:



5.20 src/BayesianClustering/Data.cpp File Reference

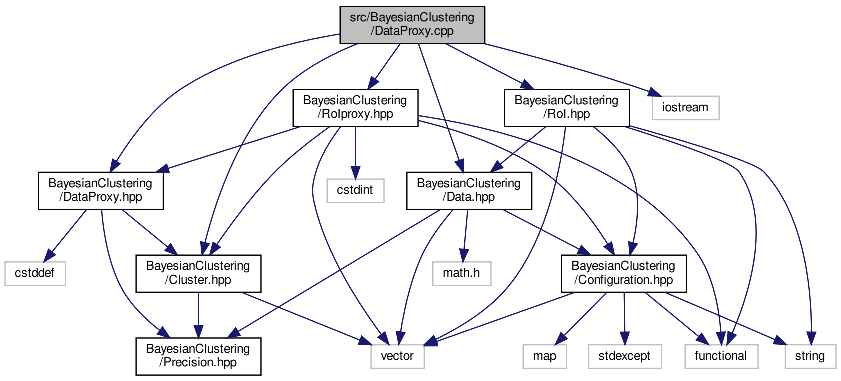
```
#include <stdlib.h>
#include <algorithm>
#include "BayesianClustering/Data.hpp"
#include "BayesianClustering/Cluster.hpp"
#include "BayesianClustering/RoI.hpp"
#include "Utilities/ProgressBar.hpp"
```

Include dependency graph for Data.cpp:



5.21 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:


src/BayesianClustering/DataProxy.cpp
BayesianClustering/RoIproxy.hpp
BayesianClustering/RoI.hpp
iostream
BayesianClustering/DataProxy.hpp
cstdint
BayesianClustering/Data.hpp
cstddef
BayesianClustering/Cluster.hpp
math.h
BayesianClustering/Configuration.hpp
BayesianClustering/Precision.hpp
vector
map
stdexcept
functional
string

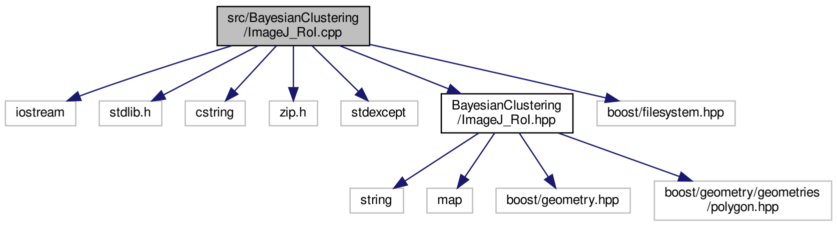
Macros

- #define RECURSION_LIMIT 75000

  The maximum depth for recursive clustering.

5.22 src/BayesianClustering/ImageJ_RoI.cpp File Reference

```
#include <iostream>
#include <stdlib.h>
#include <cstring>
#include <zip.h>
#include <stdexcept>
#include "BayesianClustering/ImageJ_RoI.hpp"
#include <boost/filesystem.hpp>
```
Include dependency graph for ImageJ_RoI.cpp:


src/BayesianClustering/ImageJ_RoI.cpp
iostream
stdlib.h
cstring
zip.h
stdexcept
BayesianClustering/ImageJ_RoI.hpp
boost/filesystem.hpp
string
map
boost/geometry.hpp
boost/geometry/geometries/polygon.hpp

Functions

- roi_polygon DecodeBinaryRoI (const uint8_t ∗const aData)

    Decode ImageJ binary RoI data to boost::geometry polygon Reverse engineered from `https://github.↩ com/imagej/ImageJ/blob/master/ij/io/RoiDecoder.java`.
- std::map< std::string, roi_polygon > OpenRoiZipfile (const std::string &aZipFileName)

    Decode ImageJ zipped binary RoI data to a map of named boost::geometry polygons.

5.22.1 Function Documentation

5.22.1.1 DecodeBinaryRoI()

```
roi_polygon DecodeBinaryRoI (
             const uint8_t *const aData )
```

Decode ImageJ binary RoI data to boost::geometry polygon Reverse engineered from `https://github.↩ com/imagej/ImageJ/blob/master/ij/io/RoiDecoder.java`.

