Bayesian Cluster Tool

Generated by Doxygen 1.9.1

Mon Jun 5 2023 15:07:26

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

Class Documentation

3.1 AuxConfiguration Class Reference

Class for storing the auxilliary configuration parameters.

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

Public Member Functions

- AuxConfiguration ()

  Default constructor.
- 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.
- const bool & validate () const

  Getter for whether or not to run the validation on the clustering.
- const std::string & inputFile () const

  Getter for the input file.
- const std::string & outputFile () const

  Getter for the output file.
- const double & ClusterR () const

  Getter for the R value for a clusterization pass.
- const double & ClusterT () const

  Getter for the T value for a clusterization pass.
- void FromCommandline (int argc, char **argv)

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

  Parse the parameters when passed in as commandline arguments.

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.
- double mClusterR

  The value of R for clustering.
- double mClusterT

  The value of T for clustering.

3.1.1 Detailed Description

Class for storing the auxilliary configuration parameters.

Definition at line 169 of file Configuration.hpp.

3.1.2 Member Function Documentation

3.1.2.1 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 202 of file Configuration.hpp.

References mClusterR.

3.1.2.2 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 205 of file Configuration.hpp.

References mClusterT.

3.1.2.3 FromCommandline()

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

Parse the parameters when passed in as commandline arguments.

Parameters

argc	The number of commandline arguments
argv	The commandline arguments

Definition at line 200 of file Configuration.cpp.

References FromVector().

3.1.2.4 FromVector()

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

Parse the parameters when passed in as commandline arguments.

Parameters

aArgs	The commandline arguments

Definition at line 206 of file Configuration.cpp.

References mClusterR, mClusterT, SetInputFile(), SetOutputFile(), and SetValidate().

Referenced by FromCommandline().

3.1.2.5 inputFile()

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

Getter for the input file.

Returns

The name of the input RoI file

Definition at line 194 of file Configuration.hpp.

References mInputFile.

3.1.2.6 outputFile()

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

Getter for the output file.

Returns

The name of the output file

Definition at line 197 of file Configuration.hpp.

References mOutputFile.

3.1.2.7 SetInputFile()

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

Setter for the input file.

Parameters

aFileName	The name of the file

Definition at line 105 of file Configuration.cpp.

References mInputFile.

Referenced by FromVector().

3.1.2.8 SetOutputFile()

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

Setter for the output file.

Parameters

aFileName	The name of the file

Definition at line 112 of file Configuration.cpp.

References mOutputFile.

Referenced by FromVector().

3.1.2.9 SetValidate()

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

Set whether to validate clusterization.

Parameters

aValidate	Whether to validate clusterization

Definition at line 98 of file Configuration.cpp.

References mValidate.

Referenced by FromVector().

3.1.2.10 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 189 of file Configuration.hpp.

References mValidate.

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

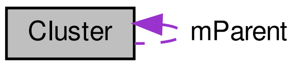
- include/BayesianClustering/Configuration.hpp
- src/BayesianClustering/Configuration.cpp

3.2 Cluster Class Reference

A class representing a cluster.

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

Collaboration diagram for Cluster:



Classes

- struct Parameter

  A struct representing the cluster parameters.

Public Member Functions

- Cluster (const std::size_t &aParamSize)

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

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

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

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

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

  Default move-assignment constructor.
- Cluster & operator+= (const Cluster &aOther)

  Add another cluster to this one.
- Cluster ∗ GetParent ()

  Get a pointer to this cluster's ultimate parent.
- void UpdateLogScore (const ScanConfiguration &aScanConfig)

  Update log-probability after a scan.

Public Attributes

- std::vector< Parameter > mParams

    Get the points after clustering.
- std::size_t mClusterSize

    The number of points in the current cluster.
- std::size_t mLastClusterSize

    The number of points in the cluster on the previous scan iteration.
- PRECISION mClusterScore

    The log-probability of the current cluster.
- Cluster ∗ mParent

    A pointer to the immediate parent of the current cluster.
- std::vector< Data ∗ > mData

    List of points in the cluster after clustering.

3.2.1 Detailed Description

A class representing a cluster.

