

Sequential Quantile Estimation

Reference Manual of a Software Tool implemented in the Context of the PhD Thesis
Sequential Analysis of Quantiles and Probability Distributions by Replicated Simulations
submitted in July 2007 at the University of Canterbury

Mirko Eickhoff

Technical Report: TR-COSC 01/07

Department of Computer Science & Software Engineering
University of Canterbury
Christchurch, New Zealand

29 June, 2007

Contents

1	Sequential Quantile Estimation Directory Hierarchy	1
1.1	Sequential Quantile Estimation Directories	1
2	Sequential Quantile Estimation Namespace Index	3
2.1	Sequential Quantile Estimation Namespace List	3
3	Sequential Quantile Estimation Hierarchical Index	5
3.1	Sequential Quantile Estimation Class Hierarchy	5
4	Sequential Quantile Estimation Data Structure Index	7
4.1	Sequential Quantile Estimation Data Structures	7
5	Sequential Quantile Estimation File Index	9
5.1	Sequential Quantile Estimation File List	9
6	Sequential Quantile Estimation Directory Documentation	11
6.1	/home/cosc/student/mei16/archiv/projects/distributionEstimation/shared/ Directory Reference	11
6.2	source/ Directory Reference	13
7	Sequential Quantile Estimation Namespace Documentation	15
7.1	lib_signals Namespace Reference	15
7.2	logInfo Namespace Reference	18
8	Sequential Quantile Estimation Data Structure Documentation	19
8.1	akaroa_import Class Reference	19
8.2	AndersonDarlingKSampleTestEqualECDFSize Class Reference	23
8.3	batch_mean_QE Class Reference	29
8.4	batching Class Reference	37
8.5	confidenceInterval_SSC_QE Class Reference	46
8.6	controller Class Reference	50

8.7	deterministic_SSC_QE Class Reference	54
8.8	deterministic_TPD Class Reference	59
8.9	error_in_FCM Class Reference	64
8.10	evolution Class Reference	66
8.11	homogeneityTest Class Reference	72
8.12	interface_multipleRuns Class Reference	73
8.13	interface_singleRun Class Reference	76
8.14	K_d_entry Struct Reference	78
8.15	KolmogorovSmirnov2SampleTest Class Reference	80
8.16	method_factory Class Reference	84
8.17	outputAnalyser Class Reference	86
8.18	pooling_QE Class Reference	89
8.19	prng Class Reference	97
8.20	quantile_estimation Class Reference	103
8.21	quantile_rank Class Reference	108
8.22	relativeErrorQuantile_SSC_QE Class Reference	115
8.23	relativeErrorRange_SSC_QE Class Reference	120
8.24	resultInfo Class Reference	124
8.25	sequential_TPD Class Reference	125
8.26	SequentialStoppingCriteria_QE Class Reference	136
8.27	SequentialStoppingCriteria_QE::estimate Struct Reference	140
8.28	setting Class Reference	142
8.29	settingEntry Class Reference	146
8.30	spectral_analysis_QE Class Reference	151
8.31	statistic_collection Class Reference	159
8.32	system_command Class Reference	176
8.33	truncation_point_detection Class Reference	181
9	Sequential Quantile Estimation File Documentation	185
9.1	akaroa_import.cc File Reference	185
9.2	akaroa_import.h File Reference	187
9.3	basic.cc File Reference	188
9.4	basic.h File Reference	195
9.5	batching.cc File Reference	204
9.6	batching.h File Reference	205
9.7	controller.cc File Reference	206
9.8	controller.h File Reference	207

9.9	environment.h File Reference	208
9.10	error.cc File Reference	210
9.11	error.h File Reference	211
9.12	homogeneityTests.cc File Reference	212
9.13	homogeneityTests.h File Reference	213
9.14	interface.cc File Reference	214
9.15	interface.h File Reference	215
9.16	logfile.cc File Reference	217
9.17	logfile.h File Reference	218
9.18	main.cc File Reference	220
9.19	main.h File Reference	222
9.20	measure.cc File Reference	223
9.21	measure.h File Reference	224
9.22	method_factory.cc File Reference	229
9.23	method_factory.h File Reference	230
9.24	prng.cc File Reference	231
9.25	prng.h File Reference	233
9.26	quantile_estimation.cc File Reference	234
9.27	quantile_estimation.h File Reference	235
9.28	resultfile.cc File Reference	236
9.29	resultfile.h File Reference	237
9.30	setting.cc File Reference	238
9.31	setting.h File Reference	239
9.32	signal_interface.cc File Reference	241
9.33	signal_interface.h File Reference	242
9.34	statistic.cc File Reference	243
9.35	statistic.h File Reference	244
9.36	system_command.cc File Reference	246
9.37	system_command.h File Reference	247
9.38	time_evolution.cc File Reference	248
9.39	time_evolution.h File Reference	249
9.40	truncation_point_detection.cc File Reference	250
9.41	truncation_point_detection.h File Reference	251

Chapter 1

Sequential Quantile Estimation Directory Hierarchy

1.1 Sequential Quantile Estimation Directories

This directory hierarchy is sorted roughly, but not completely, alphabetically:

shared	11
source	13

Chapter 2

Sequential Quantile Estimation Namespace Index

2.1 Sequential Quantile Estimation Namespace List

Here is a list of all namespaces with brief descriptions:

lib_signals	15
logInfo	18

Chapter 3

Sequential Quantile Estimation Hierarchical Index

3.1 Sequential Quantile Estimation Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

akaroa_import	19
controller	50
error_in_FCM	64
homogeneityTest	72
AndersonDarlingKSampleTestEqualECDFSize	23
KolmogorovSmirnov2SampleTest	80
interface_multipleRuns	73
interface_singleRun	76
K_d_entry	78
method_factory	84
outputAnalyser	86
batching	37
evolution	66
quantile_estimation	103
batch_mean_QE	29
pooling_QE	89
spectral_analysis_QE	151
truncation_point_detection	181
deterministic_TPD	59
sequential_TPD	125
prng	97
quantile_rank	108
resultInfo	124
SequentialStoppingCriteria_QE	136
confidenceInterval_SSC_QE	46
deterministic_SSC_QE	54
relativeErrorQuantile_SSC_QE	115
relativeErrorRange_SSC_QE	120
SequentialStoppingCriteria_QE::estimate	140
setting	142

settingEntry	146
statistic_collection	159
system_command	176

Chapter 4

Sequential Quantile Estimation Data Structure Index

4.1 Sequential Quantile Estimation Data Structures

Here are the data structures with brief descriptions:

<code>akaroa_import</code>	19
<code>AndersonDarlingKSampleTestEqualECDFSize</code>	23
<code>batch_mean_QE</code>	29
<code>batching</code>	37
<code>confidenceInterval_SSC_QE</code>	46
<code>controller</code>	50
<code>deterministic_SSC_QE</code>	54
<code>deterministic_TPD</code>	59
<code>error_in_FCM</code>	64
<code>evolution</code>	66
<code>homogeneityTest</code>	72
<code>interface_multipleRuns</code>	73
<code>interface_singleRun</code>	76
<code>K_d_entry</code>	78
<code>KolmogorovSmirnov2SampleTest</code>	80
<code>method_factory</code>	84
<code>outputAnalyser</code>	86
<code>pooling_QE</code>	89
<code>prng</code>	97
<code>quantile_estimation</code>	103
<code>quantile_rank</code>	108
<code>relativeErrorQuantile_SSC_QE</code>	115
<code>relativeErrorRange_SSC_QE</code>	120
<code>resultInfo</code>	124
<code>sequential_TPD</code>	125
<code>SequentialStoppingCriteria_QE</code>	136
<code>SequentialStoppingCriteria_QE::estimate</code>	140
<code>setting</code>	142
<code>settingEntry</code>	146
<code>spectral_analysis_QE</code>	151
<code>statistic_collection</code>	159

<code>system_command</code>	176
<code>truncation_point_detection</code>	181

Chapter 5

Sequential Quantile Estimation File Index

5.1 Sequential Quantile Estimation File List

Here is a list of all files with brief descriptions:

akaroa_import.cc	185
akaroa_import.h	187
basic.cc	188
basic.h	195
batching.cc	204
batching.h	205
controller.cc	206
controller.h	207
environment.h	208
error.cc	210
error.h	211
homogeneityTests.cc	212
homogeneityTests.h	213
interface.cc	214
interface.h	215
logfile.cc	217
logfile.h	218
main.cc	220
main.h	222
measure.cc	223
measure.h	224
method_factory.cc	229
method_factory.h	230
prng.cc	231
prng.h	233
quantile_estimation.cc	234
quantile_estimation.h	235
resultfile.cc	236
resultfile.h	237
setting.cc	238
setting.h	239

signal_interface.cc	241
signal_interface.h	242
statistic.cc	243
statistic.h	244
system_command.cc	246
system_command.h	247
time_evolution.cc	248
time_evolution.h	249
truncation_point_detection.cc	250
truncation_point_detection.h	251

Chapter 6

Sequential Quantile Estimation Directory Documentation

6.1 /home/cosc/student/mei16/archiv/projects/distribution- Estimation/shared/ Directory Reference

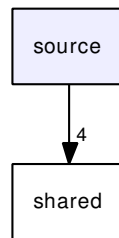
shared

Files

- file **akaroa_import.cc**
- file **akaroa_import.h**
- file **environment.h**
- file **error.cc**
- file **error.h**
- file **interface.cc**
- file **interface.h**
- file **logfile.cc**
- file **logfile.h**
- file **measure.cc**
- file **measure.h**
- file **prng.cc**
- file **prng.h**
- file **resultfile.cc**
- file **resultfile.h**
- file **setting.cc**
- file **setting.h**
- file **signal_interface.cc**
- file **signal_interface.h**
- file **statistic.cc**
- file **statistic.h**

- file `system_command.cc`
- file `system_command.h`

6.2 source/ Directory Reference



Files

- file **basic.cc**
- file **basic.h**
- file **batching.cc**
- file **batching.h**
- file **controller.cc**
- file **controller.h**
- file **homogeneityTests.cc**
- file **homogeneityTests.h**
- file **main.cc**
- file **main.h**
- file **method_factory.cc**
- file **method_factory.h**
- file **quantile_estimation.cc**
- file **quantile_estimation.h**
- file **time_evolution.cc**
- file **time_evolution.h**
- file **truncation_point_detection.cc**
- file **truncation_point_detection.h**

Chapter 7

Sequential Quantile Estimation Namespace Documentation

7.1 lib_signals Namespace Reference

Functions

- void **initializeUserDefinedSignals** (void)
- void **signal_stop** (int signr)
- void **signal_ignore** (int signr)
- void **registerChildProcess** (pid_t newProcess)
- void **sendSignalToAllChildProcesses** (int signr)

Variables

- bool **continueExecution** = true
- unsigned int **actNoChildProcesses** = 0
- const unsigned int **maxNoChildProcesses** = 1024
- pid_t **ChildProcessPIDs** [maxNoChildProcesses]
- bool **continueExecution**
- const unsigned int **maxNoChildProcesses**

7.1.1 Function Documentation

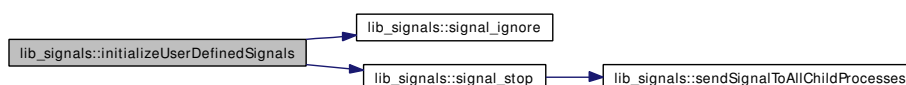
7.1.1.1 void lib_signals::initializeUserDefinedSignals (void)

Definition at line 15 of file signal_interface.cc.

References `signal_ignore()`, and `signal_stop()`.

Referenced by `main()`.

Here is the call graph for this function:



7.1.1.2 void lib_signals::registerChildProcess (pid_t newProcess)

Definition at line 38 of file signal_interface.cc.

References actNoChildProcesses, and ChildProcessPIDs.

7.1.1.3 void lib_signals::sendSignalToAllChildProcesses (int signr = SIGKILL)

Definition at line 42 of file signal_interface.cc.

References actNoChildProcesses, and ChildProcessPIDs.

Referenced by main(), and signal_stop().

7.1.1.4 void lib_signals::signal_ignore (int signr)

Definition at line 33 of file signal_interface.cc.

Referenced by initializeUserDefinedSignals().

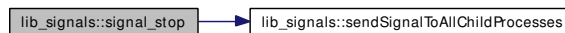
7.1.1.5 void lib_signals::signal_stop (int signr)

Definition at line 26 of file signal_interface.cc.

References continueExecution, and sendSignalToAllChildProcesses().

Referenced by initializeUserDefinedSignals().

Here is the call graph for this function:

**7.1.2 Variable Documentation****7.1.2.1 unsigned int lib_signals::actNoChildProcesses = 0**

Definition at line 11 of file signal_interface.cc.

Referenced by registerChildProcess(), and sendSignalToAllChildProcesses().

7.1.2.2 pid_t lib_signals::ChildProcessPIDs[maxNoChildProcesses]

Definition at line 13 of file signal_interface.cc.

Referenced by registerChildProcess(), and sendSignalToAllChildProcesses().

7.1.2.3 bool lib_signals::continueExecution

Definition at line 10 of file signal_interface.cc.

Referenced by main(), and signal_stop().

7.1.2.4 bool lib_signals::continueExecution = true

Definition at line 10 of file signal_interface.cc.

Referenced by main(), and signal_stop().

7.1.2.5 const unsigned int lib_signals::maxNoChildProcesses

Definition at line 12 of file signal_interface.cc.

7.1.2.6 const unsigned int lib_signals::maxNoChildProcesses = 1024

Definition at line 12 of file signal_interface.cc.

7.2 logInfo Namespace Reference

Functions

- void **open** (void)
- void **close** (void)

7.2.1 Function Documentation

7.2.1.1 void logInfo::close (void)

Definition at line 14 of file logfile.cc.

References logfile.

Referenced by main().

7.2.1.2 void logInfo::open (void)

Definition at line 8 of file logfile.cc.

References logfile.

Referenced by main().

Chapter 8

Sequential Quantile Estimation Data Structure Documentation

8.1 akaroa_import Class Reference

```
#include <akaroa_import.h>
```

Public Types

- enum { **SLOPE_PROTECTION_OFF** = 0, **SLOPE_PROTECTION_UNCONDITIONAL** = 1, **SLOPE_PROTECTION_CONDITIONAL** = 2 }

Public Member Functions

- long double **SchrubenStatistic** (long double X[], int n_t, int n_v, long double sigma_sq)
- void **SpectralVarianceAnalysisOfMean** (long double X[], int N, long double &sigma_sq, int &kappa)
- void **SpectralVarianceAnalysisOfProcess** (long double X[], int N, long double &sigma_sq, int &kappa)
- void **CalculatePeriodogram** (long double X[], int n, long double P[], int nP)
- void **LogAveragePairsAndOffset** (long double P[], long double Lfj[], int K, long double offset)
- void **LookUp_K_d** (int K, int d, long double &C1, int &C2)
- long double **LeastSquaresPolyAt0** (long double x[], long double f[], int N, int k, long double &dp0)
- void **OrthogonalPolynomialTables** (long double x[], int k, int N, long double *P[], long double A[], long double B[])
- void **OrthogonalPolynomialValues** (long double A[], long double B[], int k, int N, long double x, long double P[], long double dP[])
- long double **sqr** (long double x)
- long double **Z** (long double p)
- long double **t_distribution** (int ndf, long double p)

8.1.1 Detailed Description

Definition at line 4 of file akaroa_import.h.

8.1.2 Member Enumeration Documentation

8.1.2.1 anonymous enum

Enumerator:

SLOPE_PROTECTION_OFF
SLOPE_PROTECTION_UNCONDITIONAL
SLOPE_PROTECTION_CONDITIONAL

Definition at line 16 of file akaroa_import.h.

8.1.3 Member Function Documentation

8.1.3.1 long double akaroa_import::SchrubenStatistic (long double $X[]$, int n_t , int n_v , long double $sigma_sq$)

Definition at line 13 of file akaroa_import.cc.

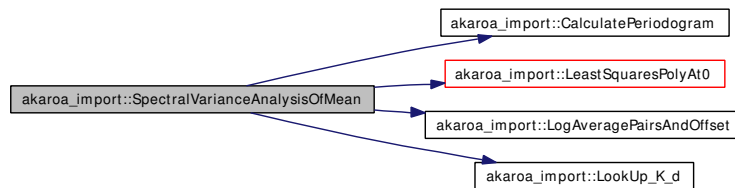
8.1.3.2 void akaroa_import::SpectralVarianceAnalysisOfMean (long double $X[]$, int N , long double & $sigma_sq$, int & $kappa$)

Definition at line 31 of file akaroa_import.cc.

References CalculatePeriodogram(), LeastSquaresPolyAt0(), LogAveragePairsAndOffset(), LookUp_K_d(), SLOPE_PROTECTION_CONDITIONAL, and SLOPE_PROTECTION_UNCONDITIONAL.

Referenced by spectral_analysis_QE::checkQuantiles().

Here is the call graph for this function:

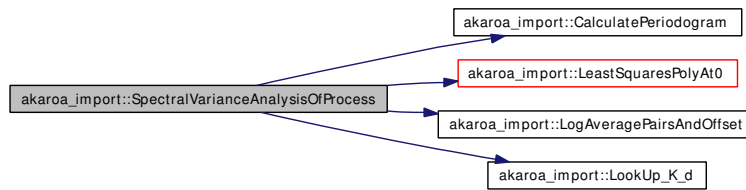


8.1.3.3 void akaroa_import::SpectralVarianceAnalysisOfProcess (long double $X[]$, int N , long double & $sigma_sq$, int & $kappa$)

Definition at line 81 of file akaroa_import.cc.

References CalculatePeriodogram(), LeastSquaresPolyAt0(), LogAveragePairsAndOffset(), LookUp_K_d(), SLOPE_PROTECTION_CONDITIONAL, and SLOPE_PROTECTION_UNCONDITIONAL.

Here is the call graph for this function:



8.1.3.4 void akaroa_import::CalculatePeriodogram (long double $X[]$, int n , long double $P[]$, int nP)

Definition at line 131 of file akaroa_import.cc.

Referenced by SpectralVarianceAnalysisOfMean(), and SpectralVarianceAnalysisOfProcess().

8.1.3.5 void akaroa_import::LogAveragePairsAndOffset (long double $P[]$, long double $Lfj[]$, int K , long double $offset$)

Definition at line 143 of file akaroa_import.cc.

Referenced by SpectralVarianceAnalysisOfMean(), and SpectralVarianceAnalysisOfProcess().

8.1.3.6 void akaroa_import::LookUp_K_d (int K , int d , long double & $C1$, int & $C2$)

Definition at line 167 of file akaroa_import.cc.

References K_d_entry::C1, K_d_entry::C2, K_d_entry::d, K_d_entry::K, and K_d_table.

Referenced by SpectralVarianceAnalysisOfMean(), and SpectralVarianceAnalysisOfProcess().

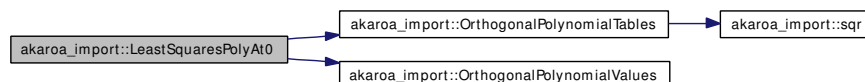
8.1.3.7 long double akaroa_import::LeastSquaresPolyAt0 (long double $x[]$, long double $f[]$, int N , int k , long double & $dp0$)

Definition at line 190 of file akaroa_import.cc.

References OrthogonalPolynomialTables(), and OrthogonalPolynomialValues().

Referenced by SpectralVarianceAnalysisOfMean(), and SpectralVarianceAnalysisOfProcess().

Here is the call graph for this function:



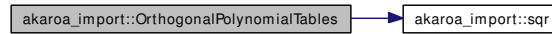
8.1.3.8 void akaroa_import::OrthogonalPolynomialTables (long double $x[]$, int k , int N , long double * $P[]$, long double $A[]$, long double $B[]$)

Definition at line 231 of file akaroa_import.cc.

References `sqr()`.

Referenced by `LeastSquaresPolyAt0()`.

Here is the call graph for this function:



8.1.3.9 void akaroa_import::OrthogonalPolynomialValues (long double $A[]$, long double $B[]$, int k , int N , long double x , long double $P[]$, long double $dP[]$)

Definition at line 264 of file `akaroa_import.cc`.

Referenced by `LeastSquaresPolyAt0()`.

8.1.3.10 long double akaroa_import::sqr (long double x) [inline]

Definition at line 15 of file `akaroa_import.h`.

Referenced by `OrthogonalPolynomialTables()`.

8.1.3.11 long double akaroa_import::Z (long double p)

Definition at line 276 of file `akaroa_import.cc`.

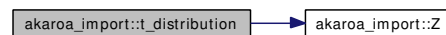
Referenced by `t_distribution()`.

8.1.3.12 long double akaroa_import::t_distribution (int ndf , long double p)

Definition at line 289 of file `akaroa_import.cc`.

References `Z()`.

Here is the call graph for this function:



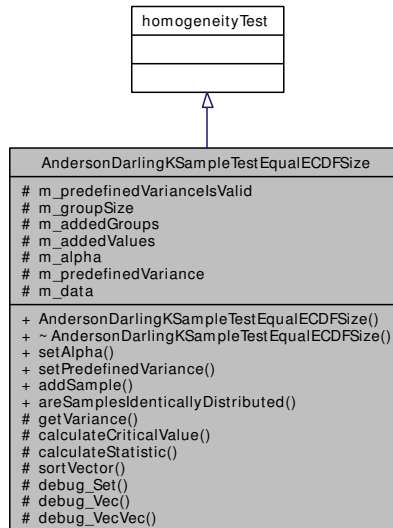
The documentation for this class was generated from the following files:

- `akaroa_import.h`
- `akaroa_import.cc`

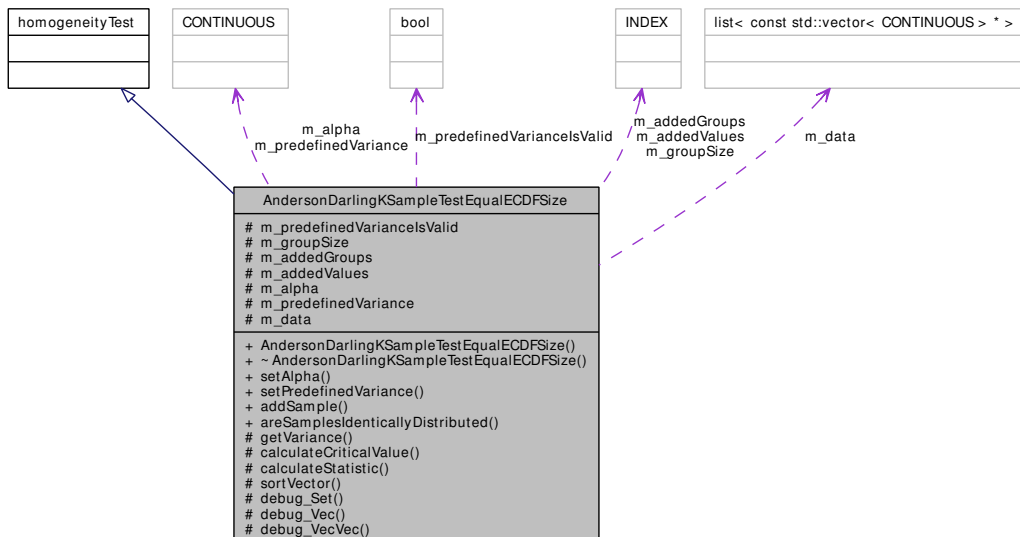
8.2 AndersonDarlingKSampleTestEqualECDFSize Class Reference

```
#include <homogeneityTests.h>
```

Inheritance diagram for AndersonDarlingKSampleTestEqualECDFSize:



Collaboration diagram for AndersonDarlingKSampleTestEqualECDFSize:



Public Member Functions

- **AndersonDarlingKSampleTestEqualECDFSize** (INDEX newGroupSize)
- **~AndersonDarlingKSampleTestEqualECDFSize** (void)
- **void setAlpha** (CONTINUOUS newAlpha)

- void **setPredefinedVariance** (CONTINUOUS newVariance)
- void **addSample** (const std::vector< CONTINUOUS > &)
- bool **areSamplesIdenticallyDistributed** (CONTINUOUS &std_statistic, CONTINUOUS &criticalValue, CONTINUOUS &variance, CONTINUOUS &statistic, CONTINUOUS &resultValue)

Protected Member Functions

- CONTINUOUS **getVariance** (void)
- CONTINUOUS **calculateCriticalValue** (void)
- CONTINUOUS **calculateStatistic** (void)
- void **sortVector** (std::vector< CONTINUOUS > &)
- void **debug_Set** (std::set< CONTINUOUS > &)
- void **debug_Vec** (std::vector< CONTINUOUS > &)
- void **debug_VecVec** (std::vector< std::vector< CONTINUOUS > > &)

Protected Attributes

- bool **m_predefinedVarianceIsValid**
- INDEX **m_groupSize**
- INDEX **m_addedGroups**
- INDEX **m_addedValues**
- CONTINUOUS **m_alpha**
- CONTINUOUS **m_predefinedVariance**
- std::list< const std::vector< CONTINUOUS > * > **m_data**

8.2.1 Detailed Description

Definition at line 16 of file homogeneityTests.h.

8.2.2 Constructor & Destructor Documentation

8.2.2.1 AndersonDarlingKSampleTestEqualECDFSize::AndersonDarlingKSampleTestEqualECDFSize (INDEX *newGroupSize*)

Definition at line 7 of file homogeneityTests.cc.

References `m_groupSize`.

8.2.2.2 AndersonDarlingKSampleTestEqualECDFSize::~AndersonDarlingKSampleTestEqualECDFSize (void)

Definition at line 16 of file homogeneityTests.cc.

8.2.3 Member Function Documentation

8.2.3.1 void AndersonDarlingKSampleTestEqualECDFSize::setAlpha (CONTINUOUS *newAlpha*)

Definition at line 19 of file homogeneityTests.cc.

References `m_alpha`.

Referenced by `sequential_TPD::homogeneityTest()`.

8.2.3.2 void AndersonDarlingKSampleTestEqualECDFSize::setPredefinedVariance (CONTINUOUS *newVariance*)

Definition at line 31 of file homogeneityTests.cc.

References `m_predefinedVariance`, and `m_predefinedVarianceIsValid`.

Referenced by `sequential_TPD::homogeneityTest()`.

8.2.3.3 void AndersonDarlingKSampleTestEqualECDFSize::addSample (const `std::vector< CONTINUOUS >` &)

Definition at line 36 of file homogeneityTests.cc.

References `m_addedGroups`, `m_addedValues`, `m_data`, and `m_groupSize`.

Referenced by `sequential_TPD::homogeneityTest()`.