Parameters

aData	A C-array containing the binary RoI data

Returns

A boost::geometry polygon containing the RoI information

Definition at line 17 of file ImageJ_RoI.cpp.

Referenced by OpenRoiZipfile().

5.22.1.2 OpenRoiZipfile()

```
std::map< std::string , roi_polygon > OpenRoiZipfile (
             const std::string & aZipFileName )
```

Decode ImageJ zipped binary RoI data to a map of named boost::geometry polygons.

Parameters

aZipFileName	The name of the zip-file to be opened

Returns

A map of named boost::geometry polygons containing the RoI information

Definition at line 31 of file ImageJ_RoI.cpp.

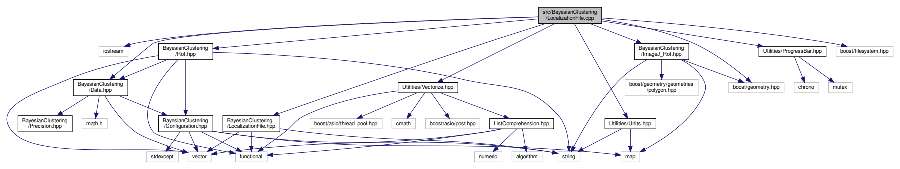
References DecodeBinaryRoI().

Referenced by LocalizationFile::ExtractRoIs(), and GetRoIs().

5.23 src/BayesianClustering/LocalizationFile.cpp File Reference

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

Include dependency graph for LocalizationFile.cpp:


src/BayesianClustering /LocalizationFile.cpp
iostream
BayesianClustering /RoI.hpp
BayesianClustering /ImageJ_RoI.hpp
Utilities/ProgressBar.hpp
boost/filesystem.hpp
BayesianClustering /Data.hpp
Utilities/Vectorize.hpp
boost/geometry/geometries /polygon.hpp
boost/geometry.hpp
chrono
mutex
BayesianClustering /Precision.hpp
math.h
BayesianClustering /Configuration.hpp
BayesianClustering /LocalizationFile.hpp
boost/asio/thread_pool.hpp
cmath
boost/asio/post.hpp
ListComprehension.hpp
Utilities/Units.hpp
stdexcept
vector
functional
numeric
algorithm
string
map

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 &aRoIid, const int &i, const int &j)

    Recursively search histogram for continuously connected regions over threshold.

5.23.1 Function Documentation

5.23.1.1 __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

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

Definition at line 30 of file LocalizationFile.cpp.

References nanometer.

Referenced by LocalizationFile::LocalizationFile().

5.23.1.2 __RecursiveSearch__()

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

Recursively search histogram for continuously connected regions over threshold.

Parameters

aHist	The histogram being searched
aRoIid	The id of the region being allocated
i	The horizontal index of the current cell in the histogram
j	The vertical index of the current cell in the histogram

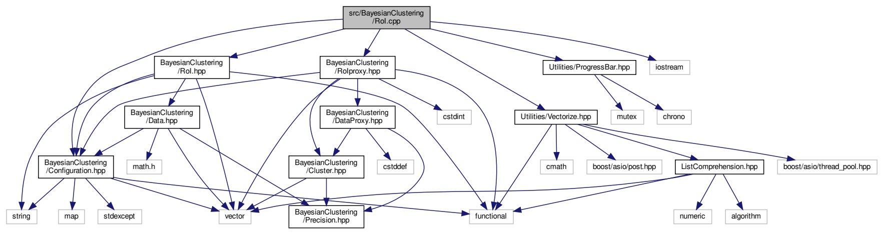
Definition at line 132 of file LocalizationFile.cpp.

Referenced by LocalizationFile::ExtractRoIs().