Definition at line 15 of file Cluster.hpp.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 Cluster() [1/4]

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

Default constructor.

Parameters

aParamSize	The number of sigma-bins

Definition at line 96 of file Cluster.cpp.

3.2.2.2 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 102 of file Cluster.cpp.

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

3.2.2.3 Cluster() [3/4]

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

Deleted copy constructor.

Parameters

aOther	Anonymous argument

3.2.2.4 Cluster() [4/4]

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

Default move constructor.

Parameters

aOther	Anonymous argument

3.2.3 Member Function Documentation

3.2.3.1 GetParent()

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

Get a pointer to this cluster's ultimate parent.

Returns

A pointer to this cluster's ultimate parent

Definition at line 163 of file Cluster.cpp.

References GetParent(), and mParent.

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

3.2.3.2 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 153 of file Cluster.cpp.

References mClusterSize, and mParams.

3.2.3.3 operator=() [1/2]

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

Default move-assignment constructor.

Returns

Reference to this, for chaining calls

Parameters

aOther	Anonymous argument

3.2.3.4 operator=() [2/2]

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

Deleted assignment operator.

Returns

Reference to this, for chaining calls

Parameters

aOther	Anonymous argument

3.2.3.5 UpdateLogScore()

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

Update log-probability after a scan.

Parameters

aScanConfig	The configuration parameters for the scan

Definition at line 122 of file Cluster.cpp.

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

3.2.4 Member Data Documentation

3.2.4.1 mParams

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

Get the points after clustering.

Returns

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

Definition at line 102 of file Cluster.hpp.

Referenced by Cluster(), DataProxy::Clusterize(), operator+=(), UpdateLogScore(), and RoIproxy::ValidateLog←↩ Score().

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

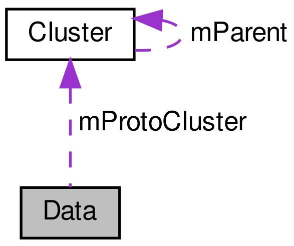
- include/BayesianClustering/Cluster.hpp
- src/BayesianClustering/Cluster.cpp

3.3 Data Class Reference

A class to store the raw data-points.

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

Collaboration diagram for Data:



Public Member Functions

- Data (const PRECISION &aX, const PRECISION &aY, const PRECISION &aS)

  Constructor.
- Data (const Data &aOther)=delete

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

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

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

  Default move-assignment constructor.
- virtual ∼Data ()

Destructor.
- bool operator< (const Data &aOther) const

  Comparison operator for sorting data-points by distance from the origin.
- PRECISION dR2 (const Data &aOther) const

  Return the squared-distance of this data-points from another.
- PRECISION dR (const Data &aOther) const

  Return the distance of this data-points from another.
- PRECISION dPhi (const Data &aOther) const

  Return the angle between this data-points and another.
- void Preprocess (std::vector< Data > &aData, const std::size_t &aIndex, const double &aMax2R, const double &aMax2R2, const std::vector< double > &aSigmabins2)

  All the necessary pre-processing to get this data-point ready for an RT-scan.
- void PreprocessLocalizationScores (std::vector< Data > &aData, const ScanConfiguration &aScanConfig, const double &aArea)

  Calculate the localization score from the local neighbourhood.
- PRECISION CalculateLocalizationScore (const std::vector< Data > &aData, const double &R, const double &aArea) const

  Calculate the localization score from the local neighbourhood.

Public Attributes

- PRECISION x

  The x-position of the data-point.
- PRECISION y

  The y-position of the data-point.
- PRECISION s

  The sigma of the data-point
- PRECISION r2

  The squared radial distance of the data-point.
- PRECISION r

  The radial distance of the data-point.
- PRECISION phi

  The phi-position of the data-point.
- std::vector< PRECISION > mLocalizationScores

  The 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.

3.3.1 Detailed Description

A class to store the raw data-points.

Definition at line 15 of file Data.hpp.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 Data() [1/3]

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

Constructor.

Parameters

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

Definition at line 12 of file Data.cpp.

3.3.2.2 Data() [2/3]

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

Deleted copy constructor.

Parameters

aOther	Anonymous argument

3.3.2.3 Data() [3/3]

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

Default move constructor.