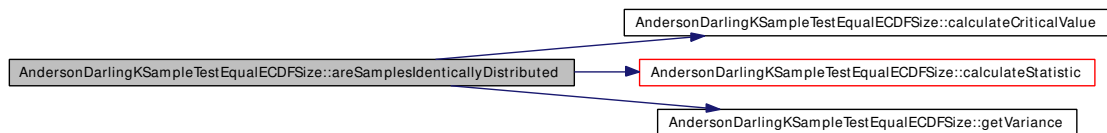
8.2.3.4 bool AndersonDarlingKSampleTestEqualECDFSize::areSamplesIdenticallyDistributed (CONTINUOUS & *std_statistic*, CONTINUOUS & *criticalValue*, CONTINUOUS & *variance*, CONTINUOUS & *statistic*, CONTINUOUS & *resultValue*)

Definition at line 44 of file homogeneityTests.cc.

References `calculateCriticalValue()`, `calculateStatistic()`, `getVariance()`, and `m_addedGroups`.

Referenced by `sequential_TPD::homogeneityTest()`.

Here is the call graph for this function:



8.2.3.5 CONTINUOUS AndersonDarlingKSampleTestEqualECDFSize::getVariance (void) [protected]

Definition at line 68 of file homogeneityTests.cc.

References `CONTINUOUS`, `m_addedGroups`, `m_addedValues`, `m_groupSize`, `m_predefinedVariance`, and `m_predefinedVarianceIsValid`.

Referenced by `areSamplesIdenticallyDistributed()`.

8.2.3.6 CONTINUOUS `AndersonDarlingKSampleTestEqualECDFSize::calculateCriticalValue` (void) [protected]

Definition at line 96 of file `homogeneityTests.cc`.

References CONTINUOUS, `m_addedGroups`, and `m_alpha`.

Referenced by `areSamplesIdenticallyDistributed()`.

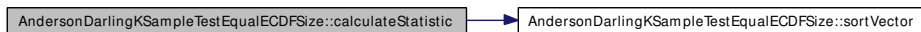
8.2.3.7 CONTINUOUS `AndersonDarlingKSampleTestEqualECDFSize::calculateStatistic` (void) [protected]

Definition at line 121 of file `homogeneityTests.cc`.

References CONTINUOUS, INDEX, `m_addedGroups`, `m_addedValues`, `m_data`, `m_groupSize`, and `sortVector()`.

Referenced by `areSamplesIdenticallyDistributed()`.

Here is the call graph for this function:



8.2.3.8 void `AndersonDarlingKSampleTestEqualECDFSize::sortVector` (`std::vector< CONTINUOUS > &`) [protected]

Definition at line 109 of file `homogeneityTests.cc`.

References INDEX.

Referenced by `calculateStatistic()`.

8.2.3.9 void `AndersonDarlingKSampleTestEqualECDFSize::debug_Set` (`std::set< CONTINUOUS > &`) [protected]

Definition at line 231 of file `homogeneityTests.cc`.

8.2.3.10 void `AndersonDarlingKSampleTestEqualECDFSize::debug_Vec` (`std::vector< CONTINUOUS > &`) [protected]

Definition at line 241 of file `homogeneityTests.cc`.

8.2.3.11 void `AndersonDarlingKSampleTestEqualECDFSize::debug_VecVec` (`std::vector< std::vector< CONTINUOUS > > &`) [protected]

Definition at line 249 of file `homogeneityTests.cc`.

References INDEX.

8.2.4 Field Documentation

8.2.4.1 `bool AndersonDarlingKSampleTestEqualECDFSize::m_predefinedVarianceIsValid` [protected]

Definition at line 31 of file `homogeneityTests.h`.

Referenced by `getVariance()`, and `setPredefinedVariance()`.

8.2.4.2 `INDEX AndersonDarlingKSampleTestEqualECDFSize::m_groupSize` [protected]

Definition at line 32 of file `homogeneityTests.h`.

Referenced by `addSample()`, `AndersonDarlingKSampleTestEqualECDFSize()`, `calculateStatistic()`, and `getVariance()`.

8.2.4.3 `INDEX AndersonDarlingKSampleTestEqualECDFSize::m_addedGroups` [protected]

Definition at line 33 of file `homogeneityTests.h`.

Referenced by `addSample()`, `areSamplesIdenticallyDistributed()`, `calculateCriticalValue()`, `calculateStatistic()`, and `getVariance()`.

8.2.4.4 `INDEX AndersonDarlingKSampleTestEqualECDFSize::m_addedValues` [protected]

Definition at line 34 of file `homogeneityTests.h`.

Referenced by `addSample()`, `calculateStatistic()`, and `getVariance()`.

8.2.4.5 `CONTINUOUS AndersonDarlingKSampleTestEqualECDFSize::m_alpha` [protected]

Definition at line 35 of file `homogeneityTests.h`.

Referenced by `calculateCriticalValue()`, and `setAlpha()`.

8.2.4.6 `CONTINUOUS AndersonDarlingKSampleTestEqualECDFSize::m_predefinedVariance` [protected]

Definition at line 36 of file `homogeneityTests.h`.

Referenced by `getVariance()`, and `setPredefinedVariance()`.

8.2.4.7 `std::list< const std::vector<CONTINUOUS>* > AndersonDarlingKSampleTestEqualECDFSize::m_data` [protected]

Definition at line 37 of file `homogeneityTests.h`.

Referenced by `addSample()`, and `calculateStatistic()`.

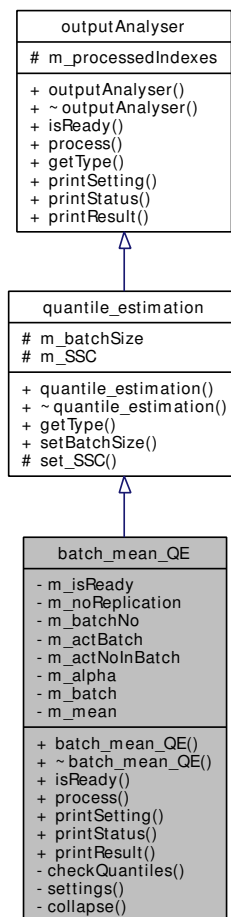
The documentation for this class was generated from the following files:

- `homogeneityTests.h`
- `homogeneityTests.cc`

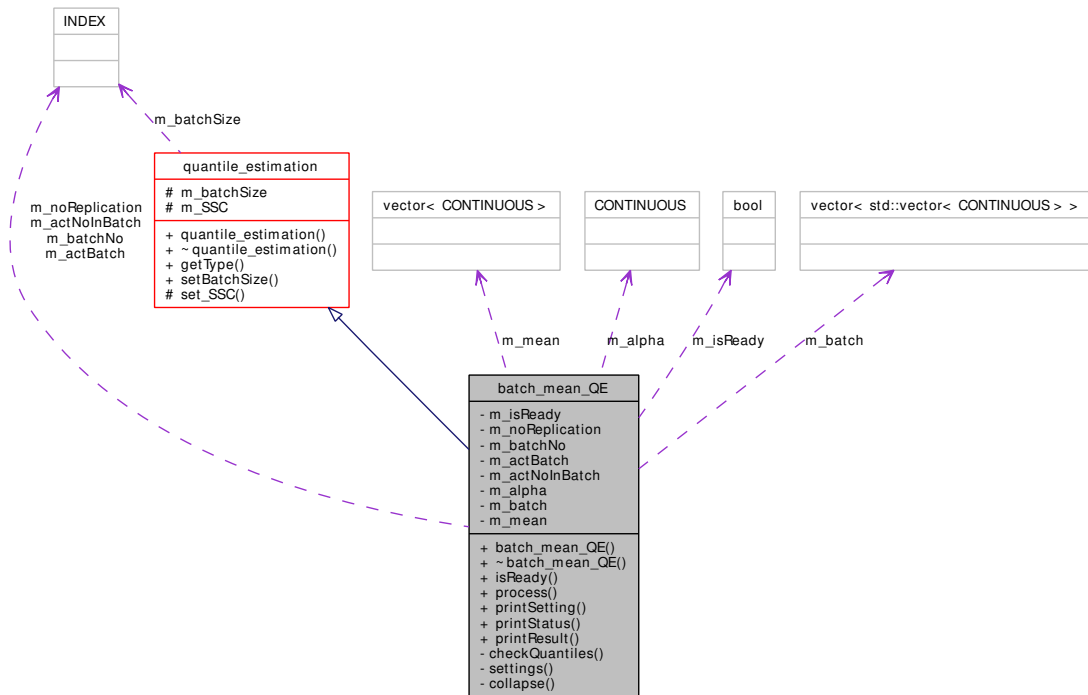
8.3 batch_mean_QE Class Reference

```
#include <quantile_estimation.h>
```

Inheritance diagram for batch_mean_QE:



Collaboration diagram for batch_mean_QE:



Public Member Functions

- `batch_mean_QE` (void)
- `~batch_mean_QE` (void)
- `bool isReady` (void) const
- `void process` (const std::list< CONTINUOUS > &)
- `void printSetting` (void)
- `void printStatus` (void)
- `void printResult` (void)
- virtual `TypeOfMethod getType` (void) const
- `void setBatchSize` (INDEX p)

Protected Member Functions

- `void set_SSC` (void)

Protected Attributes

- INDEX `m_batchSize`
- SequentialStoppingCriteria_QE * `m_SSC`
- INDEX `m_processedIndexes`

Private Member Functions

- `bool checkQuantiles` (void)
- `void settings` (void)
- `void collapse` (void)

Private Attributes

- bool `m_isReady`
- INDEX `m_noReplication`
- INDEX `m_batchNo`
- INDEX `m_actBatch`
- INDEX `m_actNoInBatch`
- CONTINUOUS `m_alpha`
- `std::vector< std::vector< CONTINUOUS > >` `m_batch`
- `std::vector< CONTINUOUS >` `m_mean`

8.3.1 Detailed Description

Definition at line 54 of file `quantile_estimation.h`.

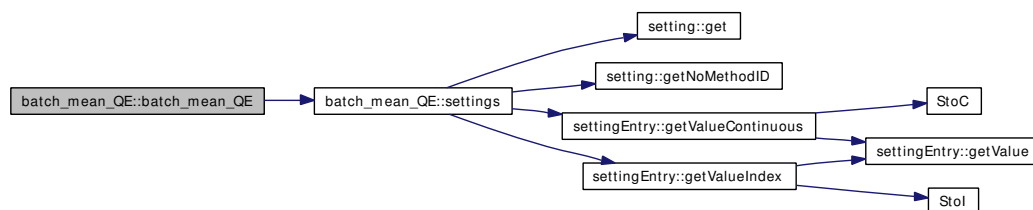
8.3.2 Constructor & Destructor Documentation

8.3.2.1 `batch_mean_QE::batch_mean_QE (void)`

Definition at line 228 of file `quantile_estimation.cc`.

References `settings()`.

Here is the call graph for this function:



8.3.2.2 `batch_mean_QE::~~batch_mean_QE (void)`

Definition at line 239 of file `quantile_estimation.cc`.

8.3.3 Member Function Documentation

8.3.3.1 `bool batch_mean_QE::isReady (void) const [virtual]`

Reimplemented from `outputAnalyser` (p. 183).

Definition at line 242 of file `quantile_estimation.cc`.

References `m_isReady`.

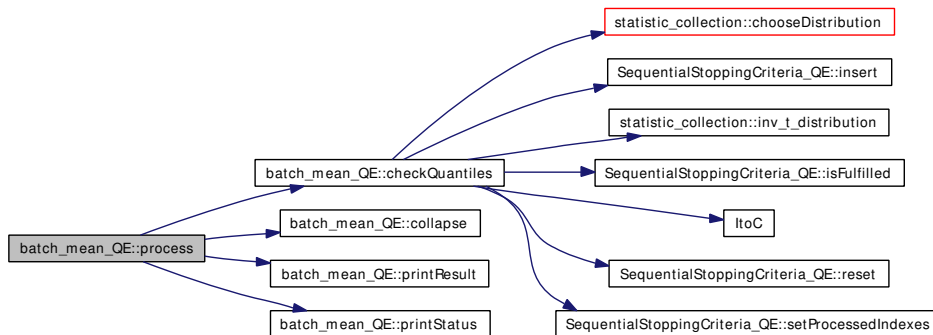
8.3.3.2 void batch_mean_QE::process (const std::list< CONTINUOUS > &) [virtual]

Reimplemented from **outputAnalyser** (p. 183).

Definition at line 246 of file quantile_estimation.cc.

References checkQuantiles(), collapse(), INDEX, m_actBatch, m_actNoInBatch, m_batch, m_batchNo, quantile_estimation::m_batchSize, m_isReady, m_mean, m_noReplication, outputAnalyser::m_processedIndexes, printResult(), and printStatus().

Here is the call graph for this function:



8.3.3.3 void batch_mean_QE::printSetting (void) [virtual]

Reimplemented from **outputAnalyser** (p. 183).

Definition at line 294 of file quantile_estimation.cc.

References logfile, m_batchNo, s_batch_mean_QE, s_batches, s_execute, and s_yes.

8.3.3.4 void batch_mean_QE::printStatus (void) [virtual]

Reimplemented from **outputAnalyser** (p. 183).

Definition at line 305 of file quantile_estimation.cc.

References logfile, m_actBatch, m_actNoInBatch, m_alpha, m_batchNo, quantile_estimation::m_batchSize, m_noReplication, outputAnalyser::m_processedIndexes, and s_batch_mean_QE.

Referenced by process().

8.3.3.5 void batch_mean_QE::printResult (void) [virtual]

Reimplemented from **outputAnalyser** (p. 183).

Definition at line 317 of file quantile_estimation.cc.

References logfile, m_actBatch, m_actNoInBatch, m_alpha, m_batchNo, quantile_estimation::m_batchSize, m_noReplication, outputAnalyser::m_processedIndexes, and s_batch_mean_QE.

Referenced by process().

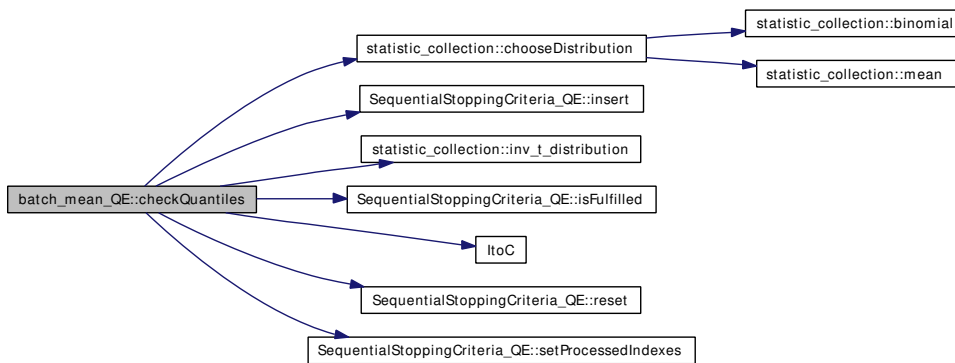
8.3.3.6 bool batch_mean_QE::checkQuantiles (void) [private]

Definition at line 329 of file quantile_estimation.cc.

References `statistic_collection::chooseDistribution()`, `CONTINUOUS`, `INDEX`, `SequentialStoppingCriteria_QE::insert()`, `statistic_collection::inv_t_distribution()`, `SequentialStoppingCriteria_QE::isFulfilled()`, `ItoC()`, `lib_statistic`, `m_alpha`, `m_batch`, `m_batchNo`, `quantile_estimation::m_batchSize`, `m_mean`, `m_noReplication`, `outputAnalyser::m_processedIndexes`, `quantile_estimation::m_SSC`, `SequentialStoppingCriteria_QE::reset()`, and `SequentialStoppingCriteria_QE::setProcessedIndexes()`.

Referenced by `process()`.

Here is the call graph for this function:



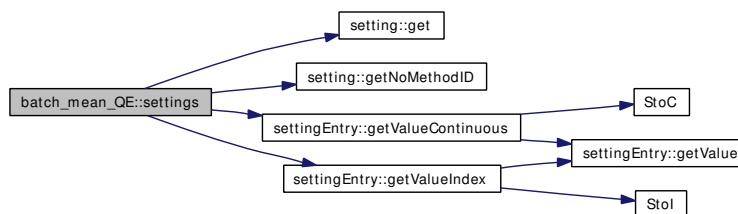
8.3.3.7 void batch_mean_QE::settings (void) [private]

Definition at line 375 of file quantile_estimation.cc.

References `setting::get()`, `setting::getNoMethodID()`, `settingEntry::getValueContinuous()`, `settingEntry::getValueIndex()`, `lib_setting`, `m_alpha`, `m_batchNo`, `m_noReplication`, `s_alpha`, `s_batch_mean_QE`, `s_batches`, and `s_replications`.

Referenced by `batch_mean_QE()`.

Here is the call graph for this function:



8.3.3.8 void batch_mean_QE::collapse (void) [private]

Definition at line 398 of file quantile_estimation.cc.

References INDEX, m_actBatch, m_actNoInBatch, m_batch, m_batchNo, quantile_estimation::m_batchSize, and m_noReplication.

Referenced by process().

8.3.3.9 TypeOfMethod quantile_estimation::getType (void) const [virtual, inherited]

Reimplemented from **outputAnalyser** (p. 87).

Definition at line 22 of file quantile_estimation.cc.

References ESTIMATOR.

8.3.3.10 void quantile_estimation::setBatchSize (INDEX p) [inherited]

Definition at line 26 of file quantile_estimation.cc.

References quantile_estimation::m_batchSize.

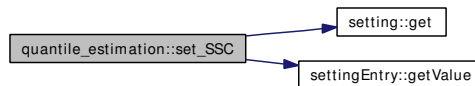
8.3.3.11 void quantile_estimation::set_SSC (void) [protected, inherited]

Definition at line 31 of file quantile_estimation.cc.

References setting::get(), settingEntry::getValue(), lib_setting, quantile_estimation::m_SSC, s_confidenceInterval_SSC_QE, s_deterministic_SSC_QE, s_execute, s_relativeErrorQuantile_SSC_QE, s_relativeErrorRange_SSC_QE, and s_yes.

Referenced by quantile_estimation::quantile_estimation().

Here is the call graph for this function:



8.3.4 Field Documentation

8.3.4.1 bool batch_mean_QE::m_isReady [private]

Definition at line 70 of file quantile_estimation.h.

Referenced by isReady(), and process().

8.3.4.2 INDEX batch_mean_QE::m_noReplication [private]

Definition at line 71 of file quantile_estimation.h.

Referenced by checkQuantiles(), collapse(), printResult(), printStatus(), process(), and settings().

8.3.4.3 INDEX batch_mean_QE::m_batchNo [private]

Definition at line 72 of file quantile_estimation.h.

Referenced by `checkQuantiles()`, `collapse()`, `printResult()`, `printSetting()`, `printStatus()`, `process()`, and `settings()`.

8.3.4.4 INDEX `batch_mean_QE::m_actBatch` [private]

Definition at line 73 of file `quantile_estimation.h`.

Referenced by `collapse()`, `printResult()`, `printStatus()`, and `process()`.

8.3.4.5 INDEX `batch_mean_QE::m_actNoInBatch` [private]

Definition at line 74 of file `quantile_estimation.h`.

Referenced by `collapse()`, `printResult()`, `printStatus()`, and `process()`.

8.3.4.6 CONTINUOUS `batch_mean_QE::m_alpha` [private]

Definition at line 75 of file `quantile_estimation.h`.

Referenced by `checkQuantiles()`, `printResult()`, `printStatus()`, and `settings()`.

8.3.4.7 `std::vector< std::vector<CONTINUOUS> > batch_mean_QE::m_batch` [private]

Definition at line 76 of file `quantile_estimation.h`.

Referenced by `checkQuantiles()`, `collapse()`, and `process()`.

8.3.4.8 `std::vector< CONTINUOUS > batch_mean_QE::m_mean` [private]

Definition at line 77 of file `quantile_estimation.h`.

Referenced by `checkQuantiles()`, and `process()`.

8.3.4.9 INDEX `quantile_estimation::m_batchSize` [protected, inherited]

Definition at line 26 of file `quantile_estimation.h`.

Referenced by `spectral_analysis_QE::checkQuantiles()`, `checkQuantiles()`, `spectral_analysis_QE::collapse()`, `collapse()`, `spectral_analysis_QE::printResult()`, `printResult()`, `pooling_QE::printResult()`, `spectral_analysis_QE::printStatus()`, `printStatus()`, `pooling_QE::printStatus()`, `spectral_analysis_QE::process()`, `process()`, `pooling_QE::process()`, and `quantile_estimation::setBatchSize()`.

8.3.4.10 `SequentialStoppingCriteria_QE*` `quantile_estimation::m_SSC` [protected, inherited]

Definition at line 27 of file `quantile_estimation.h`.

Referenced by `spectral_analysis_QE::checkQuantiles()`, `checkQuantiles()`, `pooling_QE::checkQuantiles()`, `quantile_estimation::set_SSC()`, and `quantile_estimation::~~quantile_estimation()`.

8.3.4.11 INDEX outputAnalyser::m_processedIndexes [protected, inherited]

Definition at line 20 of file basic.h.

Referenced by `evolution::calculateQuantiles()`, `spectral_analysis_QE::checkQuantiles()`, `checkQuantiles()`, `pooling_QE::checkQuantiles()`, `deterministic_TPD::isReady()`, `evolution::isReady()`, `sequential_TPD::printResult()`, `deterministic_TPD::printResult()`, `spectral_analysis_QE::printResult()`, `printResult()`, `pooling_QE::printResult()`, `batching::printResult()`, `sequential_TPD::printStatus()`, `deterministic_TPD::printStatus()`, `evolution::printStatus()`, `spectral_analysis_QE::printStatus()`, `printStatus()`, `pooling_QE::printStatus()`, `batching::printStatus()`, `sequential_TPD::process()`, `deterministic_TPD::process()`, `evolution::process()`, `spectral_analysis_QE::process()`, `process()`, `pooling_QE::process()`, `batching::process()`, `outputAnalyser::process()`, `sequential_TPD::sub_collect()`, `sequential_TPD::sub_compare()`, and `sequential_TPD::sub_initialize()`.

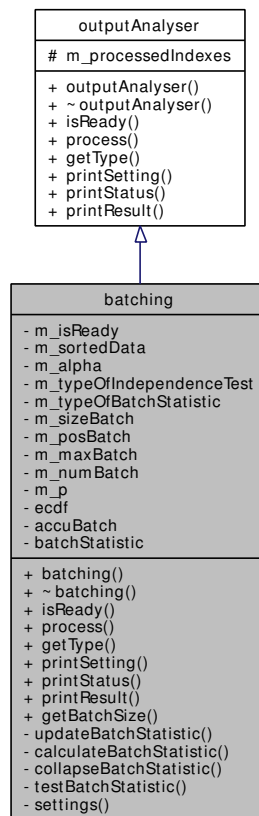
The documentation for this class was generated from the following files:

- **quantile_estimation.h**
- **quantile_estimation.cc**

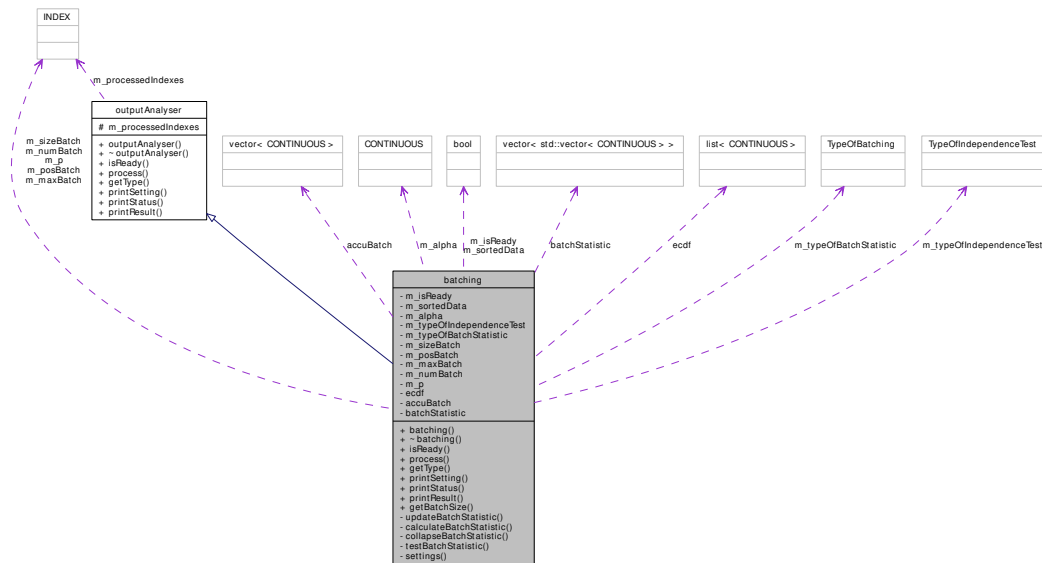
8.4 batching Class Reference

```
#include <batching.h>
```

Inheritance diagram for batching:



Collaboration diagram for batching:



Public Member Functions

- **batching** (void)
- virtual **~batching** (void)
- bool **isReady** (void) const
- void **process** (const std::list< CONTINUOUS > &)
- **TypeOfMethod** **getType** (void) const
- void **printSetting** (void)
- void **printStatus** (void)
- void **printResult** (void)
- INDEX **getBatchSize** (void) const

Protected Attributes

- INDEX **m_processedIndexes**

Private Types

- enum **TypeOfBatching** { Mean, Spacing }

Private Member Functions

- void **updateBatchStatistic** (void)
- void **calculateBatchStatistic** (void)
- void **collapseBatchStatistic** (void)
- bool **testBatchStatistic** (void)
- void **settings** (void)

Private Attributes

- bool `m_isReady`
- bool `m_sortedData`
- CONTINUOUS `m_alpha`
- `statistic_collection::TypeOfIndependenceTest` `m_typeOfIndependenceTest`
- `TypeOfBatching` `m_typeOfBatchStatistic`
- INDEX `m_sizeBatch`
- INDEX `m_posBatch`
- INDEX `m_maxBatch`
- INDEX `m_numBatch`
- INDEX `m_p`
- `std::list< CONTINUOUS >` `ecdf`
- `std::vector< CONTINUOUS >` `accuBatch`
- `std::vector< std::vector< CONTINUOUS > >` `batchStatistic`

8.4.1 Detailed Description

Definition at line 6 of file `batching.h`.

8.4.2 Member Enumeration Documentation

8.4.2.1 enum `batching::TypeOfBatching` [private]

Enumerator:

Mean

Spacing

Definition at line 20 of file `batching.h`.