5.24 src/BayesianClustering/RoI.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>
```

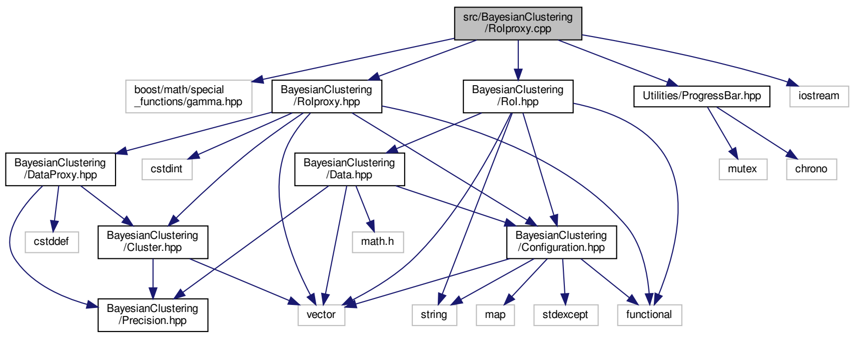
Include dependency graph for RoI.cpp:



5.25 src/BayesianClustering/RoIproxy.cpp File Reference

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

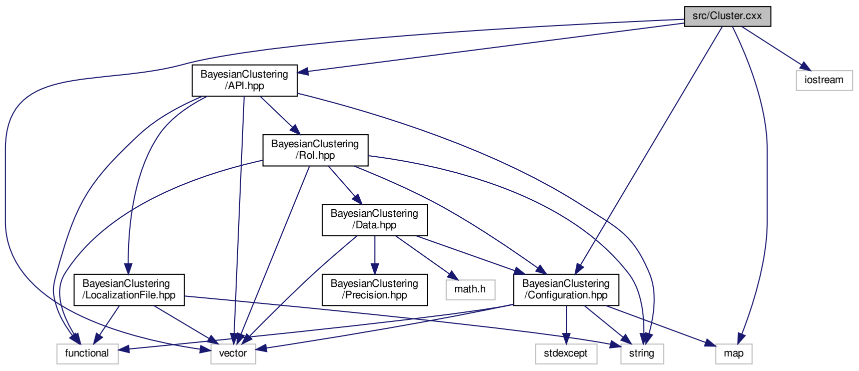
Include dependency graph for RoIproxy.cpp:



5.26 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:


src/Cluster.cxx
BayesianClustering /API.hpp
iostream
BayesianClustering /RoI.hpp
BayesianClustering /Data.hpp
BayesianClustering /LocalizationFile.hpp
BayesianClustering /Precision.hpp
math.h
BayesianClustering /Configuration.hpp
functional
vector
stdexcept
string
map

Functions

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

    Callback to report clusters.
- int main (int argc, char ∗∗argv)

    The main function.

5.26.1 Function Documentation

5.26.1.1 main()

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

The main function.

Parameters

argc	The number of commandline arguments
argv	The commandline arguments

Returns

The exit code

Definition at line 28 of file Cluster.cxx.

References AuxConfiguration::ClusterR(), AuxConfiguration::ClusterT(), AuxConfiguration::inputFile(), Aux←Configuration::outputFile(), and RunClustering().

5.26.1.2 ReportClusters()

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

Callback to report clusters.

Parameters

aClusters	A vector of clusters

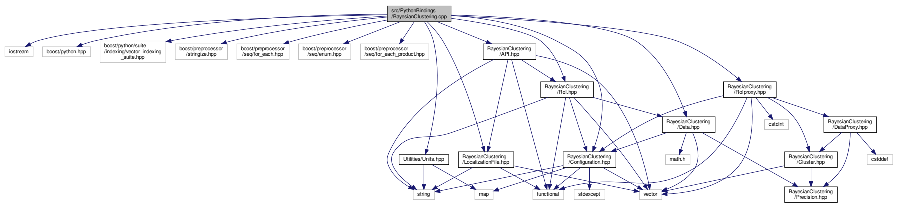
Definition at line 14 of file Cluster.cxx.

5.27 src/PythonBindings/BayesianClustering.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 <boost/preprocessor/seq/enum.hpp>
#include <boost/preprocessor/seq/for_each_product.hpp>
#include "Utilities/Units.hpp"
#include "BayesianClustering/API.hpp"
#include "BayesianClustering/LocalizationFile.hpp"
#include "BayesianClustering/Configuration.hpp"
#include "BayesianClustering/RoI.hpp"
#include "BayesianClustering/RoIproxy.hpp"
#include "BayesianClustering/Data.hpp"
```