Parameters

aOther	Anonymous argument

3.3.3 Member Function Documentation

3.3.3.1 CalculateLocalizationScore()

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

Calculate the localization score from the local neighbourhood.

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

Parameters

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

Returns

The localization score

Definition at line 107 of file Data.cpp.

References mNeighbours.

3.3.3.2 dPhi()

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

Return the angle between this data-points and another.

Returns

The angle between this data-points and another

Parameters

aOther	A data-point to compare against

Definition at line 69 of file Data.hpp.

References phi.

3.3.3.3 dR()

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

Return the distance of this data-points from another.

Returns

The distance of this data-points from another

Parameters

aOther	A data-point to compare against

Definition at line 61 of file Data.hpp.

References dR2().

3.3.3.4 dR2()

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

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

Returns

The squared-distance of this data-points from another

Parameters

aOther	A data-point to compare against

Definition at line 52 of file Data.hpp.

References x, and y.

Referenced by dR().

3.3.3.5 operator<()

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

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

Returns

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

Parameters

aOther	A data-point to compare against

Definition at line 44 of file Data.hpp.

References r.

3.3.3.6 operator=() [1/2]

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

Deleted assignment operator.

Returns

Reference to this, for chaining calls

Parameters

aOther	Anonymous argument

3.3.3.7 operator=() [2/2]

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

Default move-assignment constructor.

Returns

Reference to this, for chaining calls

Parameters

aOther	Anonymous argument

3.3.3.8 Preprocess()

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

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

Parameters

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

Definition at line 26 of file Data.cpp.

3.3.3.9 PreprocessLocalizationScores()

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

Calculate the localization score from the local neighbourhood.

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

Parameters

aData	?
aScanConfig	The configuration parameters for the scan
aArea	The area of the window for normalizing the log score

Definition at line 75 of file Data.cpp.

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

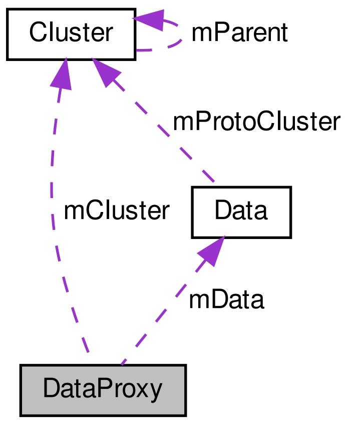
- include/BayesianClustering/Data.hpp
- src/BayesianClustering/Data.cpp

3.4 DataProxy Class Reference

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

`#include <DataProxy.hpp>`

Collaboration diagram for DataProxy:



Public Member Functions

- DataProxy (Data &aData)

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

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

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

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

  Default move-assignment constructor.
- void Clusterize (const PRECISION &a2R2, 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.

3.4.1 Detailed Description

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

Definition at line 17 of file DataProxy.hpp.

3.4.2 Constructor & Destructor Documentation

3.4.2.1 DataProxy() `[1/3]`

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

Default constructor.

Parameters

aData	The data-point for which this is the proxy

Definition at line 15 of file DataProxy.cpp.

3.4.2.2 DataProxy() `[2/3]`

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

Deleted copy constructor.

Parameters

aOther	Anonymous argument

3.4.2.3 DataProxy() `[3/3]`

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

Default move constructor.

Parameters

aOther	Anonymous argument

3.4.3 Member Function Documentation

3.4.3.1 Clusterize() [1/2]

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

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

Parameters

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

Definition at line 21 of file DataProxy.cpp.

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

Referenced by Clusterize().

3.4.3.2 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 35 of file DataProxy.cpp.

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

3.4.3.3 GetCluster()

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

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

Returns

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

Definition at line 52 of file DataProxy.hpp.

References Cluster::GetParent(), and mCluster.

Referenced by Clusterize().

3.4.3.4 operator=() [1/2]

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

Deleted assignment operator.

Returns

Reference to this, for chaining calls

Parameters

aOther	Anonymous argument

3.4.3.5 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

3.5 GSLInterpolator Class Reference

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

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

Public Member Functions

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

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

  Initialised splice constructor.
- virtual ∼GSLInterpolator ()

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

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

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

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

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

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

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

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

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

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

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

  The integral over the spline between two bounds.