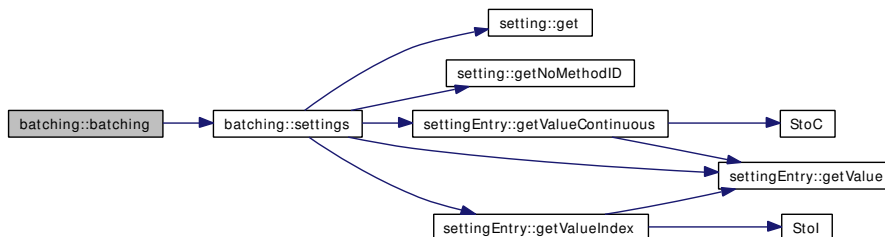
8.4.3 Constructor & Destructor Documentation

8.4.3.1 `batching::batching` (void)

Definition at line 8 of file `batching.cc`.

References `accuBatch`, `batchStatistic`, `INDEX`, `m_maxBatch`, `m_p`, and `settings()`.

Here is the call graph for this function:



8.4.3.2 `batching::~~batching (void) [virtual]`

Definition at line 36 of file `batching.cc`.

8.4.4 Member Function Documentation

8.4.4.1 `bool batching::isReady (void) const [virtual]`

Reimplemented from `outputAnalyser` (p. 183).

Definition at line 39 of file `batching.cc`.

References `m_isReady`.

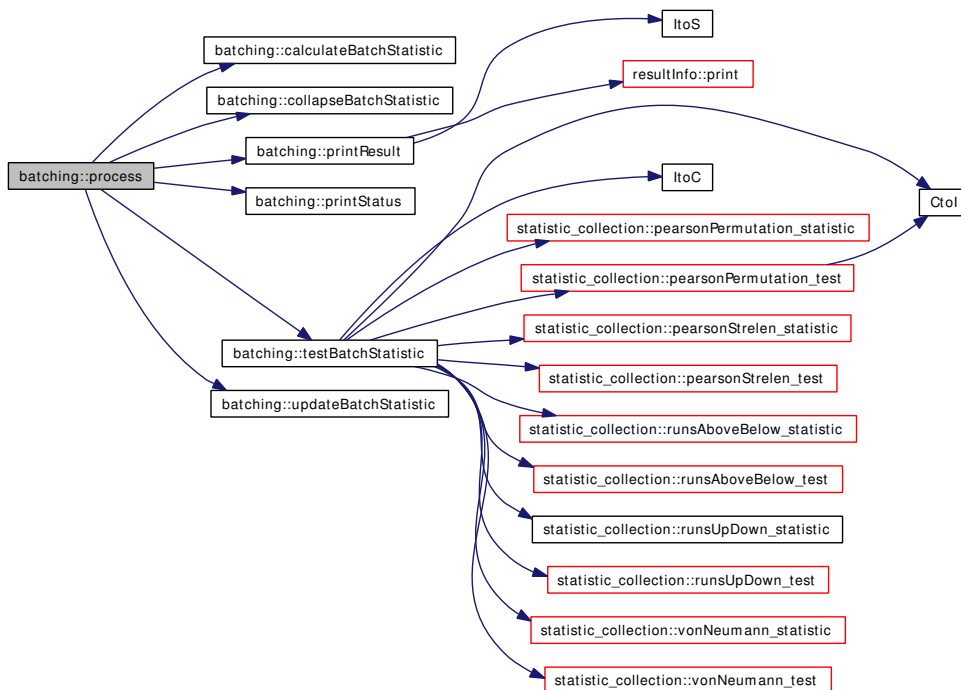
8.4.4.2 `void batching::process (const std::list< CONTINUOUS > &) [virtual]`

Reimplemented from `outputAnalyser` (p. 183).

Definition at line 43 of file `batching.cc`.

References `calculateBatchStatistic()`, `collapseBatchStatistic()`, `ecdf`, `m_isReady`, `m_maxBatch`, `m_numBatch`, `m_p`, `m_posBatch`, `outputAnalyser::m_processedIndexes`, `m_sizeBatch`, `m_sortedData`, `printResult()`, `printStatus()`, `testBatchStatistic()`, and `updateBatchStatistic()`.

Here is the call graph for this function:



8.4.4.3 `TypeOfMethod batching::getType (void) const [virtual]`

Reimplemented from `outputAnalyser` (p. 87).

Definition at line 73 of file batching.cc.

References INDEPENDENT.

8.4.4.4 void batching::printSetting (void) [virtual]

Reimplemented from **outputAnalyser** (p. 183).

Definition at line 77 of file batching.cc.

References logfile, m_alpha, m_maxBatch, m_sortedData, m_typeOfBatchStatistic, m_typeOfIndependenceTest, Mean, statistic_collection::PearsonPermutation, statistic_collection::PearsonStrelen, statistic_collection::RunsAboveBelow, statistic_collection::RunsUpDown, s_alpha, s_batch_max, s_execute, s_independence, s_mean, s_no, s_pearsonPermutation, s_pearsonStrelen, s_runsAboveBelow, s_runsUpDown, s_sequential_batching, s_sort, s_spacing, s_statistic, s_vonNeumann, s_yes, Spacing, and statistic_collection::VonNeuman.

8.4.4.5 void batching::printStatus (void) [virtual]

Reimplemented from **outputAnalyser** (p. 183).

Definition at line 106 of file batching.cc.

References logfile, m_maxBatch, m_numBatch, m_p, m_posBatch, outputAnalyser::m_processedIndexes, m_sizeBatch, and s_sequential_batching.

Referenced by process().

8.4.4.6 void batching::printResult (void) [virtual]

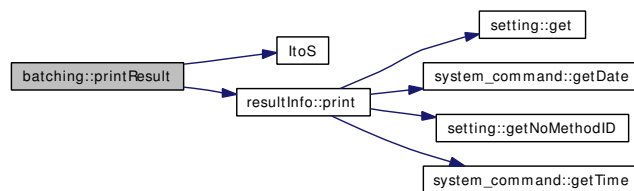
Reimplemented from **outputAnalyser** (p. 183).

Definition at line 117 of file batching.cc.

References ItoS(), logfile, m_maxBatch, m_numBatch, m_p, m_posBatch, outputAnalyser::m_processedIndexes, m_sizeBatch, resultInfo::print(), resultfile, and s_sequential_batching.

Referenced by process().

Here is the call graph for this function:



8.4.4.7 INDEX batching::getBatchSize (void) const

Definition at line 131 of file batching.cc.

References m_isReady, and m_sizeBatch.

Referenced by controller::process().

8.4.4.8 void batching::updateBatchStatistic (void) [private]

Definition at line 136 of file batching.cc.

References `accuBatch`, `ecdf`, `INDEX`, `m_p`, `m_typeOfBatchStatistic`, `Mean`, and `Spacing`.

Referenced by `process()`.

8.4.4.9 void batching::calculateBatchStatistic (void) [private]

Definition at line 150 of file batching.cc.

References `accuBatch`, `batchStatistic`, `ecdf`, `INDEX`, `m_numBatch`, `m_p`, `m_sizeBatch`, `m_typeOfBatchStatistic`, `Mean`, and `Spacing`.

Referenced by `process()`.

8.4.4.10 void batching::collapseBatchStatistic (void) [private]

Definition at line 167 of file batching.cc.

References `batchStatistic`, `INDEX`, `m_maxBatch`, `m_p`, `m_typeOfBatchStatistic`, `Mean`, and `Spacing`.

Referenced by `process()`.

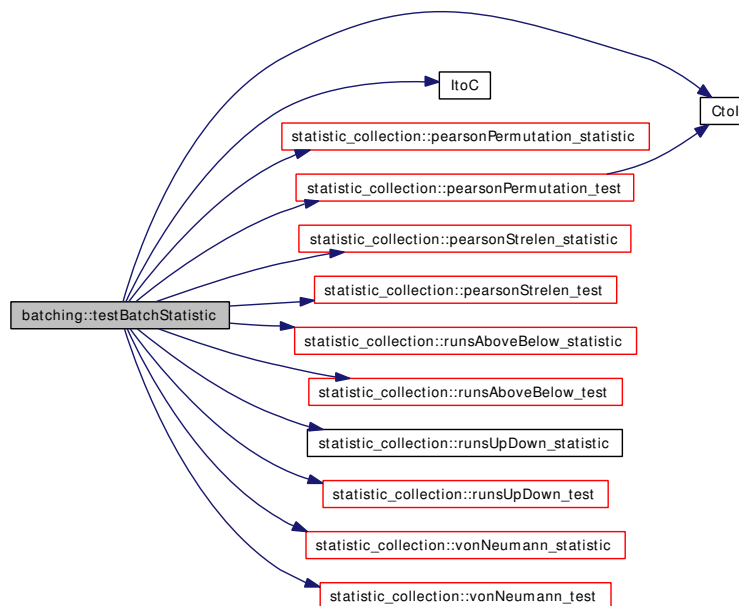
8.4.4.11 bool batching::testBatchStatistic (void) [private]

Definition at line 180 of file batching.cc.

References `batchStatistic`, `CONTINUOUS`, `CtoI()`, `INDEX`, `ItoC()`, `lib_statistic`, `logfile`, `m_alpha`, `m_maxBatch`, `m_p`, `m_sizeBatch`, `m_typeOfIndependenceTest`, `statistic_collection::PearsonPermutation`, `statistic_collection::pearsonPermutation_statistic()`, `statistic_collection::pearsonPermutation_test()`, `statistic_collection::PearsonStrelen`, `statistic_collection::pearsonStrelen_statistic()`, `statistic_collection::pearsonStrelen_test()`, `statistic_collection::runsAboveBelow_statistic()`, `statistic_collection::runsAboveBelow_test()`, `statistic_collection::runsUpDown_statistic()`, `statistic_collection::runsUpDown_test()`, `statistic_collection::VonNeuman`, `statistic_collection::vonNeumann_statistic()`, and `statistic_collection::vonNeumann_test()`.

Referenced by `process()`.

Here is the call graph for this function:



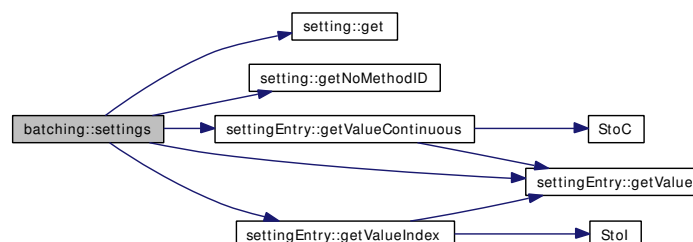
8.4.4.12 void batching::settings (void) [private]

Definition at line 275 of file `batching.cc`.

References `setting::get()`, `setting::getNoMethodID()`, `settingEntry::getValue()`, `settingEntry::getValueContinuous()`, `settingEntry::getValueIndex()`, `lib_setting`, `m_alpha`, `m_maxBatch`, `m_p`, `m_sortedData`, `m_typeOfBatchStatistic`, `m_typeOfIndependenceTest`, `Mean`, `statistic_collection::PearsonPermutation`, `statistic_collection::PearsonStrelen`, `statistic_collection::RunsAboveBelow`, `statistic_collection::RunsUpDown`, `s_alpha`, `s_auto`, `s_batch_max`, `s_independence`, `s_mean`, `s_no`, `s_pearsonPermutation`, `s_pearsonStrelen`, `s_replications`, `s_runsAboveBelow`, `s_runsUpDown`, `s_sequential_batching`, `s_sort`, `s_spacing`, `s_statistic`, `s_vonNeumann`, `s_yes`, `Spacing`, and `statistic_collection::VonNeuman`.

Referenced by `batching()`.

Here is the call graph for this function:



8.4.5 Field Documentation

8.4.5.1 `bool batching::m_isReady` [private]

Definition at line 28 of file `batching.h`.

Referenced by `getBatchSize()`, `isReady()`, and `process()`.

8.4.5.2 `bool batching::m_sortedData` [private]

Definition at line 29 of file `batching.h`.

Referenced by `printSetting()`, `process()`, and `settings()`.

8.4.5.3 `CONTINUOUS batching::m_alpha` [private]

Definition at line 30 of file `batching.h`.

Referenced by `printSetting()`, `settings()`, and `testBatchStatistic()`.

8.4.5.4 `statistic_collection::TypeOfIndependenceTest batching::m_typeOfIndependenceTest` [private]

Definition at line 31 of file `batching.h`.

Referenced by `printSetting()`, `settings()`, and `testBatchStatistic()`.

8.4.5.5 `TypeOfBatching batching::m_typeOfBatchStatistic` [private]

Definition at line 32 of file `batching.h`.

Referenced by `calculateBatchStatistic()`, `collapseBatchStatistic()`, `printSetting()`, `settings()`, and `updateBatchStatistic()`.

8.4.5.6 `INDEX batching::m_sizeBatch` [private]

Definition at line 33 of file `batching.h`.

Referenced by `calculateBatchStatistic()`, `getBatchSize()`, `printResult()`, `printStatus()`, `process()`, and `testBatchStatistic()`.

8.4.5.7 `INDEX batching::m_posBatch` [private]

Definition at line 34 of file `batching.h`.

Referenced by `printResult()`, `printStatus()`, and `process()`.

8.4.5.8 `INDEX batching::m_maxBatch` [private]

Definition at line 35 of file `batching.h`.

Referenced by `batching()`, `collapseBatchStatistic()`, `printResult()`, `printSetting()`, `printStatus()`, `process()`, `settings()`, and `testBatchStatistic()`.

8.4.5.9 INDEX batching::m_numBatch [private]

Definition at line 36 of file batching.h.

Referenced by calculateBatchStatistic(), printResult(), printStatus(), and process().

8.4.5.10 INDEX batching::m_p [private]

Definition at line 37 of file batching.h.

Referenced by batching(), calculateBatchStatistic(), collapseBatchStatistic(), printResult(), printStatus(), process(), settings(), testBatchStatistic(), and updateBatchStatistic().

8.4.5.11 std::list<CONTINUOUS> batching::ecdf [private]

Definition at line 38 of file batching.h.

Referenced by calculateBatchStatistic(), process(), and updateBatchStatistic().

8.4.5.12 std::vector<CONTINUOUS> batching::accuBatch [private]

Definition at line 39 of file batching.h.

Referenced by batching(), calculateBatchStatistic(), and updateBatchStatistic().

8.4.5.13 std::vector< std::vector<CONTINUOUS> > batching::batchStatistic [private]

Definition at line 40 of file batching.h.

Referenced by batching(), calculateBatchStatistic(), collapseBatchStatistic(), and testBatchStatistic().

8.4.5.14 INDEX outputAnalyser::m_processedIndexes [protected, inherited]

Definition at line 20 of file basic.h.

Referenced by evolution::calculateQuantiles(), spectral_analysis_QE::checkQuantiles(), batch_mean_QE::checkQuantiles(), pooling_QE::checkQuantiles(), deterministic_TPD::isReady(), evolution::isReady(), sequential_TPD::printResult(), deterministic_TPD::printResult(), spectral_analysis_QE::printResult(), batch_mean_QE::printResult(), pooling_QE::printResult(), printResult(), sequential_TPD::printStatus(), deterministic_TPD::printStatus(), evolution::printStatus(), spectral_analysis_QE::printStatus(), batch_mean_QE::printStatus(), pooling_QE::printStatus(), printStatus(), sequential_TPD::process(), deterministic_TPD::process(), evolution::process(), spectral_analysis_QE::process(), batch_mean_QE::process(), pooling_QE::process(), process(), outputAnalyser::process(), sequential_TPD::sub_collect(), sequential_TPD::sub_compare(), and sequential_TPD::sub_initialize().

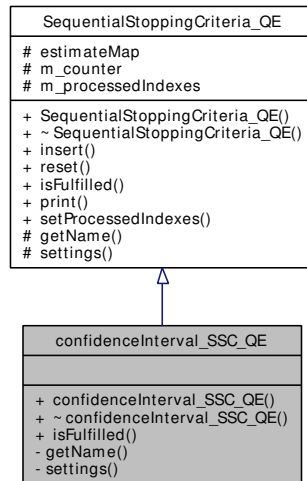
The documentation for this class was generated from the following files:

- **batching.h**
- **batching.cc**

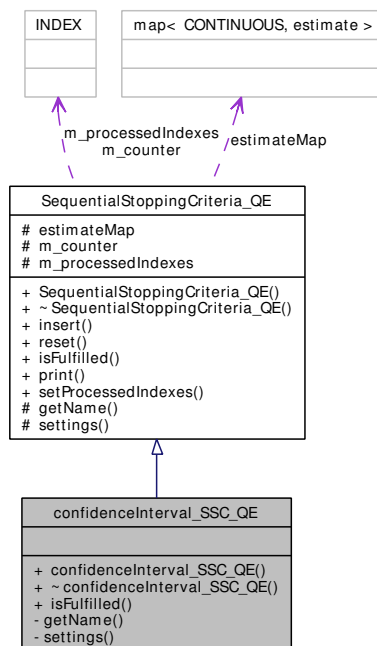
8.5 confidenceInterval_SSC_QE Class Reference

```
#include <quantile_estimation.h>
```

Inheritance diagram for confidenceInterval_SSC_QE:



Collaboration diagram for confidenceInterval_SSC_QE:



Public Member Functions

- `confidenceInterval_SSC_QE` (void)
- `~confidenceInterval_SSC_QE` (void)
- `bool isFulfilled` (void)

- void **insert** (const CONTINUOUS &location, const CONTINUOUS &probability, const CONTINUOUS &absoluteErrorNeg, const CONTINUOUS &absoluteErrorPos)
- void **reset** (void)
- void **print** (bool isFinal=false)
- void **setProcessedIndexes** (INDEX i)

Protected Attributes

- std::map< CONTINUOUS, estimate > **estimateMap**
- INDEX **m_counter**
- INDEX **m_processedIndexes**

Private Member Functions

- std::string **getName** (void)
- void **settings** (void)

8.5.1 Detailed Description

Definition at line 151 of file quantile_estimation.h.

8.5.2 Constructor & Destructor Documentation

8.5.2.1 confidenceInterval_SSC_QE::confidenceInterval_SSC_QE (void)

Definition at line 798 of file quantile_estimation.cc.

8.5.2.2 confidenceInterval_SSC_QE::~~confidenceInterval_SSC_QE (void)

Definition at line 803 of file quantile_estimation.cc.

8.5.3 Member Function Documentation

8.5.3.1 bool confidenceInterval_SSC_QE::isFulfilled (void) [virtual]

Reimplemented from **SequentialStoppingCriteria_QE** (p.138).

Definition at line 806 of file quantile_estimation.cc.

8.5.3.2 std::string confidenceInterval_SSC_QE::getName (void) [inline, private, virtual]

Reimplemented from **SequentialStoppingCriteria_QE** (p.138).

Definition at line 159 of file quantile_estimation.h.

References `s_confidenceInterval_SSC_QE`.

8.5.3.3 void confidenceInterval_SSC_QE::settings (void) [private, virtual]

Reimplemented from `SequentialStoppingCriteria_QE` (p. 139).

Definition at line 828 of file `quantile_estimation.cc`.

8.5.3.4 void SequentialStoppingCriteria_QE::insert (const CONTINUOUS & location, const CONTINUOUS & probability, const CONTINUOUS & absoluteErrorNeg, const CONTINUOUS & absoluteErrorPos) [inherited]

Definition at line 627 of file `quantile_estimation.cc`.

References `SequentialStoppingCriteria_QE::estimate::absoluteErrorNeg`, `SequentialStoppingCriteria_QE::estimate::absoluteErrorPos`, `SequentialStoppingCriteria_QE::estimateMap`, `SequentialStoppingCriteria_QE::estimate::location`, and `SequentialStoppingCriteria_QE::estimate::probability`.

Referenced by `spectral_analysis_QE::checkQuantiles()`, `batch_mean_QE::checkQuantiles()`, and `pooling_QE::checkQuantiles()`.

8.5.3.5 void SequentialStoppingCriteria_QE::reset (void) [inherited]

Definition at line 639 of file `quantile_estimation.cc`.

References `SequentialStoppingCriteria_QE::estimateMap`.

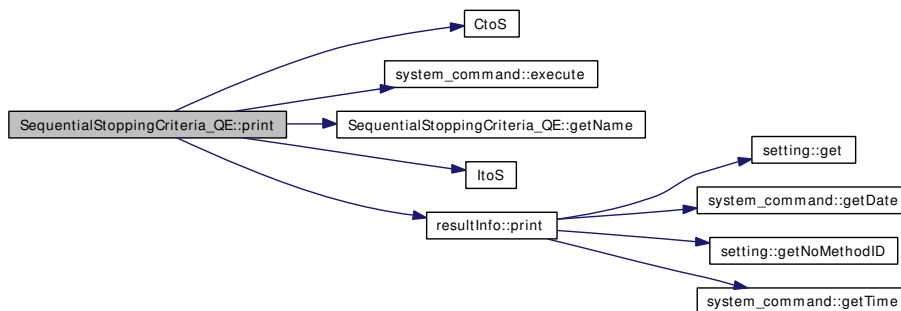
Referenced by `spectral_analysis_QE::checkQuantiles()`, `batch_mean_QE::checkQuantiles()`, and `pooling_QE::checkQuantiles()`.

8.5.3.6 void SequentialStoppingCriteria_QE::print (bool isFinal = false) [inherited]

Definition at line 652 of file `quantile_estimation.cc`.

References `CONTINUOUS`, `CtoS()`, `SequentialStoppingCriteria_QE::estimateMap`, `system_command::execute()`, `SequentialStoppingCriteria_QE::getName()`, `ItoS()`, `lib_system`, `SequentialStoppingCriteria_QE::m_counter`, `SequentialStoppingCriteria_QE::m_processedIndexes`, `resultInfo::print()`, and `resultfile`.

Here is the call graph for this function:



8.5.3.7 void SequentialStoppingCriteria_QE::setProcessedIndexes (INDEX *i*) [inline, inherited]

Definition at line 118 of file quantile_estimation.h.

References SequentialStoppingCriteria_QE::m_processedIndexes.

Referenced by spectral_analysis_QE::checkQuantiles(), batch_mean_QE::checkQuantiles(), and pooling_QE::checkQuantiles().

8.5.4 Field Documentation

8.5.4.1 std::map<CONTINUOUS,estimate> SequentialStoppingCriteria_QE::estimateMap [protected, inherited]

Definition at line 131 of file quantile_estimation.h.

Referenced by SequentialStoppingCriteria_QE::insert(), SequentialStoppingCriteria_QE::print(), and SequentialStoppingCriteria_QE::reset().

8.5.4.2 INDEX SequentialStoppingCriteria_QE::m_counter [protected, inherited]

Definition at line 132 of file quantile_estimation.h.

Referenced by SequentialStoppingCriteria_QE::print().

8.5.4.3 INDEX SequentialStoppingCriteria_QE::m_processedIndexes [protected, inherited]

Definition at line 133 of file quantile_estimation.h.

Referenced by SequentialStoppingCriteria_QE::print(), and SequentialStoppingCriteria_QE::setProcessedIndexes().

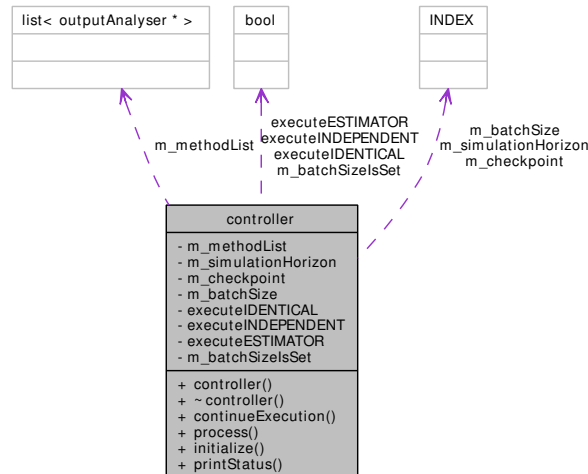
The documentation for this class was generated from the following files:

- **quantile_estimation.h**
- **quantile_estimation.cc**

8.6 controller Class Reference

```
#include <controller.h>
```

Collaboration diagram for controller:



Public Member Functions

- **controller** (void)
- **~controller** (void)
- **bool continueExecution** (void) const
- **void process** (const std::list< CONTINUOUS > &)
- **void initialize** (void)
- **void printStatus** (void) const

Private Attributes

- std::list< **outputAnalyser * >** **m_methodList**
- INDEX **m_simulationHorizon**
- INDEX **m_checkpoint**
- INDEX **m_batchSize**
- **bool executeIDENTICAL**
- **bool executeINDEPENDENT**
- **bool executeESTIMATOR**
- **bool m_batchSizeIsSet**

8.6.1 Detailed Description

Definition at line 9 of file controller.h.

8.6.2 Constructor & Destructor Documentation

8.6.2.1 controller::controller (void)

Definition at line 9 of file controller.cc.

References `m_methodList`.

8.6.2.2 controller::~~controller (void)

Definition at line 20 of file controller.cc.

References `m_methodList`.

8.6.3 Member Function Documentation

8.6.3.1 bool controller::continueExecution (void) const

Definition at line 28 of file controller.cc.

References `m_methodList`.

Referenced by `main()`.

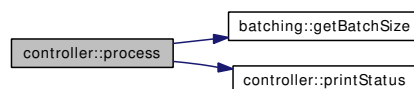
8.6.3.2 void controller::process (const std::list< CONTINUOUS > &)

Definition at line 32 of file controller.cc.

References `ESTIMATOR`, `EVOLUTION`, `executeESTIMATOR`, `executeIDENTICAL`, `executeINDEPENDENT`, `batching::getBatchSize()`, `IDENTICAL`, `INDEPENDENT`, `logfile`, `m_batchSize`, `m_batchSizeIsSet`, `m_checkpoint`, `m_methodList`, `m_simulationHorizon`, and `printStatus()`.

Referenced by `main()`.

Here is the call graph for this function:



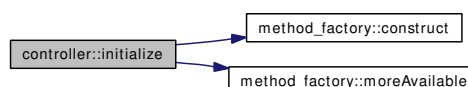
8.6.3.3 void controller::initialize (void)

Definition at line 141 of file controller.cc.

References `method_factory::construct()`, `m_methodList`, and `method_factory::moreAvailable()`.

Referenced by `main()`.

Here is the call graph for this function:



8.6.3.4 void controller::printStats (void) const

Definition at line 147 of file controller.cc.

References logfile, m_methodList, and m_simulationHorizon.

Referenced by main(), and process().

8.6.4 Field Documentation

8.6.4.1 std::list<outputAnalyser*> controller::m_methodList [private]

Definition at line 20 of file controller.h.

Referenced by continueExecution(), controller(), initialize(), printStatus(), process(), and ~controller().

8.6.4.2 INDEX controller::m_simulationHorizon [private]

Definition at line 21 of file controller.h.