Include dependency graph for BayesianClustering.cpp:



Macros

- #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_NO_CONSTRUCTOR(CLASS, DOC, ARGS) class_< CLASS >(#CLASS , DOC) BOOST_PP_SEQ_FOR_EACH(STRUCT_ARG , CLASS , ARGS)

    Helper Macro to deal with the boilerplate when dealing with structs.
- #define EXPOSE_STRUCT(CLASS, DOC, CONSTRUCTOR_ARGS, ARGS) class_< CLASS >(#CLASS , DOC , init< BOOST_PP_SEQ_ENUM(CONSTRUCTOR_ARGS) >()) 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 > >(BOOST_PP_STRINGIZE(Vector##CLASS) , BOOST_PP_STRINGIZE(An STL vector of CLASS)).def(vector_indexing_suite< std↩ ::vector< CLASS > >());

  Helper Macro to deal with the boilerplate when dealing with vectors of objects.
- #define ROICONFIGS (ImageJRoI)(AutoRoI)(ManualRoI)

  The ROI configs available.
- #define SCANCALLBACKS (std::string)(tSimpleScanCallback)(tFullScanCallback)

  The scan callbacks available.
- #define CLUSTERCALLBACKS (std::string)(tSimpleClusterCallback)(tFullClusterCallback)

  The clustering callbacks available.
- #define RUNSCAN(r, product) def("RunScan" , static_cast<void (*)(const std::string& , const BOOST_↩ PP_SEQ_ELEM(0,product)& , const ScanConfiguration& , const BOOST_PP_SEQ_ELEM(1,product)& a↩ Callback)>(&RunScan) , args("aInFile" , "aRoIConfig" , "aScanConfig" , "aHandler"));

  Macro to produce all permutations of RunScan.
- #define RUNCLUSTER(r, product) def("RunClustering" , static_cast<void (*)(const std::string& , const BOOST_PP_SEQ_ELEM(0,product)& , const double& , const double& , const BOOST_PP_SEQ_↩ ELEM(1,product)& aCallback)>(&RunClustering) , args("aInFile" , "aRoIConfig" , "aR" , "aT" , "aHandler"));

  Macro to produce all permutations of RunScan.

Functions

- std::shared_ptr< ScanConfiguration > ScanConfigurationConstructor (const std::size_t &aSigmaBins, const double &aSigmaMin, const double &aSigmaMax, const object &aInterpolator, const std::size_t &aRbins, const double &aMinScanR, const double &aMaxScanR, const std::size_t &aTbins, const double &aMin↩ ScanT, const double &aMaxScanT, const double &aPB, const double &aAlpha)

  Factory function to construct a ScanConfiguration which take the parameters directly in python.
- BOOST_PYTHON_MODULE (BayesianClustering)

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

5.27.1 Detailed Description

Self-contained sourcefile for producing python-bindings.

5.27.2 Macro Definition Documentation

5.27.2.1 EXPOSE_STRUCT

```
#define EXPOSE_STRUCT(
            CLASS,
            DOC,
            CONSTRUCTOR_ARGS,
            ARGS ) class_< CLASS >( #CLASS , DOC , init< BOOST_PP_SEQ_ENUM( CONSTRUCTOR_↩
ARGS ) >() ) BOOST_PP_SEQ_FOR_EACH( STRUCT_ARG , CLASS , ARGS )
```

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

Parameters

CLASS	The Class name
DOC	A string to be used as python documentation
CONSTRUCTOR_ARGS	Constructor argument types
ARGS	Sequence of arguments

Definition at line 91 of file BayesianClustering.cpp.

5.27.2.2 EXPOSE_STRUCT_NO_CONSTRUCTOR

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

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

Parameters

CLASS	The Class name
DOC	A string to be used as python documentation
ARGS	Sequence of arguments

Definition at line 84 of file BayesianClustering.cpp.