Private Attributes

- unsigned int nErrors

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

  Underlying GSL machinery.
- gsl_spline ∗ fSpline

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

  Underlying GSL machinery for the interpolation type.

3.5.1 Detailed Description

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

Definition at line 18 of file GSLInterpolator.hpp.

3.5.2 Constructor & Destructor Documentation

3.5.2.1 GSLInterpolator() [1/4]

```
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 7 of file GSLInterpolator.cpp.

References fInterpType, and fSpline.

3.5.2.2 GSLInterpolator() [2/4]

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

Initialised splice constructor.

Parameters

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

Definition at line 17 of file GSLInterpolator.cpp.

References fInterpType, fSpline, and SetData().

3.5.2.3 GSLInterpolator() [3/4]

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

Deleted copy constructor.

Parameters

aOther	Anonymous argument

3.5.2.4 GSLInterpolator() [4/4]

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

Default move constructor.

Parameters

aOther	Anonymous argument

3.5.3 Member Function Documentation

3.5.3.1 Deriv()

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

The first derivative of the spline at the given x.

Parameters

x	The x-coordinate at which to evaluate the derivative

Returns

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

Definition at line 100 of file GSLInterpolator.hpp.

References Evaluate(), fAccel, and fSpline.

3.5.3.2 Deriv2()

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

The second derivative of the spline at the given x.

Parameters

x	The x-coordinate at which to evaluate the derivative

Returns

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

Definition at line 108 of file GSLInterpolator.hpp.

References Evaluate(), fAccel, and fSpline.

3.5.3.3 Eval()

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

Evaluate the spline at the given x.

Parameters

x	The x-coordinate at which to evaluate the spline

Returns

The value of the spline at the given x-coordinate

Definition at line 92 of file GSLInterpolator.hpp.

References Evaluate(), fAccel, and fSpline.

Referenced by ScanConfiguration::FromVector().

3.5.3.4 Evaluate()

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

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

Parameters

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

Returns

The interpolated value

Definition at line 73 of file GSLInterpolator.hpp.

References fAccel, and nErrors.

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

3.5.3.5 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 117 of file GSLInterpolator.hpp.

References Evaluate(), fAccel, and fSpline.

3.5.3.6 operator=() `[1/2]`

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

Deleted assignment operator.

Returns

Reference to this, for chaining calls

Parameters

aOther	Anonymous argument

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

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

Default move-assignment constructor.

Returns

Reference to this, for chaining calls

Parameters

aOther	Anonymous argument

3.5.3.8 SetData() `[1/2]`

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

Set the spline data points.

Parameters

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

Returns

success or fail

Definition at line 56 of file GSLInterpolator.hpp.

Referenced by GSLInterpolator().

3.5.3.9 SetData() [2/2]

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

Set the spline data points.

Parameters

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 36 of file GSLInterpolator.cpp.

References fAccel, fInterpType, fSpline, and nErrors.

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

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

3.6 LocalizationFile Class Reference

A class to store the raw data-points.

`#include <LocalizationFile.hpp>`

Public Member Functions

- LocalizationFile (const std::string &aFilename)

  Constructor.
- LocalizationFile (const LocalizationFile &aOther)=delete

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

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

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

  Default move-assignment constructor.
- ∼LocalizationFile ()=default

  Default destructor.
- void ExtractRoIs (const std::function< void(RoI &) > &aCallback)

  Automatically extract the RoIs.

Private Attributes

- std::vector< Data > mData

  The localizations in the file.

3.6.1 Detailed Description

A class to store the raw data-points.

Definition at line 14 of file LocalizationFile.hpp.

3.6.2 Constructor & Destructor Documentation

3.6.2.1 LocalizationFile() `[1/3]`

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

Constructor.

Parameters

aFilename	The name of the localizations file

Definition at line 64 of file LocalizationFile.cpp.

References mData.

3.6.2.2 LocalizationFile() [2/3]

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

Deleted copy constructor.

Parameters

aOther	Anonymous argument

3.6.2.3 LocalizationFile() [3/3]

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

Default move constructor.

Parameters

aOther	Anonymous argument

3.6.3 Member Function Documentation

3.6.3.1 ExtractRoIs()

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

Automatically extract the RoIs.

Parameters

aCallback	A handler for each RoI found

Definition at line 106 of file LocalizationFile.cpp.

References mData, RoI::SetCentre(), and RoI::SetWidth().

3.6.3.2 operator=() [1/2]

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

Deleted assignment operator.

Returns

Reference to this, for chaining calls

Parameters

aOther	Anonymous argument

3.6.3.3 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

3.7 Cluster::Parameter Struct Reference

A struct representing the cluster parameters.

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

Public Member Functions

- Parameter ()

  Default constructor.
- Parameter & operator+= (const Parameter &aOther)

  Add another set of parameters to this set.
- double log_score () const

  Convert the parameters to a log-probability.
- 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.

3.7.1 Detailed Description

A struct representing the cluster parameters.

Definition at line 20 of file Cluster.hpp.

3.7.2 Member Function Documentation

3.7.2.1 alt_log_score()

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

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

Returns

the log-probability of this set of cluster parameters

Definition at line 50 of file Cluster.cpp.

3.7.2.2 log_score()

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

Convert the parameters to a log-probability.

Returns

the log-probability of this set of cluster parameters

Definition at line 75 of file Cluster.cpp.

3.7.2.3 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 32 of file Cluster.cpp.

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

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

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

3.8 ProgressBar Struct Reference

A utility progress-bar.

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

Public Member Functions

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

  Constructor.
- virtual ∼ProgressBar ()

  Destructor.
- void operator++ ()

  Postfix increment.
- void operator++ (int aDummy)

  Prefix increment.

Public Attributes

- float mBlockSize

  The size of each increment.
- float mNextThreshold

  The next threshold at which we will write a block to stdout.
- std::size_t mCount

  The number of times we have incremented.
- std::chrono::high_resolution_clock::time_point mStart

  A timer for end-of-task stats.

3.8.1 Detailed Description

A utility progress-bar.

Definition at line 6 of file ProgressBar.hpp.

3.8.2 Constructor & Destructor Documentation

3.8.2.1 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 7 of file ProgressBar.cpp.

3.8.3 Member Function Documentation

3.8.3.1 operator++()

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

Prefix increment.

Parameters

aDummy	Anonymous argument

Definition at line 27 of file ProgressBar.cpp.

References operator++().

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

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

3.9 ProgressBar2 Struct Reference

A utility code timer.

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

Public Member Functions

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

Public Attributes

- std::chrono::high_resolution_clock::time_point mStart
  A timer for end-of-task stats.

3.9.1 Detailed Description

A utility code timer.

Definition at line 34 of file ProgressBar.hpp.

3.9.2 Constructor & Destructor Documentation

3.9.2.1 ProgressBar2()