Referenced by printStatus(), and process().

8.6.4.3 INDEX controller::m_checkpoint [private]

Definition at line 22 of file controller.h.

Referenced by process().

8.6.4.4 INDEX controller::m_batchSize [private]

Definition at line 23 of file controller.h.

Referenced by process().

8.6.4.5 bool controller::executeIDENTICAL [private]

Definition at line 25 of file controller.h.

Referenced by process().

8.6.4.6 bool controller::executeINDEPENDENT [private]

Definition at line 26 of file controller.h.

Referenced by process().

8.6.4.7 bool controller::executeESTIMATOR [private]

Definition at line 27 of file controller.h.

Referenced by process().

8.6.4.8 `bool controller::m_batchSizeIsSet` [private]

Definition at line 28 of file `controller.h`.

Referenced by `process()`.

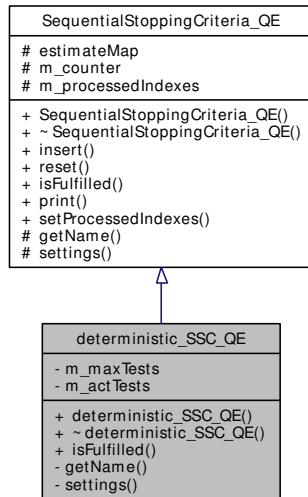
The documentation for this class was generated from the following files:

- `controller.h`
- `controller.cc`

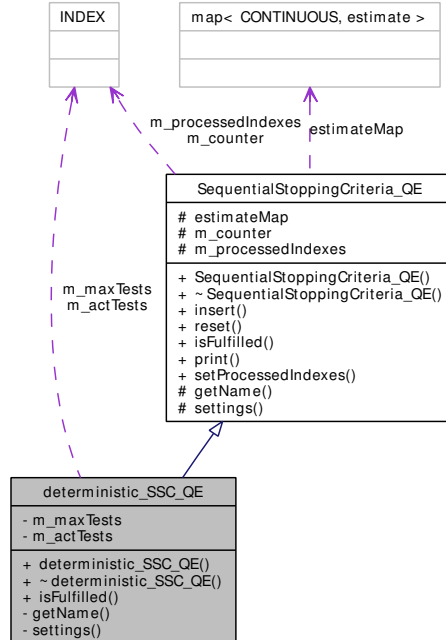
8.7 deterministic_SSC_QE Class Reference

```
#include <quantile_estimation.h>
```

Inheritance diagram for deterministic_SSC_QE:



Collaboration diagram for deterministic_SSC_QE:



Public Member Functions

- `deterministic_SSC_QE` (void)
- `~deterministic_SSC_QE` (void)

- bool **isFulfilled** (void)
- void **insert** (const CONTINUOUS &location, const CONTINUOUS &probability, const CONTINUOUS &absoluteErrorNeg, const CONTINUOUS &absoluteErrorPos)
- void **reset** (void)
- void **print** (bool isFinal=false)
- void **setProcessedIndexes** (INDEX i)

Protected Attributes

- std::map< CONTINUOUS, estimate > **estimateMap**
- INDEX **m_counter**
- INDEX **m_processedIndexes**

Private Member Functions

- std::string **getName** (void)
- void **settings** (void)

Private Attributes

- INDEX **m_maxTests**
- INDEX **m_actTests**

8.7.1 Detailed Description

Definition at line 136 of file `quantile_estimation.h`.

8.7.2 Constructor & Destructor Documentation

8.7.2.1 `deterministic_SSC_QE::deterministic_SSC_QE` (void)

Definition at line 769 of file `quantile_estimation.cc`.

8.7.2.2 `deterministic_SSC_QE::~~deterministic_SSC_QE` (void)

Definition at line 776 of file `quantile_estimation.cc`.

8.7.3 Member Function Documentation

8.7.3.1 `bool deterministic_SSC_QE::isFulfilled` (void) [virtual]

Reimplemented from `SequentialStoppingCriteria_QE` (p.138).

Definition at line 779 of file `quantile_estimation.cc`.

8.7.3.2 `std::string deterministic_SSC_QE::getName (void)` [inline, private, virtual]

Reimplemented from `SequentialStoppingCriteria_QE` (p.138).

Definition at line 144 of file `quantile_estimation.h`.

References `s_deterministic_SSC_QE`.

8.7.3.3 `void deterministic_SSC_QE::settings (void)` [private, virtual]

Reimplemented from `SequentialStoppingCriteria_QE` (p.139).

Definition at line 785 of file `quantile_estimation.cc`.

8.7.3.4 `void SequentialStoppingCriteria_QE::insert (const CONTINUOUS & location, const CONTINUOUS & probability, const CONTINUOUS & absoluteErrorNeg, const CONTINUOUS & absoluteErrorPos)` [inherited]

Definition at line 627 of file `quantile_estimation.cc`.

References `SequentialStoppingCriteria_QE::estimate::absoluteErrorNeg`, `SequentialStoppingCriteria_QE::estimate::absoluteErrorPos`, `SequentialStoppingCriteria_QE::estimateMap`, `SequentialStoppingCriteria_QE::estimate::location`, and `SequentialStoppingCriteria_QE::estimate::probability`.

Referenced by `spectral_analysis_QE::checkQuantiles()`, `batch_mean_QE::checkQuantiles()`, and `pooling_QE::checkQuantiles()`.

8.7.3.5 `void SequentialStoppingCriteria_QE::reset (void)` [inherited]

Definition at line 639 of file `quantile_estimation.cc`.

References `SequentialStoppingCriteria_QE::estimateMap`.

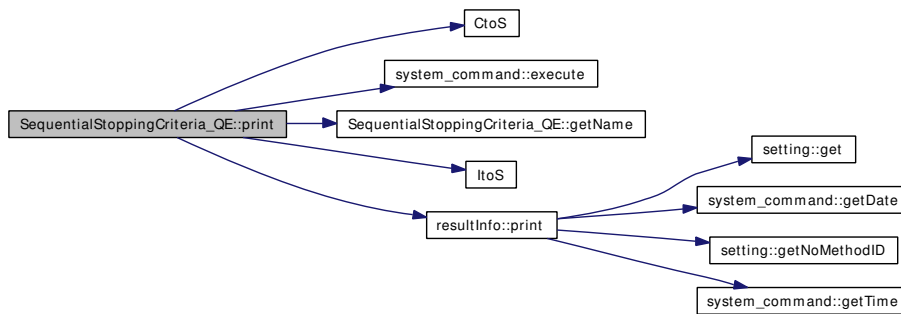
Referenced by `spectral_analysis_QE::checkQuantiles()`, `batch_mean_QE::checkQuantiles()`, and `pooling_QE::checkQuantiles()`.

8.7.3.6 `void SequentialStoppingCriteria_QE::print (bool isFinal = false)` [inherited]

Definition at line 652 of file `quantile_estimation.cc`.

References `CONTINUOUS`, `CtoS()`, `SequentialStoppingCriteria_QE::estimateMap`, `system_command::execute()`, `SequentialStoppingCriteria_QE::getName()`, `ltoS()`, `lib_system`, `SequentialStoppingCriteria_QE::m_counter`, `SequentialStoppingCriteria_QE::m_processedIndexes`, `resultInfo::print()`, and `resultfile`.

Here is the call graph for this function:



8.7.3.7 void SequentialStoppingCriteria_QE::setProcessedIndexes (INDEX *i*) [inline, inherited]

Definition at line 118 of file quantile_estimation.h.

References SequentialStoppingCriteria_QE::m_processedIndexes.

Referenced by spectral_analysis_QE::checkQuantiles(), batch_mean_QE::checkQuantiles(), and pooling_QE::checkQuantiles().

8.7.4 Field Documentation

8.7.4.1 INDEX deterministic_SSC_QE::m_maxTests [private]

Definition at line 147 of file quantile_estimation.h.

8.7.4.2 INDEX deterministic_SSC_QE::m_actTests [private]

Definition at line 148 of file quantile_estimation.h.

8.7.4.3 std::map<CONTINUOUS,estimate> SequentialStoppingCriteria_QE::estimateMap [protected, inherited]

Definition at line 131 of file quantile_estimation.h.

Referenced by SequentialStoppingCriteria_QE::insert(), SequentialStoppingCriteria_QE::print(), and SequentialStoppingCriteria_QE::reset().

8.7.4.4 INDEX SequentialStoppingCriteria_QE::m_counter [protected, inherited]

Definition at line 132 of file quantile_estimation.h.

Referenced by SequentialStoppingCriteria_QE::print().

8.7.4.5 INDEX SequentialStoppingCriteria_QE::m_processedIndexes [protected, inherited]

Definition at line 133 of file quantile_estimation.h.

Referenced by `SequentialStoppingCriteria_QE::print()`, and `SequentialStoppingCriteria_QE::set-ProcessedIndexes()`.

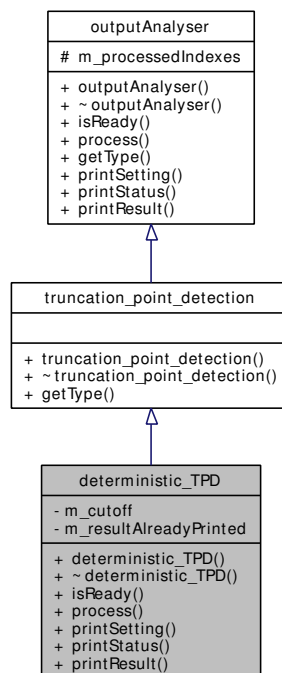
The documentation for this class was generated from the following files:

- `quantile_estimation.h`
- `quantile_estimation.cc`

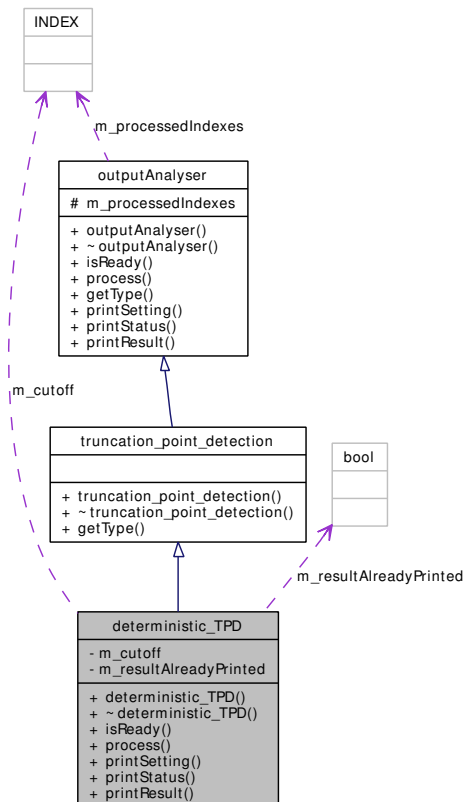
8.8 deterministic_TPD Class Reference

```
#include <truncation_point_detection.h>
```

Inheritance diagram for deterministic_TPD:



Collaboration diagram for deterministic_TPD:



Public Member Functions

- `deterministic_TPD` (void)
- `~deterministic_TPD` (void)
- `bool isReady` (void) const
- `void process` (const std::list< CONTINUOUS > &)
- `void printSetting` (void)
- `void printStatus` (void)
- `void printResult` (void)
- virtual `TypeOfMethod getType` (void) const

Protected Attributes

- `INDEX m_processedIndexes`

Private Attributes

- `INDEX m_cutoff`
- `bool m_resultAlreadyPrinted`

8.8.1 Detailed Description

Definition at line 14 of file `truncation_point_detection.h`.

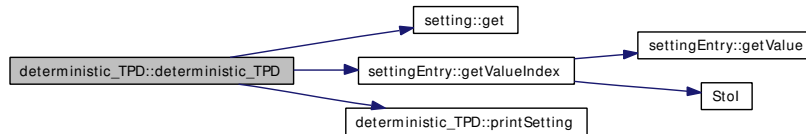
8.8.2 Constructor & Destructor Documentation

8.8.2.1 deterministic_TPD::deterministic_TPD (void)

Definition at line 22 of file truncation_point_detection.cc.

References `setting::get()`, `settingEntry::getValueIndex()`, `lib_setting`, `m_cutoff`, `printSetting()`, `s_cutoff`, and `s_deterministic_TPD`.

Here is the call graph for this function:



8.8.2.2 deterministic_TPD::~~deterministic_TPD (void)

Definition at line 34 of file truncation_point_detection.cc.

8.8.3 Member Function Documentation

8.8.3.1 bool deterministic_TPD::isReady (void) const [virtual]

Reimplemented from `outputAnalyser` (p. 183).

Definition at line 38 of file truncation_point_detection.cc.

References `m_cutoff`, and `outputAnalyser::m_processedIndexes`.

Referenced by `process()`.

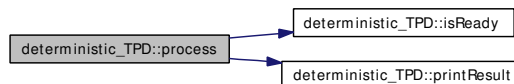
8.8.3.2 void deterministic_TPD::process (const std::list< CONTINUOUS > & [virtual]

Reimplemented from `outputAnalyser` (p. 183).

Definition at line 43 of file truncation_point_detection.cc.

References `isReady()`, `outputAnalyser::m_processedIndexes`, and `printResult()`.

Here is the call graph for this function:



8.8.3.3 void deterministic_TPD::printSetting (void) [virtual]

Reimplemented from `outputAnalyser` (p. 183).

Definition at line 48 of file truncation_point_detection.cc.

References logfile, m_cutoff, s_cutoff, s_deterministic_TPD, s_execute, and s_yes.

Referenced by deterministic_TPD().

8.8.3.4 void deterministic_TPD::printStatus (void) [virtual]

Reimplemented from **outputAnalyser** (p. 183).

Definition at line 59 of file truncation_point_detection.cc.

References logfile, m_cutoff, outputAnalyser::m_processedIndexes, and s_deterministic_TPD.

8.8.3.5 void deterministic_TPD::printResult (void) [virtual]

Reimplemented from **outputAnalyser** (p. 183).

Definition at line 66 of file truncation_point_detection.cc.

References logfile, m_cutoff, outputAnalyser::m_processedIndexes, m_resultAlreadyPrinted, and s_deterministic_TPD.

Referenced by process().

8.8.3.6 TypeOfMethod truncation_point_detection::getType (void) const [virtual, inherited]

Reimplemented from **outputAnalyser** (p. 87).

Definition at line 14 of file truncation_point_detection.cc.

References IDENTICAL.

8.8.4 Field Documentation

8.8.4.1 INDEX deterministic_TPD::m_cutoff [private]

Definition at line 25 of file truncation_point_detection.h.

Referenced by deterministic_TPD(), isReady(), printResult(), printSetting(), and printStatus().

8.8.4.2 bool deterministic_TPD::m_resultAlreadyPrinted [private]

Definition at line 26 of file truncation_point_detection.h.

Referenced by printResult().

8.8.4.3 INDEX outputAnalyser::m_processedIndexes [protected, inherited]

Definition at line 20 of file basic.h.

Referenced by evolution::calculateQuantiles(), spectral_analysis_QE::checkQuantiles(), batch_mean_QE::checkQuantiles(), pooling_QE::checkQuantiles(), isReady(), evolution::isReady(),

`sequential_TPD::printResult()`, `printResult()`, `spectral_analysis_QE::printResult()`, `batch_mean_QE::printResult()`, `pooling_QE::printResult()`, `batching::printResult()`, `sequential_TPD::printStatus()`, `printStatus()`, `evolution::printStatus()`, `spectral_analysis_QE::printStatus()`, `batch_mean_QE::printStatus()`, `pooling_QE::printStatus()`, `batching::printStatus()`, `sequential_TPD::process()`, `process()`, `evolution::process()`, `spectral_analysis_QE::process()`, `batch_mean_QE::process()`, `pooling_QE::process()`, `batching::process()`, `outputAnalyser::process()`, `sequential_TPD::sub_collect()`, `sequential_TPD::sub_compare()`, and `sequential_TPD::sub_initialize()`.

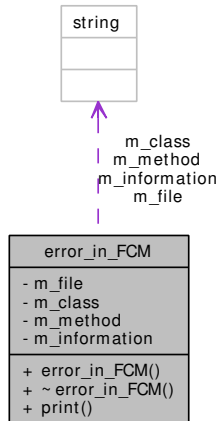
The documentation for this class was generated from the following files:

- `truncation_point_detection.h`
- `truncation_point_detection.cc`

8.9 error_in_FCM Class Reference

```
#include <error.h>
```

Collaboration diagram for error_in_FCM:



Public Member Functions

- `error_in_FCM` (`const std::string &f`, `const std::string &c`, `const std::string &m`, `const std::string &i`)
- `~error_in_FCM` ()
- `const std::string print` (void)

Private Attributes

- `const std::string m_file`
- `const std::string m_class`
- `const std::string m_method`
- `const std::string m_information`

8.9.1 Detailed Description

Definition at line 8 of file `error.h`.

8.9.2 Constructor & Destructor Documentation

8.9.2.1 `error_in_FCM::error_in_FCM` (`const std::string & f`, `const std::string & c`, `const std::string & m`, `const std::string & i`)

Definition at line 4 of file `error.cc`.

8.9.2.2 `error_in_FCM::~~error_in_FCM` ()

Definition at line 14 of file `error.cc`.

8.9.3 Member Function Documentation

8.9.3.1 `const std::string error_in_FCM::print (void)`

Definition at line 17 of file error.cc.

References `m_class`, `m_file`, `m_information`, and `m_method`.

Referenced by `main()`.

8.9.4 Field Documentation

8.9.4.1 `const std::string error_in_FCM::m_file [private]`

Definition at line 16 of file error.h.

Referenced by `print()`.

8.9.4.2 `const std::string error_in_FCM::m_class [private]`

Definition at line 17 of file error.h.

Referenced by `print()`.

8.9.4.3 `const std::string error_in_FCM::m_method [private]`

Definition at line 18 of file error.h.

Referenced by `print()`.

8.9.4.4 `const std::string error_in_FCM::m_information [private]`

Definition at line 19 of file error.h.

Referenced by `print()`.

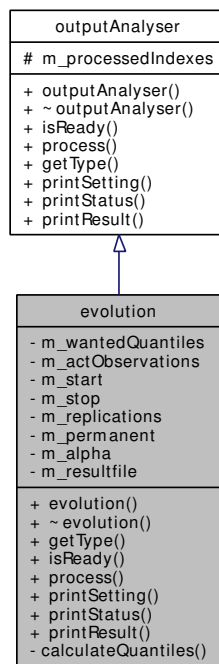
The documentation for this class was generated from the following files:

- `error.h`
- `error.cc`

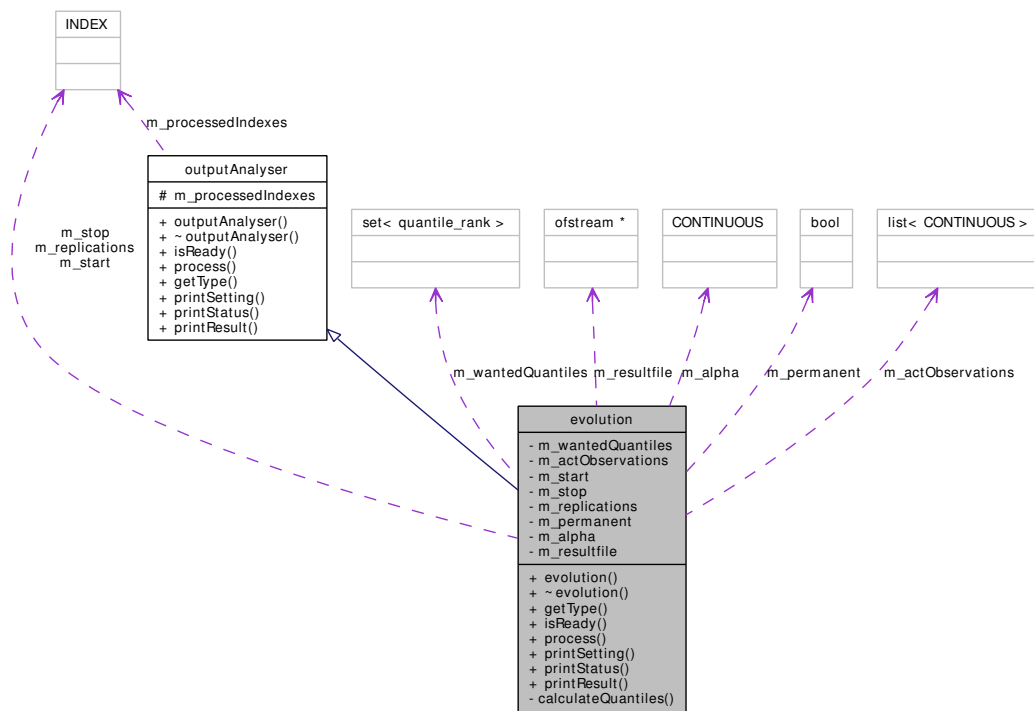
8.10 evolution Class Reference

```
#include <time_evolution.h>
```

Inheritance diagram for evolution:



Collaboration diagram for evolution:



Public Member Functions

- **evolution** (void)
- **~evolution** (void)
- **TypeOfMethod** **getType** (void) const
- **bool** **isReady** (void) const
- **void** **process** (const std::list< CONTINUOUS > &)
- **void** **printSetting** (void)
- **void** **printStatus** (void)
- **void** **printResult** (void)

Protected Attributes

- INDEX **m_processedIndexes**

Private Member Functions

- **void** **calculateQuantiles** (void)

Private Attributes

- std::set< **quantile_rank** > **m_wantedQuantiles**
- std::list< CONTINUOUS > **m_actObservations**
- INDEX **m_start**
- INDEX **m_stop**
- INDEX **m_replications**

- bool `m_permanent`
- CONTINUOUS `m_alpha`
- `std::ofstream * m_resultfile`

8.10.1 Detailed Description

Definition at line 7 of file `time_evolution.h`.

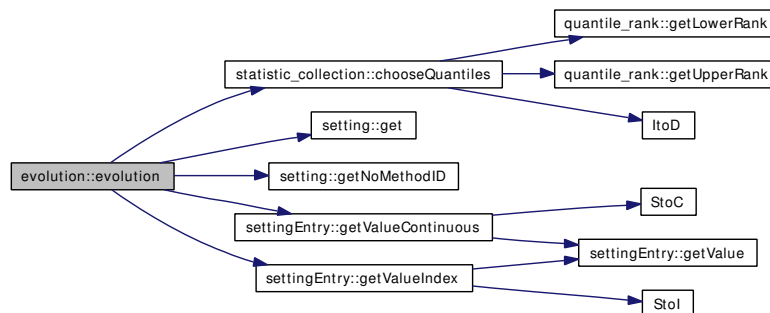
8.10.2 Constructor & Destructor Documentation

8.10.2.1 `evolution::evolution (void)`

Definition at line 4 of file `time_evolution.cc`.

References `statistic_collection::chooseQuantiles()`, `setting::get()`, `setting::getNoMethodID()`, `settingEntry::getValueContinuous()`, `settingEntry::getValueIndex()`, `lib_setting`, `lib_statistic`, `m_alpha`, `m_permanent`, `m_replications`, `m_resultfile`, `m_start`, `m_stop`, `m_wantedQuantiles`, `s_alpha`, `s_evolution`, `s_replications`, `s_start`, and `s_stop`.

Here is the call graph for this function:

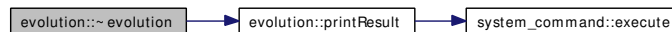


8.10.2.2 `evolution::~~evolution (void)`

Definition at line 96 of file `time_evolution.cc`.

References `m_permanent`, and `printResult()`.

Here is the call graph for this function:



8.10.3 Member Function Documentation

8.10.3.1 `evolution::getType (void) const [virtual]`

Reimplemented from `outputAnalyser` (p. 87).

Definition at line 120 of file `time_evolution.cc`.

References `EVOLUTION`.

8.10.3.2 bool evolution::isReady (void) const [virtual]

Reimplemented from **outputAnalyser** (p. 183).

Definition at line 100 of file time_evolution.cc.

References m_permanent, outputAnalyser::m_processedIndexes, and m_stop.

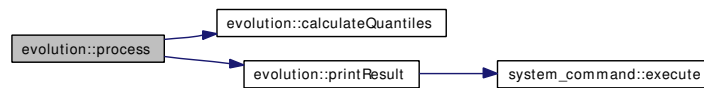
8.10.3.3 void evolution::process (const std::list< CONTINUOUS > &) [virtual]

Reimplemented from **outputAnalyser** (p. 183).

Definition at line 106 of file time_evolution.cc.

References calculateQuantiles(), m_actObservations, m_permanent, outputAnalyser::m_processedIndexes, m_replications, m_start, m_stop, m_wantedQuantiles, and printResult().

Here is the call graph for this function:

**8.10.3.4 void evolution::printSetting (void) [virtual]**

Reimplemented from **outputAnalyser** (p. 183).

Definition at line 124 of file time_evolution.cc.

References logfile, m_permanent, m_start, m_stop, m_wantedQuantiles, s_evolution, s_execute, s_no, s_permanent, s_start, s_stop, and s_yes.

8.10.3.5 void evolution::printStatus (void) [virtual]

Reimplemented from **outputAnalyser** (p. 183).

Definition at line 143 of file time_evolution.cc.

References logfile, m_permanent, outputAnalyser::m_processedIndexes, m_start, m_stop, s_evolution, s_no, s_permanent, s_start, s_stop, and s_yes.

8.10.3.6 void evolution::printResult (void) [virtual]

Reimplemented from **outputAnalyser** (p. 183).

Definition at line 155 of file time_evolution.cc.

References system_command::execute(), lib_system, m_resultfile, and s_evolution.

Referenced by process(), and ~evolution().

Here is the call graph for this function:



8.10.3.7 void evolution::calculateQuantiles (void) [private]

Definition at line 167 of file time_evolution.cc.

References INDEX, m_actObservations, outputAnalyser::m_processedIndexes, m_resultfile, and m_wantedQuantiles.

Referenced by process().

8.10.4 Field Documentation**8.10.4.1 std::set<quantile_rank> evolution::m_wantedQuantiles [private]**

Definition at line 22 of file time_evolution.h.

Referenced by calculateQuantiles(), evolution(), printSetting(), and process().

8.10.4.2 std::list<CONTINUOUS> evolution::m_actObservations [private]

Definition at line 23 of file time_evolution.h.

Referenced by calculateQuantiles(), and process().

8.10.4.3 INDEX evolution::m_start [private]

Definition at line 24 of file time_evolution.h.

Referenced by evolution(), printSetting(), printStatus(), and process().

8.10.4.4 INDEX evolution::m_stop [private]

Definition at line 25 of file time_evolution.h.

Referenced by evolution(), isReady(), printSetting(), printStatus(), and process().