5.27.2.3 EXPOSE_VECTOR

```
#define EXPOSE_VECTOR(
              CLASS ) class_< std::vector< CLASS > >( BOOST_PP_STRINGIZE( Vector##CLASS ) ,
BOOST_PP_STRINGIZE( An STL vector of CLASS ) ).def( vector_indexing_suite< std::vector< CLASS
> >() );
```

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

Parameters

CLASS	The Class name

Definition at line 95 of file BayesianClustering.cpp.

5.27.2.4 RUNCLUSTER

```
#define RUNCLUSTER(
              r,
              product ) def( "RunClustering" , static_cast<void (*)( const std::string& ,
const BOOST_PP_SEQ_ELEM(0,product)& , const double& , const double& , const BOOST_PP_SEQ_↩
ELEM(1,product)& aCallback )>(&RunClustering) , args( "aInFile" , "aRoIConfig" , "aR" , "aT"
, "aHandler" ) );
```

Macro to produce all permutations of RunScan.

Parameters

r	UNUSED
product	Sequence of each possible products

Definition at line 114 of file BayesianClustering.cpp.

5.27.2.5 RUNSCAN

```
#define RUNSCAN(
              r,
              product ) def( "RunScan" , static_cast<void (*)( const std::string& , const
BOOST_PP_SEQ_ELEM(0,product)& , const ScanConfiguration& , const BOOST_PP_SEQ_ELEM(1,product)&
aCallback )>(&RunScan) , args( "aInFile" , "aRoIConfig" , "aScanConfig" , "aHandler" ) );
```

Macro to produce all permutations of RunScan.

Parameters

r	UNUSED
product	Sequence of each possible products

Definition at line 109 of file BayesianClustering.cpp.

5.27.2.6 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

r	BOOST PP internal
CLASS	The Class name
ARG	One of the arguments

Definition at line 78 of file BayesianClustering.cpp.

5.27.3 Function Documentation

5.27.3.1 ScanConfigurationConstructor()

```
std::shared_ptr< ScanConfiguration > ScanConfigurationConstructor (
            const std::size_t & aSigmaBins,
            const double & aSigmaMin,
            const double & aSigmaMax,
            const object & 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 )
```

Factory function to construct a ScanConfiguration which take the parameters directly in python.

Parameters

aSigmaBins	The number of sigma bins
aSigmaMin	The lowest sigma bin
aSigmaMax	The highest sigma bin
aInterpolator	A python function call or python dictionary containing a set of points from which to create an interpolator
aRbins	The number of R bins to scan over
aMinScanR	The lowest value of R to scan
aMaxScanR	The largest value of R to scan
aTbins	The number of T bins to scan over
aMinScanT	The lowest value of T to scan
aMaxScanT	The largest value of T to scan
aPB	The P_b parameter
aAlpha	The alpha parameter

Returns

a shared pointer to the new ScanConfiguration

Definition at line 49 of file BayesianClustering.cpp.

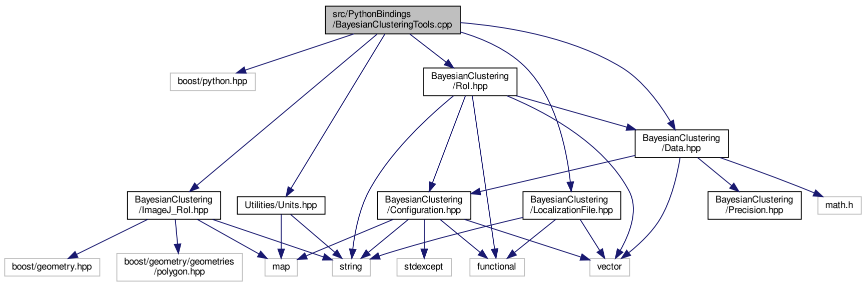
Referenced by BOOST_PYTHON_MODULE().