```
ProgressBar2::ProgressBar2 (
             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 32 of file ProgressBar.cpp.

3.9.3 Member Function Documentation

3.9.3.1 operator++()

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

Prefix increment.

Parameters

aDummy	Anonymous argument

Definition at line 44 of file ProgressBar.cpp.

References operator++().

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

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

3.10 RoI Class Reference

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

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

Classes

- struct ScanEntry

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

Public Member Functions

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

  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 & 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(const RoIproxy &, const double &, const double &, std::pair< int, int >) > &aCallback)

  Run the scan.
- void ScanRT (const ScanConfiguration &aScanConfig, const std::function< void(const std::vector< ScanEntry > &) > &aCallback)

  Run the scan.
- void Clusterize (const double &R, const double &T, const std::function< void(const RoIproxy &) > &aCallback)

  Run clusterization for a specific choice of R and T.
- void SetCentre (const double &aPhysicalCentreX, const double &aPhysicalCentreY)

  Setter for the centre of the scan window.
- void SetWidth (const double &aWidthX, const double &aWidthY)

  Setter for the size of the RoI window.
- double getCentreX () const

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

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

  Getter for the width of the ROI window.
- double getWidthY () const

  Getter for the height of the ROI window.
- double getArea () const

  Getter for the height of the ROI window.