8.10.4.5 INDEX evolution::m_replications [private]

Definition at line 26 of file time_evolution.h.

Referenced by evolution(), and process().

8.10.4.6 bool evolution::m_permanent [private]

Definition at line 27 of file time_evolution.h.

Referenced by evolution(), isReady(), printSetting(), printStatus(), process(), and ~evolution().

8.10.4.7 CONTINUOUS evolution::m_alpha [private]

Definition at line 28 of file time_evolution.h.

Referenced by evolution().

8.10.4.8 `std::ofstream*` `evolution::m_resultfile` [private]

Definition at line 29 of file `time_evolution.h`.

Referenced by `calculateQuantiles()`, `evolution()`, and `printResult()`.

8.10.4.9 `INDEX` `outputAnalyser::m_processedIndexes` [protected, inherited]

Definition at line 20 of file `basic.h`.

Referenced by `calculateQuantiles()`, `spectral_analysis_QE::checkQuantiles()`, `batch_mean_QE::checkQuantiles()`, `pooling_QE::checkQuantiles()`, `deterministic_TPD::isReady()`, `isReady()`, `sequential_TPD::printResult()`, `deterministic_TPD::printResult()`, `spectral_analysis_QE::printResult()`, `batch_mean_QE::printResult()`, `pooling_QE::printResult()`, `batching::printResult()`, `sequential_TPD::printStatus()`, `deterministic_TPD::printStatus()`, `printStatus()`, `spectral_analysis_QE::printStatus()`, `batch_mean_QE::printStatus()`, `pooling_QE::printStatus()`, `batching::printStatus()`, `sequential_TPD::process()`, `deterministic_TPD::process()`, `process()`, `spectral_analysis_QE::process()`, `batch_mean_QE::process()`, `pooling_QE::process()`, `batching::process()`, `outputAnalyser::process()`, `sequential_TPD::sub_collect()`, `sequential_TPD::sub_compare()`, and `sequential_TPD::sub_initialize()`.

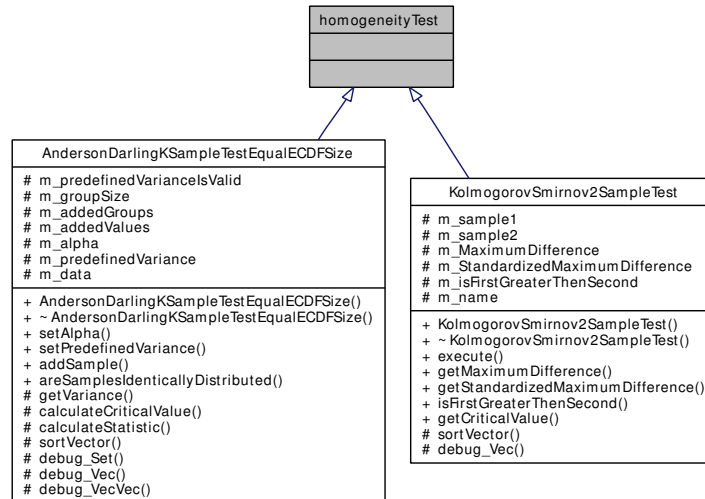
The documentation for this class was generated from the following files:

- `time_evolution.h`
- `time_evolution.cc`

8.11 homogeneityTest Class Reference

```
#include <homogeneityTests.h>
```

Inheritance diagram for homogeneityTest:



8.11.1 Detailed Description

Definition at line 9 of file `homogeneityTests.h`.

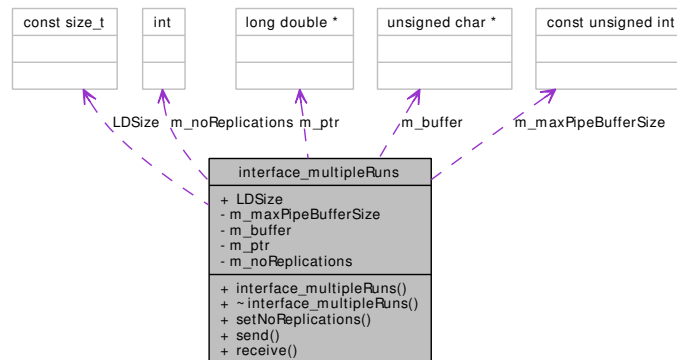
The documentation for this class was generated from the following file:

- `homogeneityTests.h`

8.12 interface _multipleRuns Class Reference

```
#include <interface.h>
```

Collaboration diagram for interface _multipleRuns:



Public Member Functions

- **interface _multipleRuns** (void)
- **~interface _multipleRuns** (void)
- **void setNoReplications** (int replications)
- **bool send** (const std::list< long double > &, int=STDOUT_FILENO)
- **bool receive** (std::list< long double > &, int=STDIN_FILENO)

Data Fields

- const size_t **LDSize**

Private Attributes

- const unsigned int **m_maxPipeBufferSize**
- unsigned char * **m_buffer**
- long double * **m_ptr**
- int **m_noReplications**

8.12.1 Detailed Description

Definition at line 29 of file interface.h.

8.12.2 Constructor & Destructor Documentation

8.12.2.1 interface _multipleRuns::interface _multipleRuns (void)

Definition at line 67 of file interface.cc.

References `m_ptr`.

8.12.2.2 `interface _multipleRuns::~~interface _multipleRuns (void)`

Definition at line 74 of file `interface.cc`.

References `m_buffer`, and `m_ptr`.

8.12.3 Member Function Documentation

8.12.3.1 `void interface _multipleRuns::setNoReplications (int replications)`

Definition at line 80 of file `interface.cc`.

References `LDSize`, `m_buffer`, and `m_noReplications`.

Referenced by `main()`.

8.12.3.2 `bool interface _multipleRuns::send (const std::list< long double > &, int = STDOUT_FILENO)`

Definition at line 86 of file `interface.cc`.

References `LDSize`, `m_buffer`, `m_maxPipeBufferSize`, `m_noReplications`, and `m_ptr`.

8.12.3.3 `bool interface _multipleRuns::receive (std::list< long double > &, int = STDIN_FILENO)`

Definition at line 121 of file `interface.cc`.

References `LDSize`, `m_buffer`, `m_maxPipeBufferSize`, `m_noReplications`, and `m_ptr`.

Referenced by `main()`.

8.12.4 Field Documentation

8.12.4.1 `const size_t interface _multipleRuns::LDSize`

Definition at line 35 of file `interface.h`.

Referenced by `receive()`, `send()`, and `setNoReplications()`.

8.12.4.2 `const unsigned int interface _multipleRuns::m_maxPipeBufferSize [private]`

Definition at line 41 of file `interface.h`.

Referenced by `receive()`, and `send()`.

8.12.4.3 `unsigned char* interface _multipleRuns::m_buffer [private]`

Definition at line 42 of file `interface.h`.

Referenced by `receive()`, `send()`, `setNoReplications()`, and `~interface _multipleRuns()`.

8.12.4.4 `long double* interface_multipleRuns::m_ptr` [private]

Definition at line 43 of file `interface.h`.

Referenced by `interface_multipleRuns()`, `receive()`, `send()`, and `~interface_multipleRuns()`.

8.12.4.5 `int interface_multipleRuns::m_noReplications` [private]

Definition at line 44 of file `interface.h`.

Referenced by `receive()`, `send()`, and `setNoReplications()`.

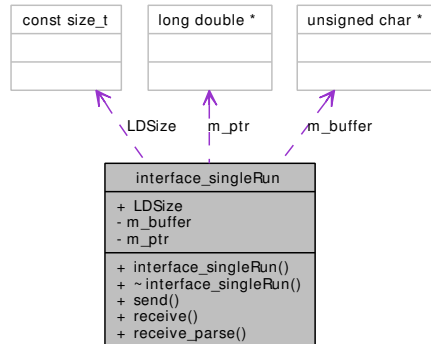
The documentation for this class was generated from the following files:

- `interface.h`
- `interface.cc`

8.13 interface_singleRun Class Reference

```
#include <interface.h>
```

Collaboration diagram for interface_singleRun:



Public Member Functions

- **interface_singleRun** (void)
- **~interface_singleRun** (void)
- **bool send** (const long double &, int=STDOUT_FILENO)
- **bool receive** (long double &, int=STDIN_FILENO)
- **bool receive_parse** (long double &, int=STDIN_FILENO)

Data Fields

- const size_t **LDSize**

Private Attributes

- unsigned char * **m_buffer**
- long double * **m_ptr**

8.13.1 Detailed Description

Definition at line 13 of file interface.h.

8.13.2 Constructor & Destructor Documentation

8.13.2.1 interface_singleRun::interface_singleRun (void)

Definition at line 13 of file interface.cc.

References LDSize, m_buffer, and m_ptr.

8.13.2.2 `interface _singleRun::~~interface _singleRun (void)`

Definition at line 19 of file `interface.cc`.

References `m_buffer`, and `m_ptr`.

8.13.3 Member Function Documentation

8.13.3.1 `bool interface _singleRun::send (const long double &, int = STDOUT_FILENO)`

Definition at line 25 of file `interface.cc`.

References `LDSize`, `m_buffer`, and `m_ptr`.

8.13.3.2 `bool interface _singleRun::receive (long double &, int = STDIN_FILENO)`

Definition at line 35 of file `interface.cc`.

References `LDSize`, `m_buffer`, and `m_ptr`.

8.13.3.3 `bool interface _singleRun::receive_parse (long double &, int = STDIN_FILENO)`

Definition at line 45 of file `interface.cc`.

8.13.4 Field Documentation

8.13.4.1 `const size_t interface _singleRun::LDSize`

Definition at line 18 of file `interface.h`.

Referenced by `interface _singleRun()`, `receive()`, and `send()`.

8.13.4.2 `unsigned char* interface _singleRun::m_buffer [private]`

Definition at line 25 of file `interface.h`.

Referenced by `interface _singleRun()`, `receive()`, `send()`, and `~interface _singleRun()`.

8.13.4.3 `long double* interface _singleRun::m_ptr [private]`

Definition at line 26 of file `interface.h`.

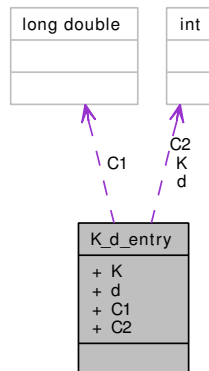
Referenced by `interface _singleRun()`, `receive()`, `send()`, and `~interface _singleRun()`.

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

- `interface.h`
- `interface.cc`

8.14 K_d_entry Struct Reference

Collaboration diagram for K_d_entry:



Data Fields

- int **K**
- int **d**
- long double **C1**
- int **C2**

8.14.1 Detailed Description

Definition at line 154 of file akaroa_import.cc.

8.14.2 Field Documentation

8.14.2.1 int K_d_entry::K

Definition at line 154 of file akaroa_import.cc.

Referenced by akaroa_import::LookUp_K_d().

8.14.2.2 int K_d_entry::d

Definition at line 154 of file akaroa_import.cc.

Referenced by akaroa_import::LookUp_K_d().

8.14.2.3 long double K_d_entry::C1

Definition at line 154 of file akaroa_import.cc.

Referenced by akaroa_import::LookUp_K_d().

8.14.2.4 int K_d_entry::C2

Definition at line 154 of file akaroa_import.cc.

Referenced by akaroa_import::LookUp_K_d().

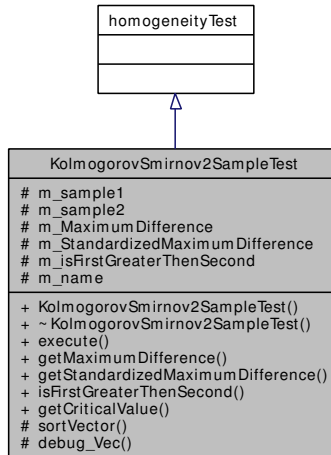
The documentation for this struct was generated from the following file:

- akaroa_import.cc

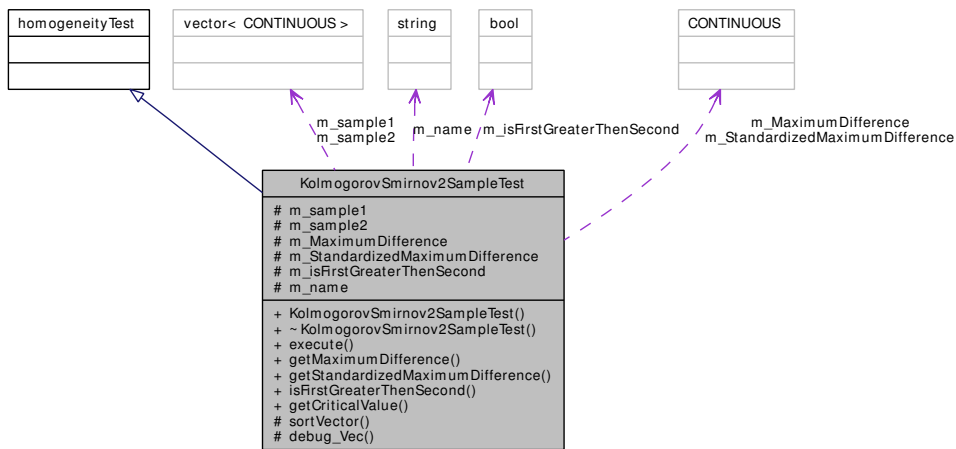
8.15 KolmogorovSmirnov2SampleTest Class Reference

```
#include <homogeneityTests.h>
```

Inheritance diagram for KolmogorovSmirnov2SampleTest:



Collaboration diagram for KolmogorovSmirnov2SampleTest:



Public Member Functions

- `KolmogorovSmirnov2SampleTest` (`const std::vector< CONTINUOUS > *`, `const std::vector< CONTINUOUS > *`, `std::string="_NoDiagram_"`)
- `~KolmogorovSmirnov2SampleTest` ()
- `bool execute` (void)
- `CONTINUOUS getMaximumDifference` (void) const
- `CONTINUOUS getStandardizedMaximumDifference` (void) const
- `bool isFirstGreaterThenSecond` (void) const
- `CONTINUOUS getCriticalValue` (void) const

Protected Member Functions

- void **sortVector** (std::vector< CONTINUOUS > &)
- void **debug_Vec** (std::vector< CONTINUOUS > &)

Protected Attributes

- std::vector< CONTINUOUS > **m_sample1**
- std::vector< CONTINUOUS > **m_sample2**
- CONTINUOUS **m_MaximumDifference**
- CONTINUOUS **m_StandardizedMaximumDifference**
- bool **m_isFirstGreaterThenSecond**
- std::string **m_name**

8.15.1 Detailed Description

Definition at line 54 of file homogeneityTests.h.

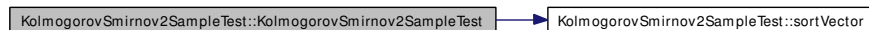
8.15.2 Constructor & Destructor Documentation

8.15.2.1 KolmogorovSmirnov2SampleTest::KolmogorovSmirnov2SampleTest (const std::vector< CONTINUOUS > *, const std::vector< CONTINUOUS > *, std::string = "_NoDiagram_")

Definition at line 265 of file homogeneityTests.cc.

References INDEX, m_sample1, m_sample2, and sortVector().

Here is the call graph for this function:



8.15.2.2 KolmogorovSmirnov2SampleTest::~~KolmogorovSmirnov2SampleTest ()

Definition at line 283 of file homogeneityTests.cc.

8.15.3 Member Function Documentation

8.15.3.1 bool KolmogorovSmirnov2SampleTest::execute (void)

Definition at line 286 of file homogeneityTests.cc.

8.15.3.2 CONTINUOUS KolmogorovSmirnov2SampleTest::getMaximumDifference (void) const

Definition at line 416 of file homogeneityTests.cc.

8.15.3.3 CONTINUOUS KolmogorovSmirnov2SampleTest::getStandardizedMaximumDifference (void) const

Definition at line 420 of file homogeneityTests.cc.

8.15.3.4 bool KolmogorovSmirnov2SampleTest::isFirstGreaterThenSecond (void) const

Definition at line 424 of file homogeneityTests.cc.

8.15.3.5 CONTINUOUS KolmogorovSmirnov2SampleTest::getCriticalValue (void) const

Definition at line 428 of file homogeneityTests.cc.

8.15.3.6 void KolmogorovSmirnov2SampleTest::sortVector (std::vector<CONTINUOUS > &) [protected]

Definition at line 448 of file homogeneityTests.cc.

Referenced by KolmogorovSmirnov2SampleTest().

8.15.3.7 void KolmogorovSmirnov2SampleTest::debug_Vec (std::vector<CONTINUOUS > &) [protected]

Definition at line 460 of file homogeneityTests.cc.

8.15.4 Field Documentation**8.15.4.1 std::vector<CONTINUOUS> KolmogorovSmirnov2SampleTest::m_sample1 [protected]**

Definition at line 68 of file homogeneityTests.h.

Referenced by KolmogorovSmirnov2SampleTest().

8.15.4.2 std::vector<CONTINUOUS> KolmogorovSmirnov2SampleTest::m_sample2 [protected]

Definition at line 69 of file homogeneityTests.h.

Referenced by KolmogorovSmirnov2SampleTest().

8.15.4.3 CONTINUOUS KolmogorovSmirnov2SampleTest::m_MaximumDifference [protected]

Definition at line 70 of file homogeneityTests.h.

8.15.4.4 CONTINUOUS KolmogorovSmirnov2SampleTest::m_StandardizedMaximumDifference [protected]

Definition at line 71 of file homogeneityTests.h.

8.15.4.5 bool KolmogorovSmirnov2SampleTest::m_isFirstGreaterThenSecond [protected]

Definition at line 72 of file homogeneityTests.h.

8.15.4.6 std::string KolmogorovSmirnov2SampleTest::m_name [protected]

Definition at line 73 of file homogeneityTests.h.

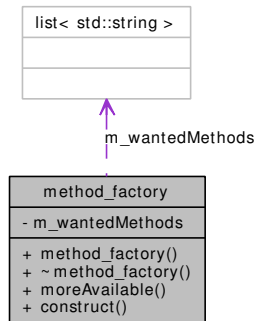
The documentation for this class was generated from the following files:

- **homogeneityTests.h**
- **homogeneityTests.cc**

8.16 method_factory Class Reference

```
#include <method_factory.h>
```

Collaboration diagram for method_factory:



Public Member Functions

- `method_factory` (void)
- `~method_factory` (void)
- `bool moreAvailable` (void)
- `outputAnalyser * construct` (void)

Private Attributes

- `std::list< std::string > m_wantedMethods`

8.16.1 Detailed Description

Definition at line 13 of file `method_factory.h`.

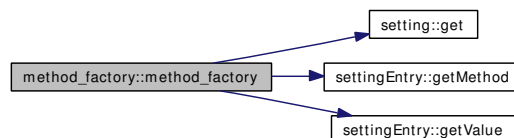
8.16.2 Constructor & Destructor Documentation

8.16.2.1 method_factory::method_factory (void)

Definition at line 4 of file `method_factory.cc`.

References `setting::get()`, `settingEntry::getMethod()`, `settingEntry::getValue()`, `INDEX`, `lib_setting`, `m_wantedMethods`, `s_batch_mean_QE`, `s_deterministic_TPD`, `s_evolution`, `s_execute`, `s_pooling_QE`, `s_sequential_batching`, `s_sequential_TPD`, `s_spectral_analysis_QE`, and `s_yes`.

Here is the call graph for this function:



8.16.2.2 method_factory::~~method_factory (void)

Definition at line 26 of file method_factory.cc.

8.16.3 Member Function Documentation

8.16.3.1 bool method_factory::moreAvailable (void)

Definition at line 29 of file method_factory.cc.

References m_wantedMethods.

Referenced by controller::initialize().

8.16.3.2 outputAnalyser * method_factory::construct (void)

Definition at line 33 of file method_factory.cc.

References m_wantedMethods, s_batch_mean_QE, s_deterministic_TPD, s_evolution, s_pooling_QE, s_sequential_batching, s_sequential_TPD, and s_spectral_analysis_QE.

Referenced by controller::initialize().

8.16.4 Field Documentation

8.16.4.1 std::list<std::string> method_factory::m_wantedMethods [private]

Definition at line 21 of file method_factory.h.

Referenced by construct(), method_factory(), and moreAvailable().

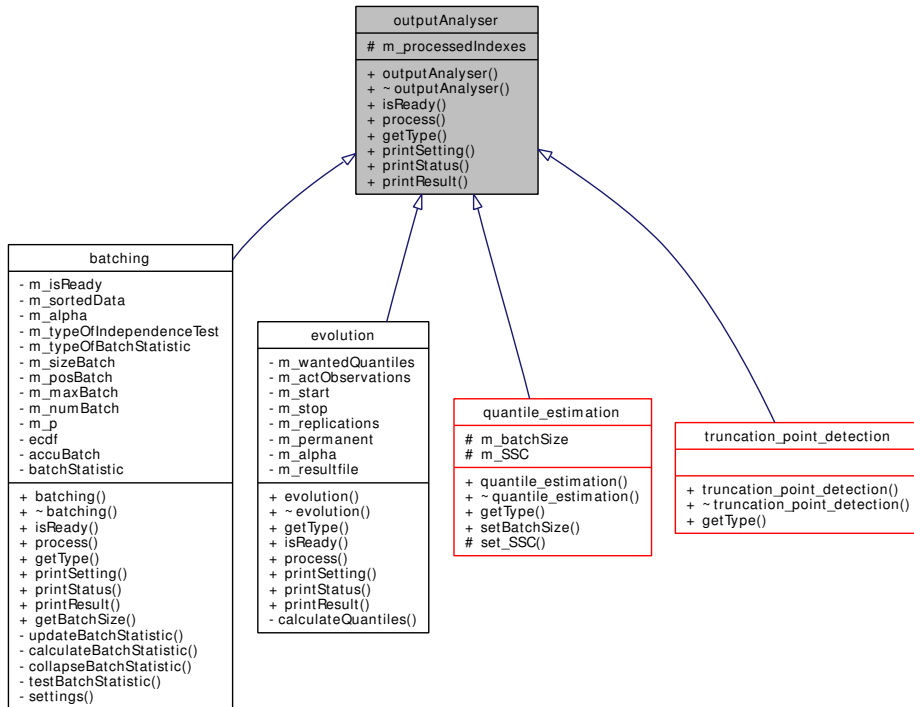
The documentation for this class was generated from the following files:

- **method_factory.h**
- **method_factory.cc**

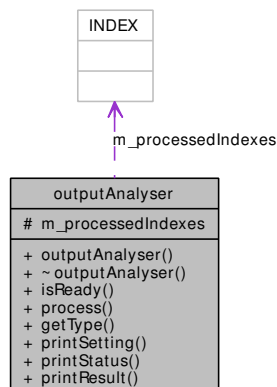
8.17 outputAnalyser Class Reference

```
#include <basic.h>
```

Inheritance diagram for outputAnalyser:



Collaboration diagram for outputAnalyser:



Public Member Functions

- `outputAnalyser` (void)
- virtual `~outputAnalyser` (void)
- virtual bool `isReady` (void) const
- virtual void `process` (const std::list< CONTINUOUS > &)

- virtual `TypeOfMethod` `getType` (void) const
- virtual void `printSetting` (void)
- virtual void `printStatus` (void)
- virtual void `printResult` (void)

Protected Attributes

- INDEX `m_processedIndexes`

8.17.1 Detailed Description

Definition at line 8 of file basic.h.

8.17.2 Constructor & Destructor Documentation

8.17.2.1 `outputAnalyser::outputAnalyser` (void)

Definition at line 4 of file basic.cc.

8.17.2.2 `outputAnalyser::~~outputAnalyser` (void) [virtual]

Definition at line 8 of file basic.cc.

8.17.3 Member Function Documentation

8.17.3.1 `bool outputAnalyser::isReady` (void) const [virtual]

Reimplemented in `batching` (p. 40), `pooling_QE` (p. 92), `batch_mean_QE` (p. 31), `spectral_analysis_QE` (p. 153), `evolution` (p. 69), `deterministic_TPD` (p. 61), and `sequential_TPD` (p. 128).

Definition at line 11 of file basic.cc.

8.17.3.2 `void outputAnalyser::process` (const std::list< CONTINUOUS > &) [virtual]

Reimplemented in `batching` (p. 40), `pooling_QE` (p. 92), `batch_mean_QE` (p. 32), `spectral_analysis_QE` (p. 153), `evolution` (p. 69), `deterministic_TPD` (p. 61), and `sequential_TPD` (p. 128).

Definition at line 15 of file basic.cc.

References `m_processedIndexes`.

8.17.3.3 `TypeOfMethod outputAnalyser::getType` (void) const [virtual]

Reimplemented in `batching` (p. 40), `quantile_estimation` (p. 156), `evolution` (p. 68), and `truncation_point_detection` (p. 183).

Definition at line 19 of file basic.cc.

References NON.

8.17.3.4 void outputAnalyser::printSetting (void) [virtual]

Reimplemented in **batching** (p. 41), **pooling_QE** (p. 92), **batch_mean_QE** (p. 32), **spectral_analysis_QE** (p. 154), **evolution** (p. 69), **deterministic_TPD** (p. 61), and **sequential_TPD** (p. 129).

Definition at line 23 of file basic.cc.

8.17.3.5 void outputAnalyser::printStatus (void) [virtual]

Reimplemented in **batching** (p. 41), **pooling_QE** (p. 92), **batch_mean_QE** (p. 32), **spectral_analysis_QE** (p. 154), **evolution** (p. 69), **deterministic_TPD** (p. 62), and **sequential_TPD** (p. 129).

Definition at line 26 of file basic.cc.

8.17.3.6 void outputAnalyser::printResult (void) [virtual]

Reimplemented in **batching** (p. 41), **pooling_QE** (p. 92), **batch_mean_QE** (p. 32), **spectral_analysis_QE** (p. 154), **evolution** (p. 69), **deterministic_TPD** (p. 62), and **sequential_TPD** (p. 129).

Definition at line 29 of file basic.cc.

8.17.4 Field Documentation**8.17.4.1 INDEX outputAnalyser::m_processedIndexes [protected]**

Definition at line 20 of file basic.h.