5.28 src/PythonBindings/BayesianClusteringTools.cpp File Reference

Self-contained sourcefile for producing python-bindings.

```
#include <boost/python.hpp>
#include "Utilities/Units.hpp"
#include "BayesianClustering/LocalizationFile.hpp"
#include "BayesianClustering/Data.hpp"
#include "BayesianClustering/RoI.hpp"
#include "BayesianClustering/ImageJ_RoI.hpp"
```

Include dependency graph for BayesianClusteringTools.cpp:



Functions

- boost::python::tuple GetLocalizations (const std::string &aFile)

  Debugging tool to get the raw x-y coordinates.
- boost::python::list GetRoIs (const std::string &aFile)

  Debugging tool to get the raw coordinates from an ImageJ RoI file
- boost::python::tuple CheckRoIs (const std::string &aFile, const std::string &aRoIFile, const double &aScale)

  Debugging tool to get the raw x-y coordinates and which RoI they are included in.
- BOOST_PYTHON_MODULE (BayesianClusteringTools)

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

5.28.1 Detailed Description

Self-contained sourcefile for producing python-bindings.

5.28.2 Function Documentation

5.28.2.1 CheckRoIs()

```
boost::python::tuple CheckRoIs (
            const std::string & aFile,
            const std::string & aRoIFile,
            const double & aScale )
```

Debugging tool to get the raw x-y coordinates and which RoI they are included in.

Parameters

aFile	The name of the localizations file
aRoIFile	The name of an ImageJ RoI file file
aScale	The size of the LSB in the ImageJ file

Returns

A python tuple of the raw localizations and a list of tuples containing the x-coordinates and the y-coordinates of the localizations in each RoI (both optimised for displaying in MatPlotLib)

Definition at line 62 of file BayesianClusteringTools.cpp.

References LocalizationFile::data(), and LocalizationFile::ExtractRoIs().

Referenced by BOOST_PYTHON_MODULE().

5.28.2.2 GetLocalizations()

```
boost::python::tuple GetLocalizations (
             const std::string & aFile )
```

Debugging tool to get the raw x-y coordinates.

Parameters

aFile	The name of the localizations file

Returns

The x-coordinates and the y-coordinates of the raw points as a python tuple (optimised for displaying in Mat←PlotLib)

Definition at line 24 of file BayesianClusteringTools.cpp.

References LocalizationFile::data().

Referenced by BOOST_PYTHON_MODULE().

5.28.2.3 GetRoIs()

```
boost::python::list GetRoIs (
             const std::string & aFile )
```

Debugging tool to get the raw coordinates from an ImageJ RoI file

Parameters

aFile	The name of an ImageJ RoI file

Returns

A list of python tuples, each containing the x-coordinates and the y-coordinates of the polygon points (optimised for displaying in MatPlotLib)

Definition at line 39 of file BayesianClusteringTools.cpp.

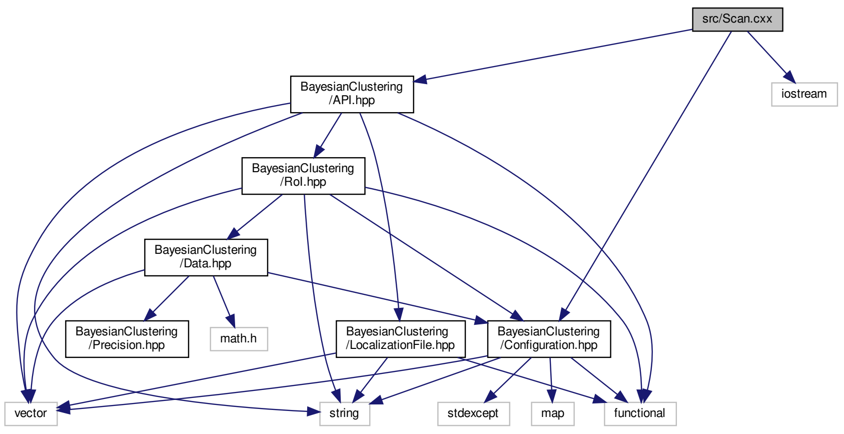
References OpenRoiZipfile().

Referenced by BOOST_PYTHON_MODULE().

5.29 src/Scan.cxx File Reference

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

Include dependency graph for Scan.cxx:


src/Scan.cxx
BayesianClustering /API.hpp
iostream
BayesianClustering /RoI.hpp
BayesianClustering /Data.hpp
BayesianClustering /Precision.hpp
math.h
BayesianClustering /LocalizationFile.hpp
BayesianClustering /Configuration.hpp
vector
string
stdexcept
map
functional

Functions

- int main (int argc, char ∗∗argv)

    The main function.