Public Attributes

- std::vector< Data > mData

  The collection of raw data points.

Private Attributes

- double mPhysicalCentreX

  The x-coordinate of the centre of the window in physical units.
- double mPhysicalCentreY

  The y-coordinate of the centre of the window in physical units.
- double mWidthX

  The width of the window in the x-direction in physical units.
- double mWidthY

  The width of the window in the y-direction in physical units.
- double mArea

  The area of the window in physical units.

3.10.1 Detailed Description

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

Definition at line 17 of file RoI.hpp.

3.10.2 Constructor & Destructor Documentation

3.10.2.1 RoI() [1/3]

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

Default Constructor.

Parameters

aData	The set of data-points in the RoI

Definition at line 16 of file RoI.cpp.

References mData.

3.10.2.2 RoI() [2/3]

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

Deleted copy constructor.

Parameters

aOther	Anonymous argument

3.10.2.3 RoI() [3/3]

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

Default move constructor.

Parameters

aOther	Anonymous argument

3.10.3 Member Function Documentation

3.10.3.1 Clusterize()

```
void RoI::Clusterize (
            const double & R,
            const double & T,
            const std::function< void(const 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 62 of file RoI.cpp.

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

3.10.3.2 getArea()

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

Getter for the height of the ROI window.

Returns

The height of the ROI window

Definition at line 104 of file RoI.hpp.

References mArea.

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

3.10.3.3 getCentreX()

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

Getter for the x-coordinate of the physical centre.

Returns

The x-coordinate of the physical centre

Definition at line 89 of file RoI.hpp.

References mPhysicalCentreX.

3.10.3.4 getCentreY()

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

Getter for the y-coordinate of the physical centre.

Returns

The y-coordinate of the physical centre

Definition at line 92 of file RoI.hpp.

References mPhysicalCentreY.

3.10.3.5 getWidthX()

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

Getter for the width of the ROI window.

Returns

The width of the ROI window

Definition at line 96 of file RoI.hpp.

References mWidthX.

3.10.3.6 getWidthY()

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

Getter for the height of the ROI window.

Returns

The height of the ROI window

Definition at line 100 of file RoI.hpp.

References mWidthY.

3.10.3.7 operator=() [1/2]

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

Deleted assignment operator.

Returns

Reference to this, for chaining calls

Parameters

aOther	Anonymous argument

3.10.3.8 operator=() [2/2]

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

Default move-assignment constructor.

Returns

Reference to this, for chaining calls

Parameters

aOther	Anonymous argument

3.10.3.9 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 27 of file RoI.cpp.

References mData.

Referenced by Clusterize(), and ScanRT().

3.10.3.10 ScanRT() [1/2]

```
void RoI::ScanRT (
            const ScanConfiguration & aScanConfig,
            const std::function< void(const RoIproxy &, const double &, const double &, std←
::pair< int, int >) > & aCallback )
```

Run the scan.

Parameters

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

Definition at line 36 of file RoI.cpp.

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

Referenced by ScanRT().

3.10.3.11 ScanRT() [2/2]

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

Run the scan.

Parameters

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

Definition at line 53 of file RoI.cpp.

References RoIproxy::mLogP, and ScanRT().

3.10.3.12 SetCentre()

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

Setter for the centre of the scan window.

Parameters

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)

Definition at line 74 of file RoI.cpp.

References mPhysicalCentreX, and mPhysicalCentreY.

Referenced by LocalizationFile::ExtractRoIs().

3.10.3.13 SetWidth()

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

Setter for the size of the RoI window.

Parameters

aWidthX	The width of the window in physical units
aWidthY	The height of the window in physical units

Definition at line 81 of file RoI.cpp.

References mArea, mWidthX, and mWidthY.

Referenced by LocalizationFile::ExtractRoIs().

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

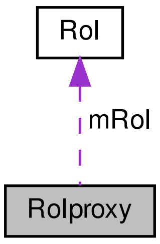
- include/BayesianClustering/RoI.hpp
- src/BayesianClustering/RoI.cpp

3.11 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.
- void CheckClusterization (const double &R, const double &T)

  Run validation tests on the clusters.
- void ScanRT (const ScanConfiguration &aScanConfig, const std::function< void(const RoIproxy &, const double &, const double &, std::pair< int, int >) > &aCallback, const uint8_t &aParallelization=1, const uint8←_t &aOffset=0, const bool &aValidate=false)

  Run an RT-scan.
- void Clusterize (const double &R, const double &T, const std::function< void(const 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.

3.11.1 Detailed Description

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

Definition at line 17 of file RoIproxy.hpp.

3.11.2 Constructor & Destructor Documentation

3.11.2.1 RoIproxy() [1/3]

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

Default constructor.

Parameters

aRoI	An RoI for which this is a lightweight proxy

Definition at line 16 of file RoIproxy.cpp.

References mClusters, RoI::mData, and mData.

3.11.2.2 RoIproxy() [2/3]

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

Deleted copy constructor.

Parameters

aOther	Anonymous argument

3.11.2.3 RoIproxy() [3/3]

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

Default move constructor.

Parameters

aOther	Anonymous argument

3.11.3 Member Function Documentation

3.11.3.1 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 24 of file RoIproxy.cpp.

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

3.11.3.2 Clusterize()