Referenced by `evolution::calculateQuantiles()`, `spectral_analysis_QE::checkQuantiles()`, `batch_mean_QE::checkQuantiles()`, `pooling_QE::checkQuantiles()`, `deterministic_TPD::isReady()`, `evolution::isReady()`, `sequential_TPD::printResult()`, `deterministic_TPD::printResult()`, `spectral_analysis_QE::printResult()`, `batch_mean_QE::printResult()`, `pooling_QE::printResult()`, `batching::printResult()`, `sequential_TPD::printStatus()`, `deterministic_TPD::printStatus()`, `evolution::printStatus()`, `spectral_analysis_QE::printStatus()`, `batch_mean_QE::printStatus()`, `pooling_QE::printStatus()`, `batching::printStatus()`, `sequential_TPD::process()`, `deterministic_TPD::process()`, `evolution::process()`, `spectral_analysis_QE::process()`, `batch_mean_QE::process()`, `pooling_QE::process()`, `batching::process()`, `process()`, `sequential_TPD::sub_collect()`, `sequential_TPD::sub_compare()`, and `sequential_TPD::sub_initialize()`.

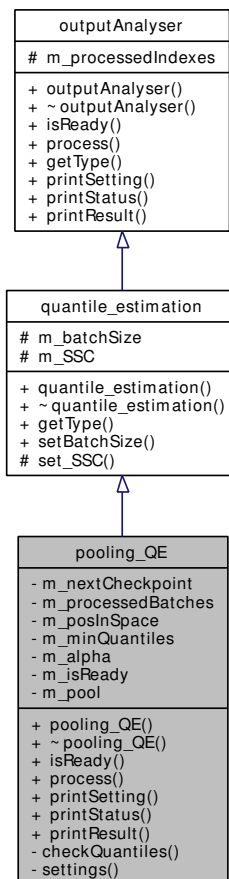
The documentation for this class was generated from the following files:

- **basic.h**
- **basic.cc**

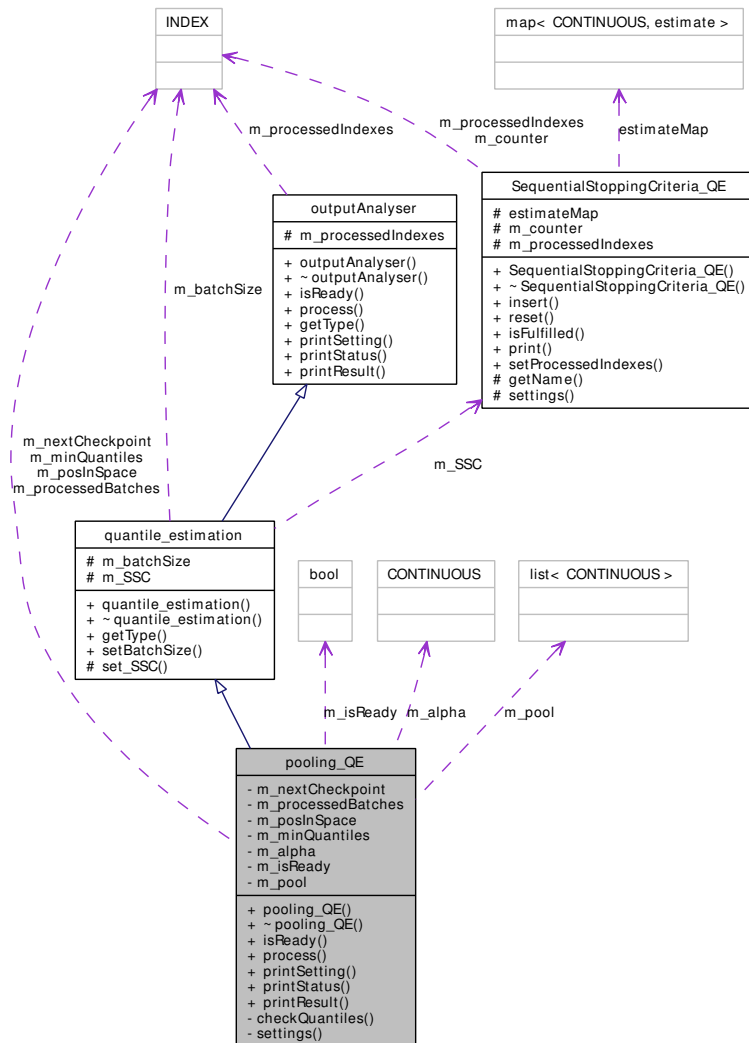
8.18 pooling_QE Class Reference

```
#include <quantile_estimation.h>
```

Inheritance diagram for pooling_QE:



Collaboration diagram for pooling_QE:



Public Member Functions

- **pooling_QE** (void)
- **~pooling_QE** (void)
- **bool isReady** (void) const
- void **process** (const std::list< CONTINUOUS > &)
- void **printSetting** (void)
- void **printStatus** (void)
- void **printResult** (void)
- virtual **TypeOfMethod getType** (void) const
- void **setBatchSize** (INDEX p)

Protected Member Functions

- void **set_SSC** (void)

Protected Attributes

- INDEX `m_batchSize`
- `SequentialStoppingCriteria_QE * m_SSC`
- INDEX `m_processedIndexes`

Private Member Functions

- bool `checkQuantiles` (void)
- void `settings` (void)

Private Attributes

- INDEX `m_nextCheckpoint`
- INDEX `m_processedBatches`
- INDEX `m_posInSpace`
- INDEX `m_minQuantiles`
- CONTINUOUS `m_alpha`
- bool `m_isReady`
- `std::list< CONTINUOUS > m_pool`

8.18.1 Detailed Description

Definition at line 30 of file `quantile_estimation.h`.

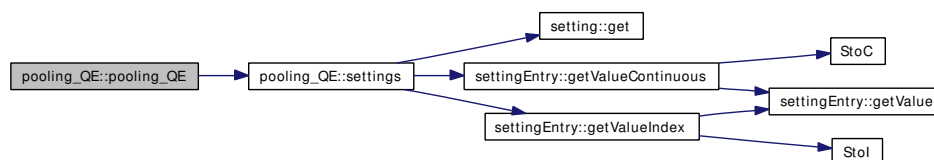
8.18.2 Constructor & Destructor Documentation

8.18.2.1 pooling_QE::pooling_QE (void)

Definition at line 74 of file `quantile_estimation.cc`.

References `settings()`.

Here is the call graph for this function:



8.18.2.2 pooling_QE::~~pooling_QE (void)

Definition at line 85 of file `quantile_estimation.cc`.

8.18.3 Member Function Documentation

8.18.3.1 `bool pooling_QE::isReady (void) const` [virtual]

Reimplemented from `outputAnalyser` (p. 183).

Definition at line 88 of file `quantile_estimation.cc`.

References `m_isReady`.

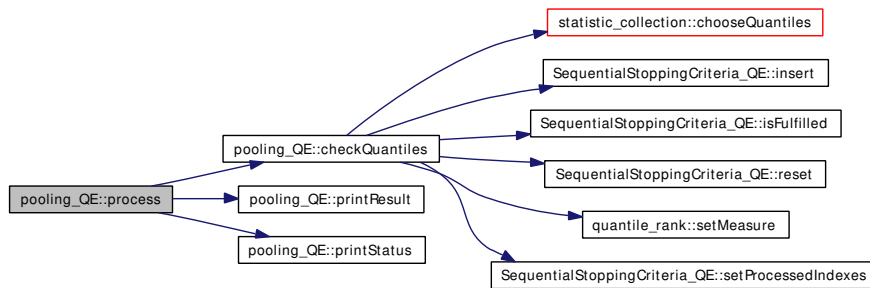
8.18.3.2 `void pooling_QE::process (const std::list< CONTINUOUS > &)` [virtual]

Reimplemented from `outputAnalyser` (p. 183).

Definition at line 92 of file `quantile_estimation.cc`.

References `checkQuantiles()`, `quantile_estimation::m_batchSize`, `m_isReady`, `m_nextCheckpoint`, `m_pool`, `m_posInSpace`, `m_processedBatches`, `outputAnalyser::m_processedIndexes`, `printResult()`, and `printStatus()`.

Here is the call graph for this function:



8.18.3.3 `void pooling_QE::printSetting (void)` [virtual]

Reimplemented from `outputAnalyser` (p. 183).

Definition at line 117 of file `quantile_estimation.cc`.

References `logfile`, `m_minQuantiles`, `s_execute`, `s_pooling_QE`, `s_quantiles_min`, and `s_yes`.

8.18.3.4 `void pooling_QE::printStatus (void)` [virtual]

Reimplemented from `outputAnalyser` (p. 183).

Definition at line 128 of file `quantile_estimation.cc`.

References `logfile`, `quantile_estimation::m_batchSize`, `m_nextCheckpoint`, `m_pool`, `m_posInSpace`, `m_processedBatches`, `outputAnalyser::m_processedIndexes`, and `s_pooling_QE`.

Referenced by `process()`.

8.18.3.5 `void pooling_QE::printResult (void)` [virtual]

Reimplemented from `outputAnalyser` (p. 183).

Definition at line 139 of file `quantile_estimation.cc`.

References `logfile`, `quantile_estimation::m_batchSize`, `m_nextCheckpoint`, `m_pool`, `m_posInSpace`, `m_processedBatches`, `outputAnalyser::m_processedIndexes`, and `s_pooling_QE`.

Referenced by `process()`.

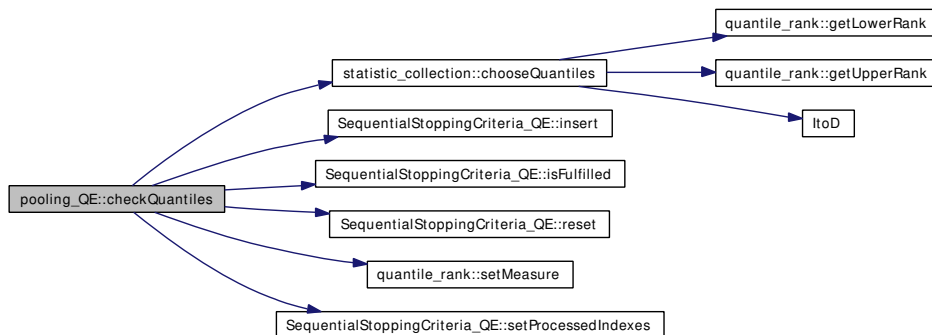
8.18.3.6 `bool pooling_QE::checkQuantiles (void) [private]`

Definition at line 150 of file `quantile_estimation.cc`.

References `statistic_collection::chooseQuantiles()`, `CONTINUOUS`, `INDEX`, `SequentialStoppingCriteria_QE::insert()`, `SequentialStoppingCriteria_QE::isFulfilled()`, `lib_statistic`, `m_alpha`, `m_minQuantiles`, `m_pool`, `outputAnalyser::m_processedIndexes`, `quantile_estimation::m_SSC`, `SequentialStoppingCriteria_QE::reset()`, `quantile_rank::setMeasure()`, and `SequentialStoppingCriteria_QE::setProcessedIndexes()`.

Referenced by `process()`.

Here is the call graph for this function:



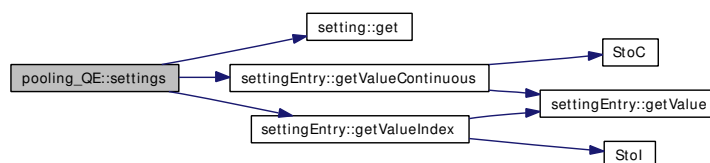
8.18.3.7 `void pooling_QE::settings (void) [private]`

Definition at line 209 of file `quantile_estimation.cc`.

References `setting::get()`, `settingEntry::getValueContinuous()`, `settingEntry::getValueIndex()`, `lib_setting`, `m_alpha`, `m_minQuantiles`, `s_alpha`, `s_pooling_QE`, and `s_quantiles_min`.

Referenced by `pooling_QE()`.

Here is the call graph for this function:



8.18.3.8 TypeOfMethod `quantile_estimation::getType (void) const` [virtual, inherited]

Reimplemented from `outputAnalyser` (p. 87).

Definition at line 22 of file `quantile_estimation.cc`.

References `ESTIMATOR`.

8.18.3.9 `void quantile_estimation::setBatchSize (INDEX p)` [inherited]

Definition at line 26 of file `quantile_estimation.cc`.

References `quantile_estimation::m_batchSize`.

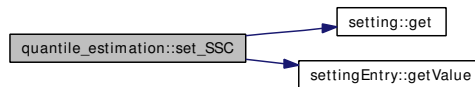
8.18.3.10 `void quantile_estimation::set_SSC (void)` [protected, inherited]

Definition at line 31 of file `quantile_estimation.cc`.

References `setting::get()`, `settingEntry::getValue()`, `lib_setting`, `quantile_estimation::m_SSC`, `s_confidenceInterval_SSC_QE`, `s_deterministic_SSC_QE`, `s_execute`, `s_relativeErrorQuantile_SSC_QE`, `s_relativeErrorRange_SSC_QE`, and `s_yes`.

Referenced by `quantile_estimation::quantile_estimation()`.

Here is the call graph for this function:



8.18.4 Field Documentation

8.18.4.1 `INDEX pooling_QE::m_nextCheckpoint` [private]

Definition at line 45 of file `quantile_estimation.h`.

Referenced by `printResult()`, `printStatus()`, and `process()`.

8.18.4.2 `INDEX pooling_QE::m_processedBatches` [private]

Definition at line 46 of file `quantile_estimation.h`.

Referenced by `printResult()`, `printStatus()`, and `process()`.

8.18.4.3 `INDEX pooling_QE::m_posInSpace` [private]

Definition at line 47 of file `quantile_estimation.h`.

Referenced by `printResult()`, `printStatus()`, and `process()`.

8.18.4.4 `INDEX pooling_QE::m_minQuantiles` [private]

Definition at line 48 of file `quantile_estimation.h`.

Referenced by `checkQuantiles()`, `printSetting()`, and `settings()`.

8.18.4.5 CONTINUOUS pooling_QE::m_alpha [private]

Definition at line 49 of file `quantile_estimation.h`.

Referenced by `checkQuantiles()`, and `settings()`.

8.18.4.6 bool pooling_QE::m_isReady [private]

Definition at line 50 of file `quantile_estimation.h`.

Referenced by `isReady()`, and `process()`.

8.18.4.7 std::list<CONTINUOUS> pooling_QE::m_pool [private]

Definition at line 51 of file `quantile_estimation.h`.

Referenced by `checkQuantiles()`, `printResult()`, `printStatus()`, and `process()`.

8.18.4.8 INDEX quantile_estimation::m_batchSize [protected, inherited]

Definition at line 26 of file `quantile_estimation.h`.

Referenced by `spectral_analysis_QE::checkQuantiles()`, `batch_mean_QE::checkQuantiles()`, `spectral_analysis_QE::collapse()`, `batch_mean_QE::collapse()`, `spectral_analysis_QE::printResult()`, `batch_mean_QE::printResult()`, `printResult()`, `spectral_analysis_QE::printStatus()`, `batch_mean_QE::printStatus()`, `printStatus()`, `spectral_analysis_QE::process()`, `batch_mean_QE::process()`, `process()`, and `quantile_estimation::setBatchSize()`.

8.18.4.9 SequentialStoppingCriteria_QE* quantile_estimation::m_SSC [protected, inherited]

Definition at line 27 of file `quantile_estimation.h`.

Referenced by `spectral_analysis_QE::checkQuantiles()`, `batch_mean_QE::checkQuantiles()`, `checkQuantiles()`, `quantile_estimation::set_SSC()`, and `quantile_estimation::~~quantile_estimation()`.

8.18.4.10 INDEX outputAnalyser::m_processedIndexes [protected, inherited]

Definition at line 20 of file `basic.h`.

Referenced by `evolution::calculateQuantiles()`, `spectral_analysis_QE::checkQuantiles()`, `batch_mean_QE::checkQuantiles()`, `checkQuantiles()`, `deterministic_TPD::isReady()`, `evolution::isReady()`, `sequential_TPD::printResult()`, `deterministic_TPD::printResult()`, `spectral_analysis_QE::printResult()`, `batch_mean_QE::printResult()`, `printResult()`, `batching::printResult()`, `sequential_TPD::printStatus()`, `deterministic_TPD::printStatus()`, `evolution::printStatus()`, `spectral_analysis_QE::printStatus()`, `batch_mean_QE::printStatus()`, `printStatus()`, `batching::printStatus()`, `sequential_TPD::process()`, `deterministic_TPD::process()`, `evolution::process()`, `spectral_analysis_QE::process()`, `batch_mean_QE::process()`, `process()`, `batching::process()`, `outputAnalyser::process()`, `sequential_TPD::sub_collect()`, `sequential_TPD::sub_compare()`, and `sequential_TPD::sub_initialize()`.

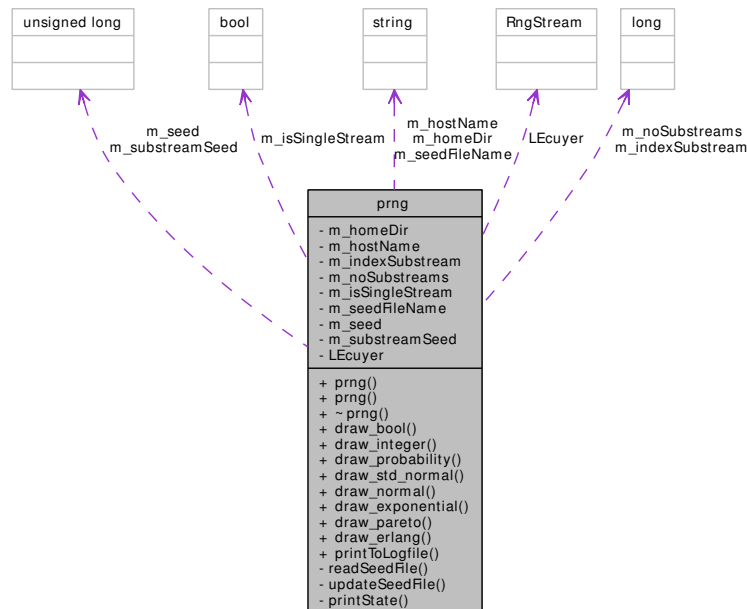
The documentation for this class was generated from the following files:

- `quantile_estimation.h`
- `quantile_estimation.cc`

8.19 prng Class Reference

```
#include <prng.h>
```

Collaboration diagram for prng:



Public Member Functions

- **prng** (void)
- **prng** (long indexSubstream, long noSubstreams)
- **~prng** (void)
- **bool draw_bool** (void)
- **long draw_integer** (long low, long high)
- **double draw_probability** (void)
- **double draw_std_normal** (void)
- **double draw_normal** (double mu, double sigma)
- **double draw_exponential** (double beta)
- **double draw_pareto** (double alpha)
- **double draw_erlang** (double beta, long dim)
- **void printToLogfile** (void)

Private Member Functions

- **void readSeedFile** (void)
- **void updateSeedFile** (void)
- **void printState** (void)

Private Attributes

- `std::string m_homeDir`
- `std::string m_hostName`
- `long m_indexSubstream`
- `long m_noSubstreams`
- `bool m_isSingleStream`
- `std::string m_seedFileName`
- `unsigned long m_seed [6]`
- `unsigned long m_substreamSeed [6]`
- `RngStream LEcuyer`

8.19.1 Detailed Description

Definition at line 7 of file `prng.h`.

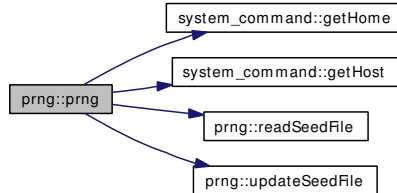
8.19.2 Constructor & Destructor Documentation

8.19.2.1 `prng::prng (void)`

Definition at line 15 of file `prng.cc`.

References `system_command::getHome()`, `system_command::getHost()`, `LEcuyer`, `lib_system`, `m_homeDir`, `m_hostName`, `m_seed`, `m_substreamSeed`, `readSeedFile()`, and `updateSeedFile()`.

Here is the call graph for this function:

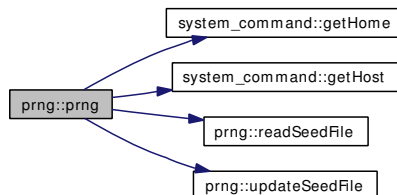


8.19.2.2 `prng::prng (long indexSubstream, long noSubstreams)`

Definition at line 53 of file `prng.cc`.

References `system_command::getHome()`, `system_command::getHost()`, `LEcuyer`, `lib_system`, `m_homeDir`, `m_hostName`, `m_seed`, `m_substreamSeed`, `readSeedFile()`, and `updateSeedFile()`.

Here is the call graph for this function:



8.19.2.3 prng::~~prng (void)

Definition at line 79 of file prng.cc.

8.19.3 Member Function Documentation

8.19.3.1 bool prng::draw_bool (void)

Definition at line 82 of file prng.cc.

References draw_probability().

Here is the call graph for this function:



8.19.3.2 long prng::draw_integer (long low, long high)

Definition at line 88 of file prng.cc.

References LEcuyer.

Referenced by `statistic_collection::generateRandomPermutation()`, and `sequential_TPD::sub_compare()`.

8.19.3.3 double prng::draw_probability (void)

Definition at line 92 of file prng.cc.

References LEcuyer.

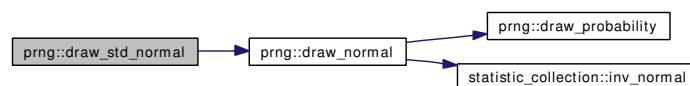
Referenced by `draw_bool()`, `draw_erlang()`, `draw_exponential()`, `draw_normal()`, `draw_pareto()`, and `sequential_TPD::sub_compare()`.

8.19.3.4 double prng::draw_std_normal (void)

Definition at line 96 of file prng.cc.

References draw_normal().

Here is the call graph for this function:



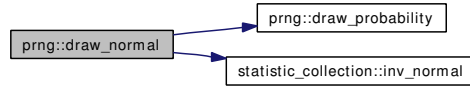
8.19.3.5 double prng::draw_normal (double mu, double sigma)

Definition at line 100 of file prng.cc.

References draw_probability(), `statistic_collection::inv_normal()`, and `lib_statistic`.

Referenced by `draw_std_normal()`.

Here is the call graph for this function:



8.19.3.6 `double prng::draw_exponential (double beta)`

Definition at line 107 of file `prng.cc`.

References `draw_probability()`.

Here is the call graph for this function:



8.19.3.7 `double prng::draw_pareto (double alpha)`

Definition at line 114 of file `prng.cc`.

References `draw_probability()`.

Here is the call graph for this function:



8.19.3.8 `double prng::draw_erlang (double beta, long dim)`

Definition at line 121 of file `prng.cc`.

References `draw_probability()`.

Here is the call graph for this function:



8.19.3.9 `void prng::printToLogfile (void)`

Definition at line 133 of file `prng.cc`.

References `logfile`, `m_indexSubstream`, `m_isSingleStream`, `m_noSubstreams`, `m_seed`, and `m_substreamSeed`.

Referenced by `main()`.

8.19.3.10 void prng::readSeedFile (void) [private]

Definition at line 153 of file prng.cc.

References m_homeDir, m_hostName, m_isSingleStream, m_seed, and m_seedFileName.

Referenced by prng().

8.19.3.11 void prng::updateSeedFile (void) [private]

Definition at line 174 of file prng.cc.

References LEcuyer, and m_seedFileName.

Referenced by prng().

8.19.3.12 void prng::printStats (void) [private]

Definition at line 185 of file prng.cc.

References LEcuyer.

8.19.4 Field Documentation**8.19.4.1 std::string prng::m_homeDir [private]**

Definition at line 25 of file prng.h.

Referenced by prng(), and readSeedFile().

8.19.4.2 std::string prng::m_hostName [private]

Definition at line 26 of file prng.h.

Referenced by prng(), and readSeedFile().

8.19.4.3 long prng::m_indexSubstream [private]

Definition at line 32 of file prng.h.

Referenced by printToLogfile().

8.19.4.4 long prng::m_noSubstreams [private]

Definition at line 33 of file prng.h.

Referenced by printToLogfile().

8.19.4.5 bool prng::m_isSingleStream [private]

Definition at line 34 of file prng.h.

Referenced by printToLogfile(), and readSeedFile().

8.19.4.6 `std::string prng::m_seedFileName` [private]

Definition at line 35 of file prng.h.

Referenced by `readSeedFile()`, and `updateSeedFile()`.

8.19.4.7 `unsigned long prng::m_seed[6]` [private]

Definition at line 36 of file prng.h.

Referenced by `printToLogfile()`, `prng()`, and `readSeedFile()`.

8.19.4.8 `unsigned long prng::m_substreamSeed[6]` [private]

Definition at line 37 of file prng.h.

Referenced by `printToLogfile()`, and `prng()`.

8.19.4.9 `RngStream prng::LEcuyer` [private]

Definition at line 39 of file prng.h.

Referenced by `draw_integer()`, `draw_probability()`, `printState()`, `prng()`, and `updateSeedFile()`.

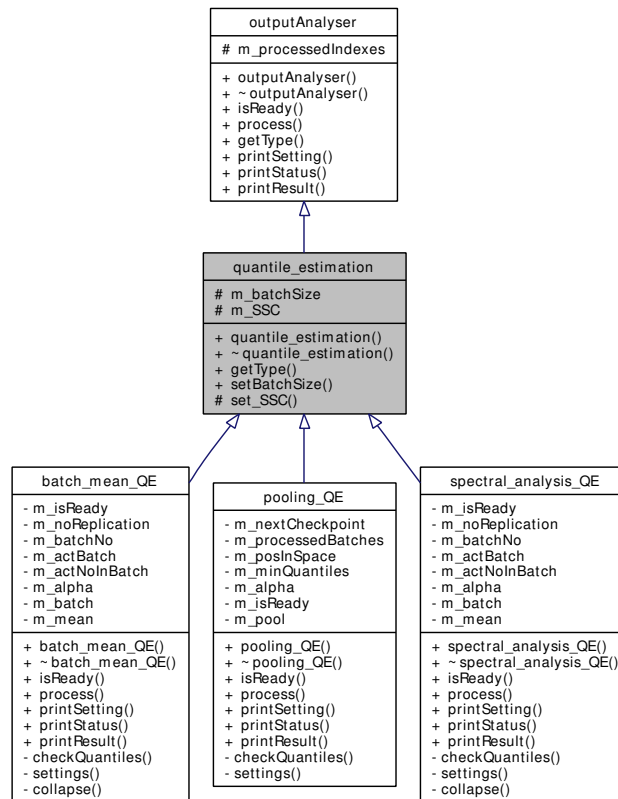
The documentation for this class was generated from the following files:

- `prng.h`
- `prng.cc`

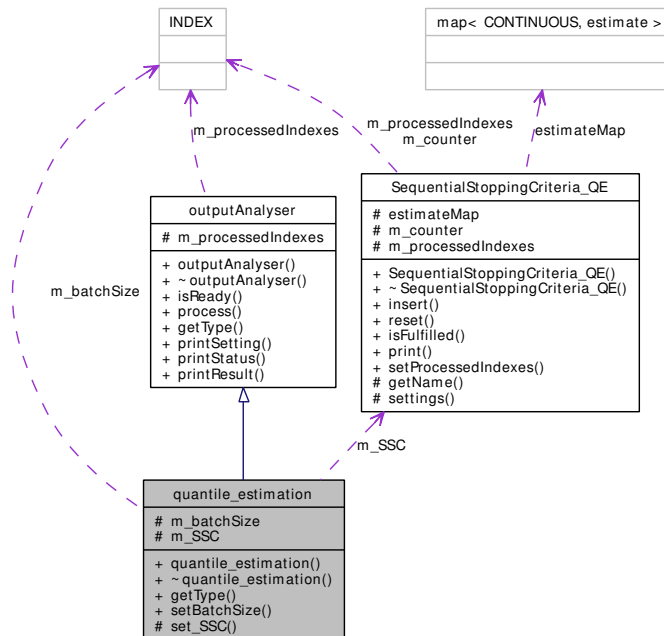
8.20 quantile_estimation Class Reference

```
#include <quantile_estimation.h>
```

Inheritance diagram for quantile_estimation:



Collaboration diagram for quantile_estimation:



Public Member Functions

- `quantile_estimation` (void)
- virtual `~quantile_estimation` (void)
- virtual `TypeOfMethod getType` (void) const
- void `setBatchSize` (INDEX p)
- virtual bool `isReady` (void) const
- virtual void `process` (const std::list< CONTINUOUS > &)
- virtual void `printSetting` (void)
- virtual void `printStatus` (void)
- virtual void `printResult` (void)

Protected Member Functions

- void `set_SSC` (void)

Protected Attributes

- INDEX `m_batchSize`
- `SequentialStoppingCriteria_QE * m_SSC`
- INDEX `m_processedIndexes`

8.20.1 Detailed Description

Definition at line 10 of file `quantile_estimation.h`.