5.29.1 Function Documentation

5.29.1.1 main()

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

The main function.

Parameters

argc	The number of commandline arguments
argv	The commandline arguments

Returns

The exit code

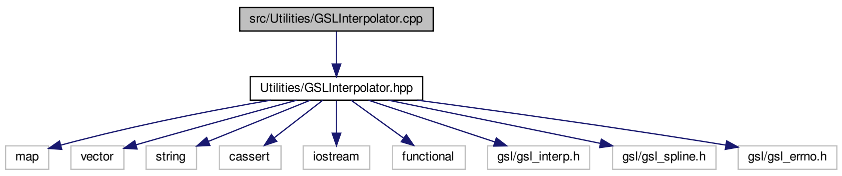
Definition at line 15 of file Scan.cxx.

References AuxConfiguration::configFile(), AuxConfiguration::inputFile(), AuxConfiguration::outputFile(), and Run↩ Scan().

5.30 src/Utilities/GSLInterpolator.cpp File Reference

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

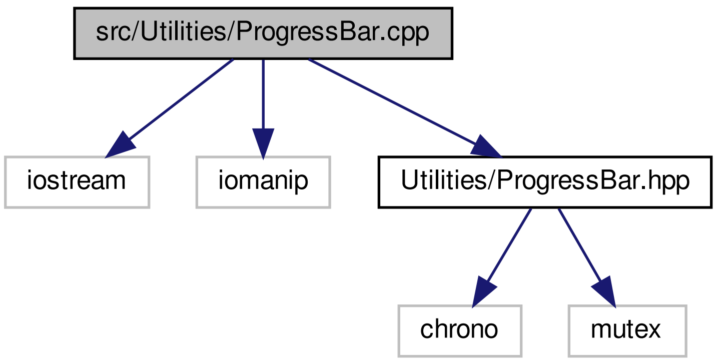
Include dependency graph for GSLInterpolator.cpp:



5.31 src/Utilities/ProgressBar.cpp File Reference

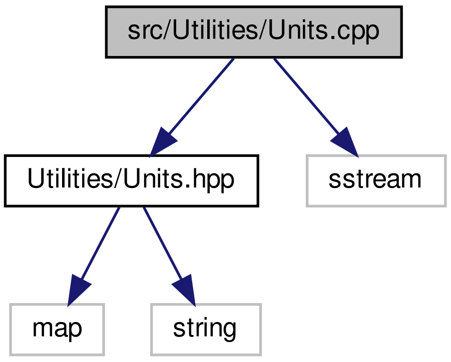
```
#include <iostream>
#include <iomanip>
#include "Utilities/ProgressBar.hpp"
```

Include dependency graph for ProgressBar.cpp:



5.32 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.

5.32.1 Function Documentation

5.32.1.1 StrToDist()

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

Convert a string representation to a distance.

Parameters

aStr	A string representation of a distance

Returns

The literal value

Definition at line 12 of file Units.cpp.

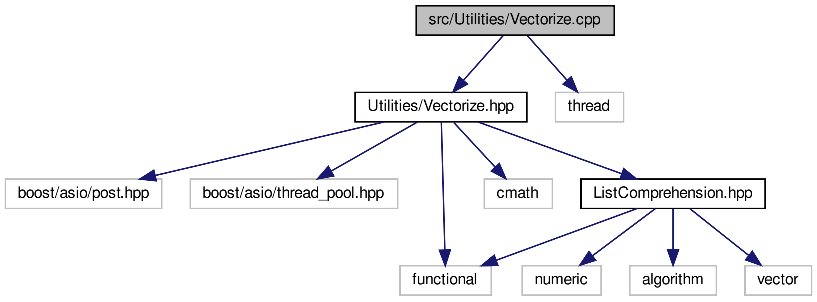
References UnitMap.

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

5.33 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.

5.33.1 Variable Documentation

5.33.1.1 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 RoI::ScanRT().

Index