```
void RoIproxy::Clusterize (
            const double & R,
            const double & T,
            const std::function< void(const 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 138 of file RoIproxy.cpp.

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

Referenced by RoI::Clusterize().

3.11.3.3 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 68 of file RoIproxy.hpp.

References mData.

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

3.11.3.4 operator=() [1/2]

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

Deleted assignment operator.

Returns

Reference to this, for chaining calls

Parameters

aOther	Anonymous argument

3.11.3.5 operator=() [2/2]

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

Default move-assignment constructor.

Returns

Reference to this, for chaining calls

Parameters

aOther	Anonymous argument

3.11.3.6 ScanRT()

```
void RoIproxy::ScanRT (
             const ScanConfiguration & aScanConfig,
             const std::function< void(const RoIproxy &, const double &, const double &, std↩
::pair< int, int >) > & aCallback,
             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
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 96 of file RoIproxy.cpp.

3.11.3.7 UpdateLogScore()

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

Update log-probability after a scan.

Parameters

aScanConfig	The configuration parameters for the scan

Definition at line 223 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().

3.11.3.8 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:

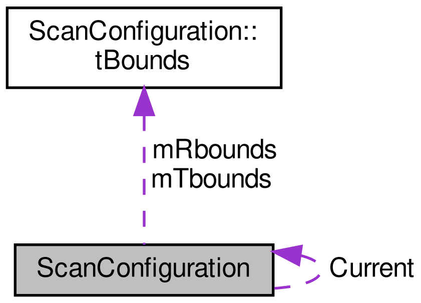
- include/BayesianClustering/RoIproxy.hpp
- src/BayesianClustering/RoIproxy.cpp

3.12 ScanConfiguration Class Reference

Class for storing the scan configuration parameters.

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

Collaboration diagram for ScanConfiguration:



Classes

- struct tBounds

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

Public Member Functions

- ScanConfiguration ()

  Default constructor.
- void SetSigmaParameters (const std::size_t &aSigmacount, 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=0.0, const double &aMaxScanR=-1)

  Setter for the R bins for the RT scan.
- void SetTBins (const std::size_t &aTbins, const double &aMinScanT=0.0, const double &aMaxScanT=-1)
- void SetPb (const double &aPB)

  Setter for the P_b parameter.
- void SetAlpha (const double &aAlpha)

  Setter for the alpha parameter.
- void FromCommandline (int argc, char **argv)

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

  Parse the parameters when passed in as commandline arguments.
- 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

Static Public Attributes

- static ScanConfiguration $*$ Current

  A single global copy of the global variables.

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

3.12.1 Detailed Description

Class for storing the scan configuration parameters.

Definition at line 10 of file Configuration.hpp.

3.12.2 Member Function Documentation

3.12.2.1 alpha()

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

Getter for the alpha parameter

Returns

The alpha parameter

Definition at line 122 of file Configuration.hpp.

References mAlpha.

Referenced by RoIproxy::UpdateLogScore().

3.12.2.2 FromCommandline()

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

Parse the parameters when passed in as commandline arguments.

Parameters

argc	The number of commandline arguments
argv	The commandline arguments

Definition at line 137 of file Configuration.cpp.

References FromVector().

3.12.2.3 FromVector()

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

Parse the parameters when passed in as commandline arguments.

Parameters

aArgs	The commandline arguments

Definition at line 143 of file Configuration.cpp.

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

Referenced by FromCommandline().

3.12.2.4 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 86 of file Configuration.hpp.

References mLogProbabilitySigma.

Referenced by Cluster::UpdateLogScore().

3.12.2.5 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 103 of file Configuration.hpp.

References mLogProbabilitySigma.

3.12.2.6 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 125 of file Configuration.hpp.

References mLogAlpha.

Referenced by RoIproxy::UpdateLogScore().

3.12.2.7 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 128 of file Configuration.hpp.

References mLogGammaAlpha.

Referenced by RoIproxy::UpdateLogScore().

3.12.2.8 logPb()

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

Logarithm of the P_b parameter

Returns

Logarithm of the P_b parameter

Definition at line 115 of file Configuration.hpp.

References mLogPb.

Referenced by RoIproxy::UpdateLogScore().

3.12.2.9 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 118 of file Configuration.hpp.

References mLogPbDagger.

Referenced by RoIproxy::UpdateLogScore().

3.12.2.10 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 83 of file Configuration.hpp.

References mProbabilitySigma.

3.12.2.11 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 99 of file Configuration.hpp.

References mProbabilitySigma.

3.12.2.12 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 107 of file Configuration.hpp.

References mRbounds.

Referenced by RoI::ScanRT().

3.12.2.13 SetAlpha()

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

Setter for the alpha parameter.

Parameters

aAlpha	The alpha parameter

Definition at line 81 of file Configuration.cpp.

References mAlpha, mLogAlpha, and mLogGammaAlpha.

Referenced by FromVector().

3.12.2.14 SetPb()

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

Setter for the P_b parameter.

Parameters

aPB	The P_b parameter

Definition at line 74 of file Configuration.cpp.

References mLogPb, and mLogPbDagger.

Referenced by FromVector().

3.12.2.15 SetRBins()