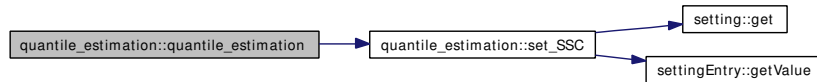
8.20.2 Constructor & Destructor Documentation

8.20.2.1 quantile_estimation::quantile_estimation (void)

Definition at line 8 of file quantile_estimation.cc.

References `set_SSC()`.

Here is the call graph for this function:



8.20.2.2 quantile_estimation::~~quantile_estimation (void) [virtual]

Definition at line 15 of file quantile_estimation.cc.

References `m_SSC`.

8.20.3 Member Function Documentation

8.20.3.1 TypeOfMethod quantile_estimation::getType (void) const [virtual]

Reimplemented from `outputAnalyser` (p. 87).

Definition at line 22 of file quantile_estimation.cc.

References `ESTIMATOR`.

8.20.3.2 void quantile_estimation::setBatchSize (INDEX p)

Definition at line 26 of file quantile_estimation.cc.

References `m_batchSize`.

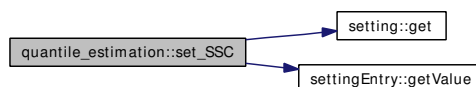
8.20.3.3 void quantile_estimation::set_SSC (void) [protected]

Definition at line 31 of file quantile_estimation.cc.

References `setting::get()`, `settingEntry::getValue()`, `lib_setting`, `m_SSC`, `s_confidenceInterval_SSC_QE`, `s_deterministic_SSC_QE`, `s_execute`, `s_relativeErrorQuantile_SSC_QE`, `s_relativeErrorRange_SSC_QE`, and `s_yes`.

Referenced by `quantile_estimation()`.

Here is the call graph for this function:



8.20.3.4 `bool outputAnalyser::isReady (void) const` [virtual, inherited]

Reimplemented in `batching` (p.40), `pooling_QE` (p.92), `batch_mean_QE` (p.31), `spectral_analysis_QE` (p.153), `evolution` (p.69), `deterministic_TPD` (p.61), and `sequential_TPD` (p.128).

Definition at line 11 of file basic.cc.

8.20.3.5 `void outputAnalyser::process (const std::list< CONTINUOUS > &)` [virtual, inherited]

Reimplemented in `batching` (p.40), `pooling_QE` (p.92), `batch_mean_QE` (p.32), `spectral_analysis_QE` (p.153), `evolution` (p.69), `deterministic_TPD` (p.61), and `sequential_TPD` (p.128).

Definition at line 15 of file basic.cc.

References `outputAnalyser::m_processedIndexes`.

8.20.3.6 `void outputAnalyser::printSetting (void)` [virtual, inherited]

Reimplemented in `batching` (p.41), `pooling_QE` (p.92), `batch_mean_QE` (p.32), `spectral_analysis_QE` (p.154), `evolution` (p.69), `deterministic_TPD` (p.61), and `sequential_TPD` (p.129).

Definition at line 23 of file basic.cc.

8.20.3.7 `void outputAnalyser::printStatus (void)` [virtual, inherited]

Reimplemented in `batching` (p.41), `pooling_QE` (p.92), `batch_mean_QE` (p.32), `spectral_analysis_QE` (p.154), `evolution` (p.69), `deterministic_TPD` (p.62), and `sequential_TPD` (p.129).

Definition at line 26 of file basic.cc.

8.20.3.8 `void outputAnalyser::printResult (void)` [virtual, inherited]

Reimplemented in `batching` (p.41), `pooling_QE` (p.92), `batch_mean_QE` (p.32), `spectral_analysis_QE` (p.154), `evolution` (p.69), `deterministic_TPD` (p.62), and `sequential_TPD` (p.129).

Definition at line 29 of file basic.cc.

8.20.4 Field Documentation**8.20.4.1** `INDEX quantile_estimation::m_batchSize` [protected]

Definition at line 26 of file `quantile_estimation.h`.

Referenced by `spectral_analysis_QE::checkQuantiles()`, `batch_mean_QE::checkQuantiles()`, `spectral_analysis_QE::collapse()`, `batch_mean_QE::collapse()`, `spectral_analysis_QE::printResult()`, `batch_mean_QE::printResult()`, `pooling_QE::printResult()`, `spectral_analysis_QE::printStatus()`, `batch_mean_QE::printStatus()`, `pooling_QE::printStatus()`, `spectral_`

analysis_QE::process(), batch_mean_QE::process(), pooling_QE::process(), and setBatchSize().

8.20.4.2 SequentialStoppingCriteria_QE* quantile_estimation::m_SSC [protected]

Definition at line 27 of file quantile_estimation.h.

Referenced by spectral_analysis_QE::checkQuantiles(), batch_mean_QE::checkQuantiles(), pooling_QE::checkQuantiles(), set_SSC(), and ~quantile_estimation().

8.20.4.3 INDEX outputAnalyser::m_processedIndexes [protected, inherited]

Definition at line 20 of file basic.h.

Referenced by evolution::calculateQuantiles(), spectral_analysis_QE::checkQuantiles(), batch_mean_QE::checkQuantiles(), pooling_QE::checkQuantiles(), deterministic_TPD::isReady(), evolution::isReady(), sequential_TPD::printResult(), deterministic_TPD::printResult(), spectral_analysis_QE::printResult(), batch_mean_QE::printResult(), pooling_QE::printResult(), batching::printResult(), sequential_TPD::printStatus(), deterministic_TPD::printStatus(), evolution::printStatus(), spectral_analysis_QE::printStatus(), batch_mean_QE::printStatus(), pooling_QE::printStatus(), batching::printStatus(), sequential_TPD::process(), deterministic_TPD::process(), evolution::process(), spectral_analysis_QE::process(), batch_mean_QE::process(), pooling_QE::process(), batching::process(), outputAnalyser::process(), sequential_TPD::sub_collect(), sequential_TPD::sub_compare(), and sequential_TPD::sub_initialize().

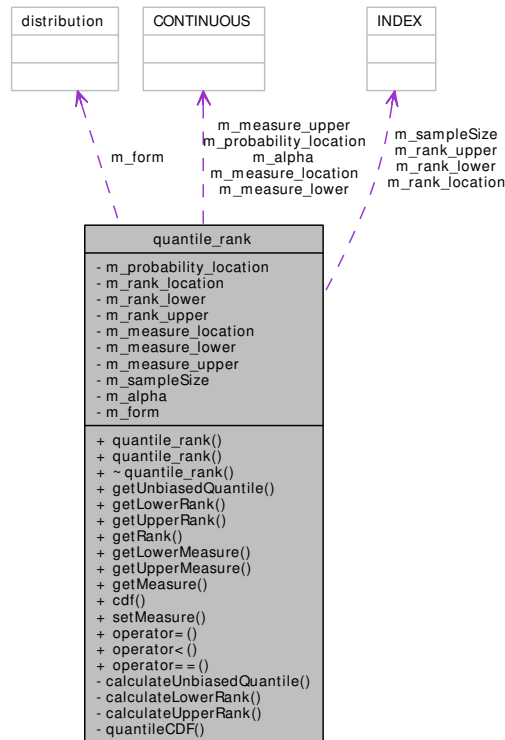
The documentation for this class was generated from the following files:

- **quantile_estimation.h**
- **quantile_estimation.cc**

8.21 quantile_rank Class Reference

```
#include <statistic.h>
```

Collaboration diagram for quantile_rank:



Public Member Functions

- **quantile_rank** (INDEX rank, INDEX sampleSize, CONTINUOUS alpha, **distribution** form=UNSPECIFIED)
- **quantile_rank** (const **quantile_rank** &other)
- **~quantile_rank** (void)
- CONTINUOUS **getUnbiasedQuantile** (void) const
- INDEX **getLowerRank** (void) const
- INDEX **getUpperRank** (void) const
- INDEX **getRank** (void) const
- CONTINUOUS **getLowerMeasure** (void) const
- CONTINUOUS **getUpperMeasure** (void) const
- CONTINUOUS **getMeasure** (void) const
- void **cdf** (std::list< CONTINUOUS > &cumulation) const
- void **setMeasure** (CONTINUOUS measure_lower, CONTINUOUS measure_location, CONTINUOUS measure_upper)
- const **quantile_rank** & **operator=** (const **quantile_rank** &other)
- bool **operator<** (const **quantile_rank** &other) const
- bool **operator==** (const **quantile_rank** &other) const

Private Member Functions

- CONTINUOUS **calculateUnbiasedQuantile** (INDEX rank, INDEX sampleSize, **distribution** form=UNSPECIFIED)
- INDEX **calculateLowerRank** (CONTINUOUS quantile, INDEX rank, INDEX sampleSize, CONTINUOUS alpha)
- INDEX **calculateUpperRank** (CONTINUOUS quantile, INDEX rank, INDEX sampleSize, CONTINUOUS alpha)
- CONTINUOUS **quantileCDF** (INDEX rank, INDEX p, CONTINUOUS q) const

Private Attributes

- CONTINUOUS **m_probability_location**
- INDEX **m_rank_location**
- INDEX **m_rank_lower**
- INDEX **m_rank_upper**
- CONTINUOUS **m_measure_location**
- CONTINUOUS **m_measure_lower**
- CONTINUOUS **m_measure_upper**
- INDEX **m_sampleSize**
- CONTINUOUS **m_alpha**
- **distribution m_form**

8.21.1 Detailed Description

Definition at line 152 of file statistic.h.

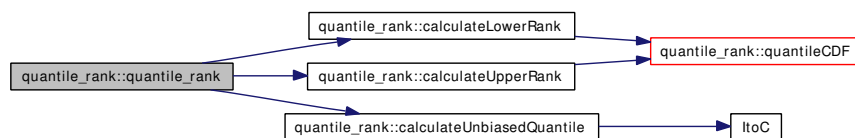
8.21.2 Constructor & Destructor Documentation

8.21.2.1 quantile_rank::quantile_rank (INDEX *rank*, INDEX *sampleSize*, CONTINUOUS *alpha*, *distribution form* = UNSPECIFIED)

Definition at line 1544 of file statistic.cc.

References `calculateLowerRank()`, `calculateUnbiasedQuantile()`, `calculateUpperRank()`, `m_alpha`, `m_form`, `m_probability_location`, `m_rank_location`, `m_rank_lower`, `m_rank_upper`, and `m_sampleSize`.

Here is the call graph for this function:



8.21.2.2 quantile_rank::quantile_rank (const quantile_rank & *other*)

Definition at line 1567 of file statistic.cc.

8.21.2.3 `quantile_rank::~~quantile_rank (void)`

Definition at line 1580 of file `statistic.cc`.

8.21.3 Member Function Documentation

8.21.3.1 CONTINUOUS `quantile_rank::getUnbiasedQuantile (void) const` [inline]

Definition at line 162 of file `statistic.h`.

References `m_probability_location`.

8.21.3.2 INDEX `quantile_rank::getLowerRank (void) const` [inline]

Definition at line 163 of file `statistic.h`.

References `m_rank_lower`.

Referenced by `statistic_collection::chooseQuantiles()`, and `statistic_collection::chooseQuantiles_old()`.

8.21.3.3 INDEX `quantile_rank::getUpperRank (void) const` [inline]

Definition at line 164 of file `statistic.h`.

References `m_rank_upper`.

Referenced by `statistic_collection::chooseQuantiles()`, and `statistic_collection::chooseQuantiles_old()`.

8.21.3.4 INDEX `quantile_rank::getRank (void) const` [inline]

Definition at line 165 of file `statistic.h`.

References `m_rank_location`.

Referenced by `statistic_collection::chooseQuantiles_old()`.

8.21.3.5 CONTINUOUS `quantile_rank::getLowerMeasure (void) const` [inline]

Definition at line 166 of file `statistic.h`.

References `m_measure_lower`.

8.21.3.6 CONTINUOUS `quantile_rank::getUpperMeasure (void) const` [inline]

Definition at line 167 of file `statistic.h`.

References `m_measure_upper`.

8.21.3.7 CONTINUOUS `quantile_rank::getMeasure (void) const` [inline]

Definition at line 168 of file `statistic.h`.

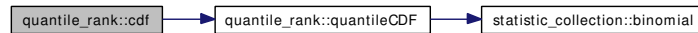
References `m_measure_location`.

8.21.3.8 `void quantile_rank::cdf (std::list< CONTINUOUS > & cumulation) const`

Definition at line 1654 of file `statistic.cc`.

References `INDEX`, `m_probability_location`, `m_sampleSize`, and `quantileCDF()`.

Here is the call graph for this function:



8.21.3.9 `void quantile_rank::setMeasure (CONTINUOUS measure_lower, CONTINUOUS measure_location, CONTINUOUS measure_upper) [inline]`

Definition at line 171 of file `statistic.h`.

References `m_measure_location`, `m_measure_lower`, and `m_measure_upper`.

Referenced by `pooling_QE::checkQuantiles()`.

8.21.3.10 `const quantile_rank& quantile_rank::operator= (const quantile_rank & other) [inline]`

Definition at line 179 of file `statistic.h`.

References `m_alpha`, `m_measure_location`, `m_measure_lower`, `m_measure_upper`, `m_probability_location`, `m_rank_location`, `m_rank_lower`, `m_rank_upper`, and `m_sampleSize`.

8.21.3.11 `bool quantile_rank::operator< (const quantile_rank & other) const [inline]`

Definition at line 192 of file `statistic.h`.

References `m_rank_location`.

8.21.3.12 `bool quantile_rank::operator== (const quantile_rank & other) const [inline]`

Definition at line 196 of file `statistic.h`.

References `m_alpha`, `m_rank_location`, and `m_sampleSize`.

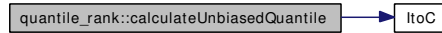
8.21.3.13 `CONTINUOUS quantile_rank::calculateUnbiasedQuantile (INDEX rank, INDEX sampleSize, distribution form = UNSPECIFIED) [private]`

Definition at line 1583 of file `statistic.cc`.

References `CONTINUOUS`, `EXPONENTIAL`, `ItoC()`, `NORMAL`, `UNIFORM`, and `UNSPECIFIED`.

Referenced by `quantile_rank()`.

Here is the call graph for this function:



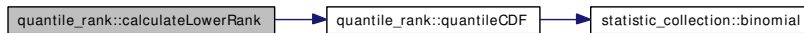
8.21.3.14 INDEX `quantile_rank::calculateLowerRank` (CONTINUOUS *quantile*, INDEX *rank*, INDEX *sampleSize*, CONTINUOUS *alpha*) [private]

Definition at line 1622 of file `statistic.cc`.

References INDEX, and `quantileCDF()`.

Referenced by `quantile_rank()`.

Here is the call graph for this function:



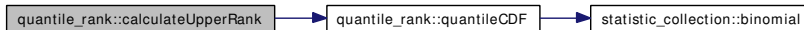
8.21.3.15 INDEX `quantile_rank::calculateUpperRank` (CONTINUOUS *quantile*, INDEX *rank*, INDEX *sampleSize*, CONTINUOUS *alpha*) [private]

Definition at line 1634 of file `statistic.cc`.

References INDEX, and `quantileCDF()`.

Referenced by `quantile_rank()`.

Here is the call graph for this function:



8.21.3.16 CONTINUOUS `quantile_rank::quantileCDF` (INDEX *rank*, INDEX *p*, CONTINUOUS *q*) `const` [private]

Definition at line 1646 of file `statistic.cc`.

References `statistic_collection::binomial()`, CONTINUOUS, and `lib_statistic`.

Referenced by `calculateLowerRank()`, `calculateUpperRank()`, and `cdf()`.

Here is the call graph for this function:



8.21.4 Field Documentation

8.21.4.1 CONTINUOUS quantile_rank::m_probability_location [private]

Definition at line 202 of file statistic.h.

Referenced by `cdf()`, `getUnbiasedQuantile()`, `operator=()`, and `quantile_rank()`.

8.21.4.2 INDEX quantile_rank::m_rank_location [private]

Definition at line 206 of file statistic.h.

Referenced by `getRank()`, `operator<()`, `operator=()`, `operator==(())`, and `quantile_rank()`.

8.21.4.3 INDEX quantile_rank::m_rank_lower [private]

Definition at line 207 of file statistic.h.

Referenced by `getLowerRank()`, `operator=()`, and `quantile_rank()`.

8.21.4.4 INDEX quantile_rank::m_rank_upper [private]

Definition at line 208 of file statistic.h.

Referenced by `getUpperRank()`, `operator=()`, and `quantile_rank()`.

8.21.4.5 CONTINUOUS quantile_rank::m_measure_location [private]

Definition at line 209 of file statistic.h.

Referenced by `getMeasure()`, `operator=()`, and `setMeasure()`.

8.21.4.6 CONTINUOUS quantile_rank::m_measure_lower [private]

Definition at line 210 of file statistic.h.

Referenced by `getLowerMeasure()`, `operator=()`, and `setMeasure()`.

8.21.4.7 CONTINUOUS quantile_rank::m_measure_upper [private]

Definition at line 211 of file statistic.h.

Referenced by `getUpperMeasure()`, `operator=()`, and `setMeasure()`.

8.21.4.8 INDEX quantile_rank::m_sampleSize [private]

Definition at line 212 of file statistic.h.

Referenced by `cdf()`, `operator=()`, `operator==(())`, and `quantile_rank()`.

8.21.4.9 CONTINUOUS quantile_rank::m_alpha [private]

Definition at line 213 of file statistic.h.

Referenced by `operator=()`, `operator==()`, and `quantile_rank()`.

8.21.4.10 `distribution quantile_rank::m_form` [private]

Definition at line 214 of file `statistic.h`.

Referenced by `quantile_rank()`.

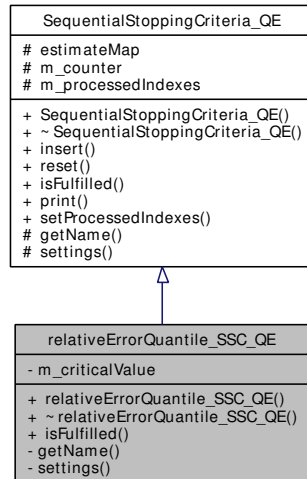
The documentation for this class was generated from the following files:

- `statistic.h`
- `statistic.cc`

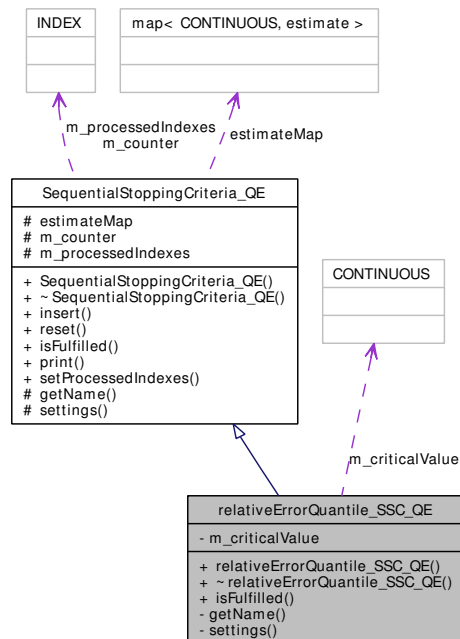
8.22 relativeErrorQuantile_SSC_QE Class Reference

```
#include <quantile_estimation.h>
```

Inheritance diagram for relativeErrorQuantile_SSC_QE:



Collaboration diagram for relativeErrorQuantile_SSC_QE:



Public Member Functions

- `relativeErrorQuantile_SSC_QE` (void)
- `~relativeErrorQuantile_SSC_QE` (void)
- `bool isFulfilled` (void)

- void **insert** (const CONTINUOUS &location, const CONTINUOUS &probability, const CONTINUOUS &absoluteErrorNeg, const CONTINUOUS &absoluteErrorPos)
- void **reset** (void)
- void **print** (bool isFinal=false)
- void **setProcessedIndexes** (INDEX i)

Protected Attributes

- std::map< CONTINUOUS, estimate > **estimateMap**
- INDEX **m_counter**
- INDEX **m_processedIndexes**

Private Member Functions

- std::string **getName** (void)
- void **settings** (void)

Private Attributes

- CONTINUOUS **m_criticalValue**

8.22.1 Detailed Description

Definition at line 163 of file `quantile_estimation.h`.

8.22.2 Constructor & Destructor Documentation

8.22.2.1 `relativeErrorQuantile_SSC_QE::relativeErrorQuantile_SSC_QE` (void)

Definition at line 835 of file `quantile_estimation.cc`.

8.22.2.2 `relativeErrorQuantile_SSC_QE::~~relativeErrorQuantile_SSC_QE` (void)

Definition at line 841 of file `quantile_estimation.cc`.

8.22.3 Member Function Documentation

8.22.3.1 `bool relativeErrorQuantile_SSC_QE::isFulfilled` (void) [virtual]

Reimplemented from `SequentialStoppingCriteria_QE` (p.138).

Definition at line 844 of file `quantile_estimation.cc`.

References `CONTINUOUS`, and `m_criticalValue`.

8.22.3.2 `std::string relativeErrorQuantile_SSC_QE::getName (void)` [inline, private, virtual]

Reimplemented from `SequentialStoppingCriteria_QE` (p.138).

Definition at line 172 of file `quantile_estimation.h`.

References `s_relativeErrorQuantile_SSC_QE`.

8.22.3.3 `void relativeErrorQuantile_SSC_QE::settings (void)` [private, virtual]

Reimplemented from `SequentialStoppingCriteria_QE` (p.139).

Definition at line 862 of file `quantile_estimation.cc`.

8.22.3.4 `void SequentialStoppingCriteria_QE::insert (const CONTINUOUS & location, const CONTINUOUS & probability, const CONTINUOUS & absoluteErrorNeg, const CONTINUOUS & absoluteErrorPos)` [inherited]

Definition at line 627 of file `quantile_estimation.cc`.

References `SequentialStoppingCriteria_QE::estimate::absoluteErrorNeg`, `SequentialStoppingCriteria_QE::estimate::absoluteErrorPos`, `SequentialStoppingCriteria_QE::estimateMap`, `SequentialStoppingCriteria_QE::estimate::location`, and `SequentialStoppingCriteria_QE::estimate::probability`.

Referenced by `spectral_analysis_QE::checkQuantiles()`, `batch_mean_QE::checkQuantiles()`, and `pooling_QE::checkQuantiles()`.

8.22.3.5 `void SequentialStoppingCriteria_QE::reset (void)` [inherited]

Definition at line 639 of file `quantile_estimation.cc`.

References `SequentialStoppingCriteria_QE::estimateMap`.

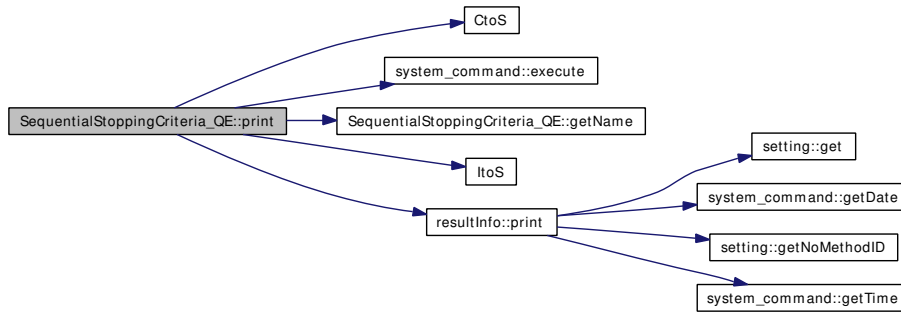
Referenced by `spectral_analysis_QE::checkQuantiles()`, `batch_mean_QE::checkQuantiles()`, and `pooling_QE::checkQuantiles()`.

8.22.3.6 `void SequentialStoppingCriteria_QE::print (bool isFinal = false)` [inherited]

Definition at line 652 of file `quantile_estimation.cc`.

References `CONTINUOUS`, `CtoS()`, `SequentialStoppingCriteria_QE::estimateMap`, `system_command::execute()`, `SequentialStoppingCriteria_QE::getName()`, `ItoS()`, `lib_system`, `SequentialStoppingCriteria_QE::m_counter`, `SequentialStoppingCriteria_QE::m_processedIndexes`, `resultInfo::print()`, and `resultfile`.

Here is the call graph for this function:



8.22.3.7 void SequentialStoppingCriteria_QE::setProcessedIndexes (INDEX *i*) [inline, inherited]

Definition at line 118 of file `quantile_estimation.h`.

References `SequentialStoppingCriteria_QE::m_processedIndexes`.

Referenced by `spectral_analysis_QE::checkQuantiles()`, `batch_mean_QE::checkQuantiles()`, and `pooling_QE::checkQuantiles()`.

8.22.4 Field Documentation

8.22.4.1 CONTINUOUS `relativeErrorQuantile_SSC_QE::m_criticalValue` [private]

Definition at line 175 of file `quantile_estimation.h`.

Referenced by `isFulfilled()`.