```
void ScanConfiguration::SetRBins (
            const std::size_t & aRbins,
            const double & aMinScanR = 0.0,
            const double & aMaxScanR = -1 )
```

Setter for the R bins for the RT scan.

Parameters

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 54 of file Configuration.cpp.

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

Referenced by FromVector().

3.12.2.16 SetSigmaParameters()

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

Setter for the sigma-bins to be integrated over.

Parameters

aSigmacount	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 32 of file Configuration.cpp.

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

Referenced by FromVector().

3.12.2.17 SetTBins()

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

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 64 of file Configuration.cpp.

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

Referenced by FromVector().

3.12.2.18 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 77 of file Configuration.hpp.

References mSigmabins.

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

3.12.2.19 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 91 of file Configuration.hpp.

References mSigmabins.

3.12.2.20 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 80 of file Configuration.hpp.

References mSigmabins2.

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

3.12.2.21 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 95 of file Configuration.hpp.

References mSigmabins2.

3.12.2.22 sigmacount()

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

Getter for the sigma count.

Returns

The sigma count

Definition at line 69 of file Configuration.hpp.

References mSigmacount.

Referenced by RoIproxy::ValidateLogScore().

3.12.2.23 sigmaspacing()

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

Getter for the sigma spacing.

Returns

The sigma spacing

Definition at line 73 of file Configuration.hpp.

References mSigmaspacing.

3.12.2.24 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 110 of file Configuration.hpp.

References mTbounds.

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

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

3.13 RoI::ScanEntry Struct Reference

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

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

Public Attributes

- double r

  The R parameter.
- double t

  The T parameter.
- PRECISION score

  The score.

3.13.1 Detailed Description

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

Definition at line 22 of file RoI.hpp.

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

- include/BayesianClustering/RoI.hpp

3.14 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.

3.14.1 Detailed Description

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

Definition at line 15 of file Configuration.hpp.

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

- include/BayesianClustering/Configuration.hpp

Index