8.22.4.2 `std::map<CONTINUOUS,estimate> SequentialStoppingCriteria_QE::estimateMap` [protected, inherited]

Definition at line 131 of file `quantile_estimation.h`.

Referenced by `SequentialStoppingCriteria_QE::insert()`, `SequentialStoppingCriteria_QE::print()`, and `SequentialStoppingCriteria_QE::reset()`.

8.22.4.3 INDEX `SequentialStoppingCriteria_QE::m_counter` [protected, inherited]

Definition at line 132 of file `quantile_estimation.h`.

Referenced by `SequentialStoppingCriteria_QE::print()`.

8.22.4.4 INDEX `SequentialStoppingCriteria_QE::m_processedIndexes` [protected, inherited]

Definition at line 133 of file `quantile_estimation.h`.

Referenced by `SequentialStoppingCriteria_QE::print()`, and `SequentialStoppingCriteria_QE::setProcessedIndexes()`.

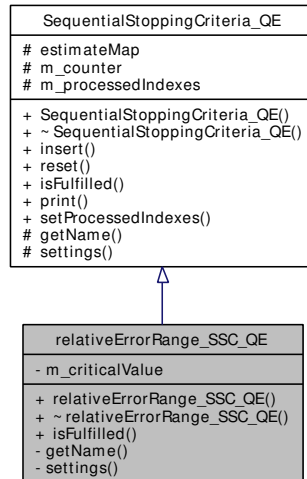
The documentation for this class was generated from the following files:

- `quantile_estimation.h`
- `quantile_estimation.cc`

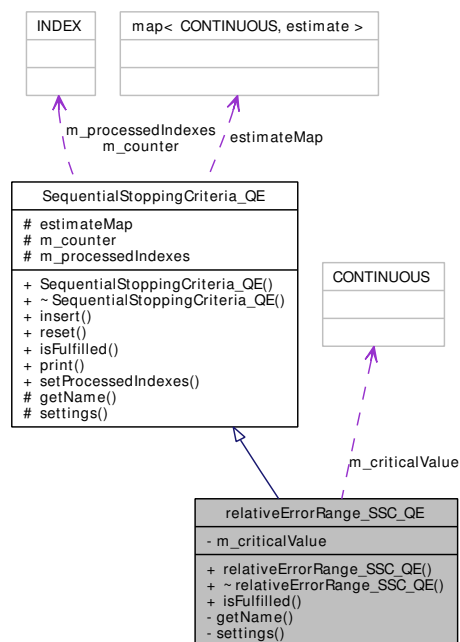
8.23 relativeErrorRange_SSC_QE Class Reference

```
#include <quantile_estimation.h>
```

Inheritance diagram for relativeErrorRange_SSC_QE:



Collaboration diagram for relativeErrorRange_SSC_QE:



Public Member Functions

- `relativeErrorRange_SSC_QE` (void)
- `~relativeErrorRange_SSC_QE` (void)
- `bool isFulfilled` (void)

- void **insert** (const CONTINUOUS &location, const CONTINUOUS &probability, const CONTINUOUS &absoluteErrorNeg, const CONTINUOUS &absoluteErrorPos)
- void **reset** (void)
- void **print** (bool isFinal=false)
- void **setProcessedIndexes** (INDEX i)

Protected Attributes

- `std::map< CONTINUOUS, estimate >` **estimateMap**
- INDEX **m_counter**
- INDEX **m_processedIndexes**

Private Member Functions

- `std::string` **getName** (void)
- void **settings** (void)

Private Attributes

- CONTINUOUS **m_criticalValue**

8.23.1 Detailed Description

Definition at line 178 of file `quantile_estimation.h`.

8.23.2 Constructor & Destructor Documentation

8.23.2.1 `relativeErrorRange_SSC_QE::relativeErrorRange_SSC_QE` (void)

Definition at line 876 of file `quantile_estimation.cc`.

8.23.2.2 `relativeErrorRange_SSC_QE::~~relativeErrorRange_SSC_QE` (void)

Definition at line 882 of file `quantile_estimation.cc`.

8.23.3 Member Function Documentation

8.23.3.1 `bool relativeErrorRange_SSC_QE::isFulfilled` (void) [virtual]

Reimplemented from `SequentialStoppingCriteria_QE` (p.138).

Definition at line 885 of file `quantile_estimation.cc`.

8.23.3.2 `std::string relativeErrorRange_SSC_QE::getName` (void) [inline, private, virtual]

Reimplemented from `SequentialStoppingCriteria_QE` (p.138).

Definition at line 187 of file `quantile_estimation.h`.

References `s_relativeErrorRange_SSC_QE`.

8.23.3.3 `void relativeErrorRange_SSC_QE::settings (void)` [private, virtual]

Reimplemented from `SequentialStoppingCriteria_QE` (p.139).

Definition at line 908 of file `quantile_estimation.cc`.

8.23.3.4 `void SequentialStoppingCriteria_QE::insert (const CONTINUOUS & location, const CONTINUOUS & probability, const CONTINUOUS & absoluteErrorNeg, const CONTINUOUS & absoluteErrorPos)` [inherited]

Definition at line 627 of file `quantile_estimation.cc`.

References `SequentialStoppingCriteria_QE::estimate::absoluteErrorNeg`, `SequentialStoppingCriteria_QE::estimate::absoluteErrorPos`, `SequentialStoppingCriteria_QE::estimateMap`, `SequentialStoppingCriteria_QE::estimate::location`, and `SequentialStoppingCriteria_QE::estimate::probability`.

Referenced by `spectral_analysis_QE::checkQuantiles()`, `batch_mean_QE::checkQuantiles()`, and `pooling_QE::checkQuantiles()`.

8.23.3.5 `void SequentialStoppingCriteria_QE::reset (void)` [inherited]

Definition at line 639 of file `quantile_estimation.cc`.

References `SequentialStoppingCriteria_QE::estimateMap`.

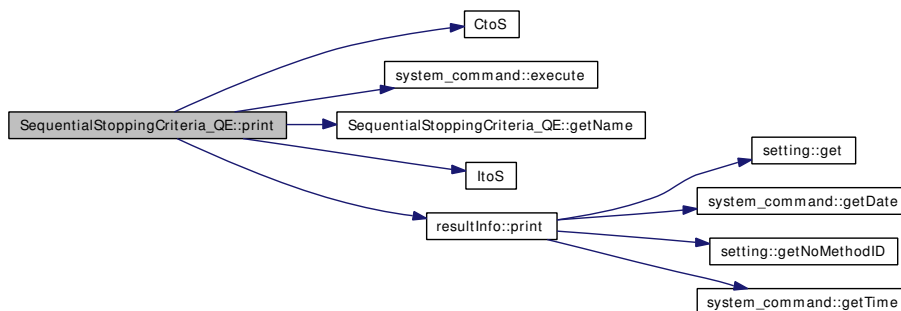
Referenced by `spectral_analysis_QE::checkQuantiles()`, `batch_mean_QE::checkQuantiles()`, and `pooling_QE::checkQuantiles()`.

8.23.3.6 `void SequentialStoppingCriteria_QE::print (bool isFinal = false)` [inherited]

Definition at line 652 of file `quantile_estimation.cc`.

References `CONTINUOUS`, `CtoS()`, `SequentialStoppingCriteria_QE::estimateMap`, `system_command::execute()`, `SequentialStoppingCriteria_QE::getName()`, `ItoS()`, `lib_system`, `SequentialStoppingCriteria_QE::m_counter`, `SequentialStoppingCriteria_QE::m_processedIndexes`, `resultInfo::print()`, and `resultfile`.

Here is the call graph for this function:



8.23.3.7 void SequentialStoppingCriteria_QE::setProcessedIndexes (INDEX *i*) [inline, inherited]

Definition at line 118 of file quantile_estimation.h.

References SequentialStoppingCriteria_QE::m_processedIndexes.

Referenced by spectral_analysis_QE::checkQuantiles(), batch_mean_QE::checkQuantiles(), and pooling_QE::checkQuantiles().

8.23.4 Field Documentation

8.23.4.1 CONTINUOUS relativeErrorRange_SSC_QE::m_criticalValue [private]

Definition at line 190 of file quantile_estimation.h.

8.23.4.2 std::map<CONTINUOUS,estimate> SequentialStoppingCriteria_QE::estimateMap [protected, inherited]

Definition at line 131 of file quantile_estimation.h.

Referenced by SequentialStoppingCriteria_QE::insert(), SequentialStoppingCriteria_QE::print(), and SequentialStoppingCriteria_QE::reset().

8.23.4.3 INDEX SequentialStoppingCriteria_QE::m_counter [protected, inherited]

Definition at line 132 of file quantile_estimation.h.

Referenced by SequentialStoppingCriteria_QE::print().

8.23.4.4 INDEX SequentialStoppingCriteria_QE::m_processedIndexes [protected, inherited]

Definition at line 133 of file quantile_estimation.h.

Referenced by SequentialStoppingCriteria_QE::print(), and SequentialStoppingCriteria_QE::setProcessedIndexes().

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

- `quantile_estimation.h`
- `quantile_estimation.cc`

8.24 resultInfo Class Reference

```
#include <resultfile.h>
```

Public Member Functions

- void **print** (const std::string &message, const std::string &method)

8.24.1 Detailed Description

Definition at line 8 of file resultfile.h.

8.24.2 Member Function Documentation

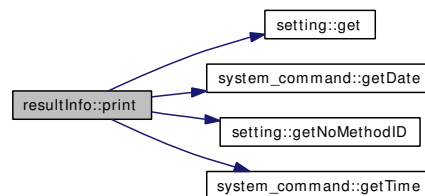
8.24.2.1 void resultInfo::print (const std::string & *message*, const std::string & *method*)

Definition at line 12 of file resultfile.cc.

References `setting::get()`, `system_command::getDate()`, `setting::getNoMethodID()`, `system_command::getTime()`, `lib_setting`, and `lib_system`.

Referenced by `SequentialStoppingCriteria_QE::print()`, `sequential_TPD::printResult()`, and `batching::printResult()`.

Here is the call graph for this function:



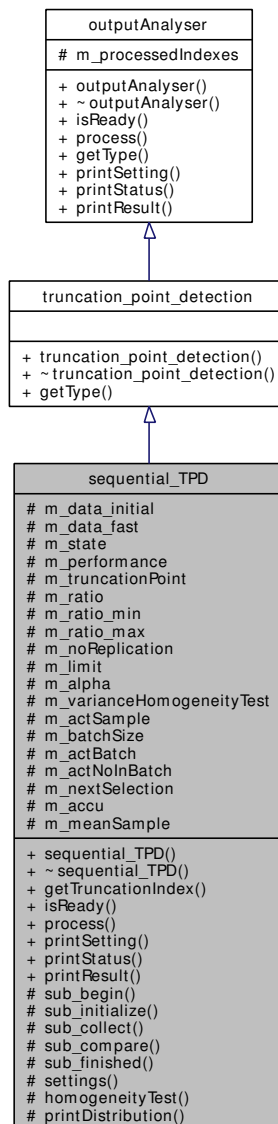
The documentation for this class was generated from the following files:

- `resultfile.h`
- `resultfile.cc`

8.25 sequential_TPD Class Reference

```
#include <truncation_point_detection.h>
```

Inheritance diagram for sequential_TPD:



Collaboration diagram for sequential_TPD:



Public Member Functions

- `sequential_TPD ()`
- `virtual ~sequential_TPD (void)`
- `INDEX getTruncationIndex (void) const`
- `virtual bool isReady (void) const`
- `virtual void process (const std::list< CONTINUOUS > &)`
- `virtual void printSetting (void)`
- `virtual void printStatus (void)`
- `virtual void printResult (void)`
- `virtual TypeOfMethod getType (void) const`

Protected Types

- `enum state { begin = 1, initialize, collect, finished }`
- `enum performance { exact = 1, precise, fast }`

Protected Member Functions

- `void sub_begin (void)`
- `void sub_initialize (void)`
- `void sub_collect (void)`
- `void sub_compare (void)`
- `void sub_finished (void)`
- `void settings (void)`
- `bool homogeneityTest (std::vector< CONTINUOUS > &p1, std::vector< CONTINUOUS > &p2)`
- `void printDistribution (void)`

Protected Attributes

- `std::list< std::vector< CONTINUOUS > > m_data_initial`
- `std::vector< std::vector< CONTINUOUS > > m_data_fast`
- state `m_state`
- performance `m_performance`
- INDEX `m_truncationPoint`
- INDEX `m_ratio`
- INDEX `m_ratio_min`
- INDEX `m_ratio_max`
- INDEX `m_noReplication`
- INDEX `m_limit`
- CONTINUOUS `m_alpha`
- CONTINUOUS `m_varianceHomogeneityTest`
- `std::vector< CONTINUOUS > m_actSample`
- INDEX `m_batchSize`
- INDEX `m_actBatch`
- INDEX `m_actNoInBatch`
- INDEX `m_nextSelection`
- `std::vector< CONTINUOUS > m_accu`
- `std::vector< CONTINUOUS > m_meanSample`
- INDEX `m_processedIndexes`

8.25.1 Detailed Description

Definition at line 29 of file `truncation_point_detection.h`.

8.25.2 Member Enumeration Documentation

8.25.2.1 `enum sequential_TPD::state` [protected]

Enumerator:

begin
initialize
collect
finished

Definition at line 43 of file `truncation_point_detection.h`.

8.25.2.2 `enum sequential_TPD::performance` [protected]

Enumerator:

exact
precise
fast

Definition at line 44 of file `truncation_point_detection.h`.

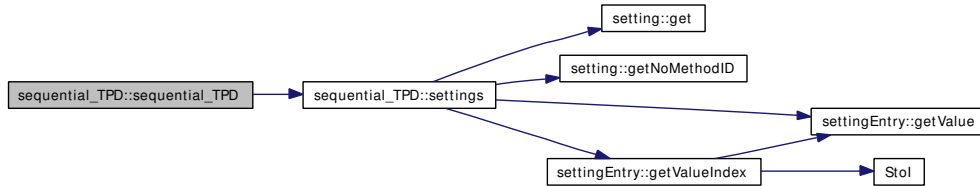
8.25.3 Constructor & Destructor Documentation

8.25.3.1 `sequential_TPD::sequential_TPD ()`

Definition at line 79 of file `truncation_point_detection.cc`.

References `INDEX`, `m_accu`, `m_actSample`, `m_meanSample`, `m_noReplication`, and `settings()`.

Here is the call graph for this function:



8.25.3.2 `sequential_TPD::~~sequential_TPD (void) [virtual]`

Definition at line 100 of file `truncation_point_detection.cc`.

8.25.4 Member Function Documentation

8.25.4.1 `INDEX sequential_TPD::getTruncationIndex (void) const`

Definition at line 103 of file `truncation_point_detection.cc`.

References `finished`, `m_state`, and `m_truncationPoint`.

8.25.4.2 `bool sequential_TPD::isReady (void) const [virtual]`

Reimplemented from `outputAnalyser` (p. 183).

Definition at line 108 of file `truncation_point_detection.cc`.

References `finished`, and `m_state`.

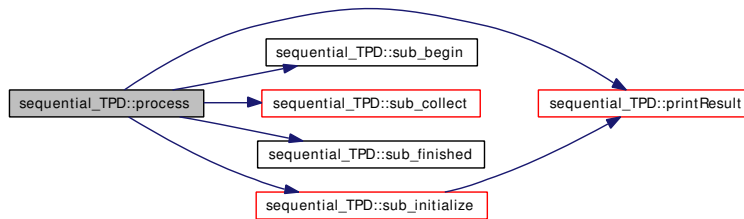
8.25.4.3 `void sequential_TPD::process (const std::list< CONTINUOUS > &)` [virtual]

Reimplemented from `outputAnalyser` (p. 183).

Definition at line 113 of file `truncation_point_detection.cc`.

References `begin`, `collect`, `exact`, `fast`, `finished`, `INDEX`, `initialize`, `logfile`, `m_actSample`, `m_limit`, `m_noReplication`, `m_performance`, `outputAnalyser::m_processedIndexes`, `m_state`, `m_truncationPoint`, `precise`, `printResult()`, `s_sequential_TPD`, `sub_begin()`, `sub_collect()`, `sub_finished()`, and `sub_initialize()`.

Here is the call graph for this function:



8.25.4.4 void sequential_TPD::printSetting (void) [virtual]

Reimplemented from **outputAnalyser** (p. 183).

Definition at line 147 of file truncation_point_detection.cc.

References exact, fast, ItoS(), logfile, m_alpha, m_limit, m_performance, m_ratio, m_ratio_max, m_ratio_min, precise, s_alpha, s_auto, s_exact, s_execute, s_fast, s_limit, s_performance, s_precise, s_ratio, s_ratio_max, s_ratio_min, s_sequential_TPD, and s_yes.

Here is the call graph for this function:



8.25.4.5 void sequential_TPD::printStatus (void) [virtual]

Reimplemented from **outputAnalyser** (p. 183).

Definition at line 174 of file truncation_point_detection.cc.

References begin, collect, exact, fast, finished, initialize, logfile, m_actBatch, m_actNoInBatch, m_alpha, m_batchSize, m_data_fast, m_data_initial, m_limit, m_nextSelection, m_noReplication, m_performance, outputAnalyser::m_processedIndexes, m_ratio, m_ratio_max, m_ratio_min, m_state, m_truncationPoint, m_varianceHomogeneityTest, precise, s_exact, s_fast, s_precise, and s_sequential_TPD.

Referenced by sub_compare().

8.25.4.6 void sequential_TPD::printResult (void) [virtual]

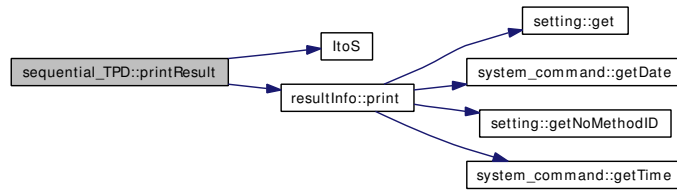
Reimplemented from **outputAnalyser** (p. 183).

Definition at line 211 of file truncation_point_detection.cc.

References begin, collect, exact, fast, finished, initialize, ItoS(), logfile, m_actBatch, m_actNoInBatch, m_alpha, m_batchSize, m_data_fast, m_data_initial, m_limit, m_nextSelection, m_noReplication, m_performance, outputAnalyser::m_processedIndexes, m_ratio, m_ratio_max, m_ratio_min, m_state, m_truncationPoint, m_varianceHomogeneityTest, precise, resultInfo::print(), resultfile, s_exact, s_fast, s_precise, and s_sequential_TPD.

Referenced by process(), sub_compare(), and sub_initialize().

Here is the call graph for this function:



8.25.4.7 void sequential_TPD::sub_begin (void) [protected]

Definition at line 251 of file truncation_point_detection.cc.

References INDEX, initialize, m_accu, m_actSample, m_data_initial, m_noReplication, m_performance, m_state, and precise.

Referenced by process().

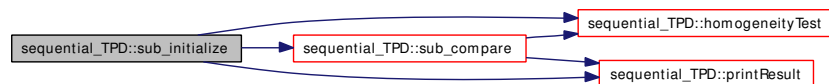
8.25.4.8 void sequential_TPD::sub_initialize (void) [protected]

Definition at line 258 of file truncation_point_detection.cc.

References fast, finished, homogeneityTest(), INDEX, m_accu, m_actSample, m_data_fast, m_data_initial, m_noReplication, m_performance, outputAnalyser::m_processedIndexes, m_ratio, m_ratio_max, m_ratio_min, m_state, m_truncationPoint, precise, printResult(), and sub_compare().

Referenced by process().

Here is the call graph for this function:



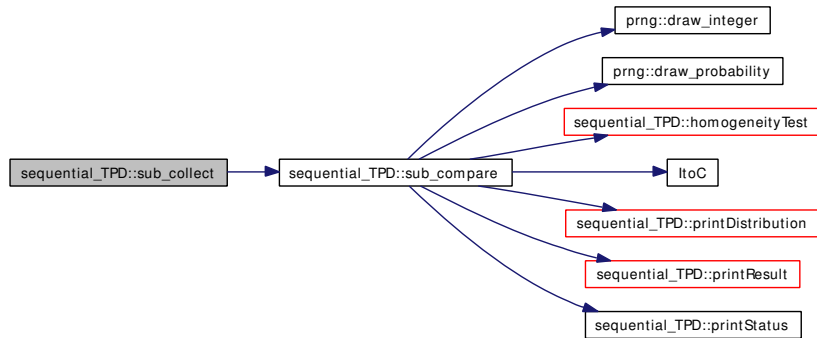
8.25.4.9 void sequential_TPD::sub_collect (void) [protected]

Definition at line 287 of file truncation_point_detection.cc.

References exact, fast, INDEX, m_accu, m_actBatch, m_actNoInBatch, m_actSample, m_batchSize, m_data_fast, m_data_initial, m_nextSelection, m_noReplication, m_performance, outputAnalyser::m_processedIndexes, m_ratio, precise, and sub_compare().

Referenced by process().

Here is the call graph for this function:



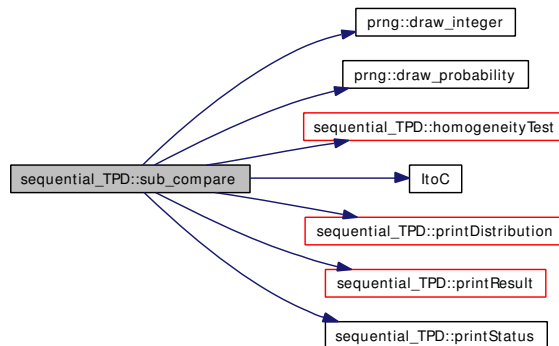
8.25.4.10 void sequential_TPD::sub_compare (void) [protected]

Definition at line 315 of file truncation_point_detection.cc.

References collect, prng::draw_integer(), prng::draw_probability(), exact, fast, finished, homogeneityTest(), INDEX, ItoC(), lib_prng, m_accu, m_actBatch, m_actNoInBatch, m_batchSize, m_data_fast, m_data_initial, m_meanSample, m_nextSelection, m_noReplication, m_performance, outputAnalyser::m_processedIndexes, m_ratio, m_state, m_truncationPoint, precise, printDistribution(), printResult(), and printStatus().

Referenced by sub_collect(), and sub_initialize().

Here is the call graph for this function:



8.25.4.11 void sequential_TPD::sub_finished (void) [protected]

Definition at line 418 of file truncation_point_detection.cc.

Referenced by process().

8.25.4.12 void sequential_TPD::settings (void) [protected]

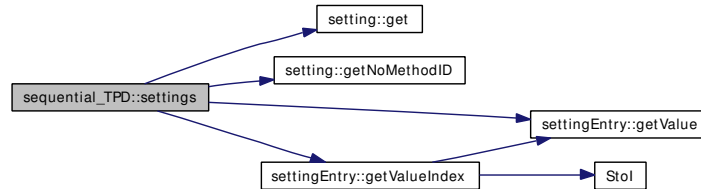
Definition at line 422 of file truncation_point_detection.cc.

References exact, fast, setting::get(), setting::getNoMethodID(), settingEntry::getValue(), settingEntry::getValueIndex(), lib_setting, m_alpha, m_limit, m_noReplication, m_performance, m_

ratio, m_ratio_max, m_ratio_min, precise, s_alpha, s_auto, s_exact, s_limit, s_performance, s_precise, s_ratio, s_ratio_max, s_ratio_min, s_replications, and s_sequential_TPD.

Referenced by sequential_TPD().

Here is the call graph for this function:



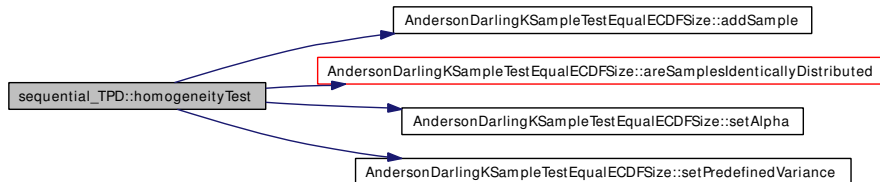
8.25.4.13 bool sequential_TPD::homogeneityTest (std::vector< CONTINUOUS > & p1, std::vector< CONTINUOUS > & p2) [protected]

Definition at line 475 of file truncation_point_detection.cc.

References AndersonDarlingKSampleTestEqualECDFSize::addSample(), AndersonDarlingKSampleTestEqualECDFSize::areSamplesIdenticallyDistributed(), CONTINUOUS, m_alpha, m_noReplication, m_varianceHomogeneityTest, AndersonDarlingKSampleTestEqualECDFSize::setAlpha(), and AndersonDarlingKSampleTestEqualECDFSize::setPredefinedVariance().

Referenced by sub_compare(), and sub_initialize().

Here is the call graph for this function:



8.25.4.14 void sequential_TPD::printDistribution (void) [protected]

Definition at line 493 of file truncation_point_detection.cc.

References exact, system_command::execute(), fast, INDEX, lib_system, m_data_fast, m_data_initial, m_performance, m_ratio, m_truncationPoint, precise, s_exact, s_fast, s_precise, and s_sequential_TPD.

Referenced by sub_compare().

Here is the call graph for this function:



8.25.4.15 TypeOfMethod `truncation_point_detection::getType (void) const` [virtual, inherited]

Reimplemented from `outputAnalyser` (p. 87).

Definition at line 14 of file `truncation_point_detection.cc`.

References IDENTICAL.

8.25.5 Field Documentation

8.25.5.1 `std::list< std::vector<CONTINUOUS> > sequential_TPD::m_data_initial` [protected]

Definition at line 46 of file `truncation_point_detection.h`.

Referenced by `printDistribution()`, `printResult()`, `printStatus()`, `sub_begin()`, `sub_collect()`, `sub_compare()`, and `sub_initialize()`.

8.25.5.2 `std::vector< std::vector<CONTINUOUS> > sequential_TPD::m_data_fast` [protected]

Definition at line 47 of file `truncation_point_detection.h`.

Referenced by `printDistribution()`, `printResult()`, `printStatus()`, `sub_collect()`, `sub_compare()`, and `sub_initialize()`.

8.25.5.3 `state sequential_TPD::m_state` [protected]

Definition at line 49 of file `truncation_point_detection.h`.

Referenced by `getTruncationIndex()`, `isReady()`, `printResult()`, `printStatus()`, `process()`, `sub_begin()`, `sub_compare()`, and `sub_initialize()`.

8.25.5.4 `performance sequential_TPD::m_performance` [protected]

Definition at line 50 of file `truncation_point_detection.h`.

Referenced by `printDistribution()`, `printResult()`, `printSetting()`, `printStatus()`, `process()`, `settings()`, `sub_begin()`, `sub_collect()`, `sub_compare()`, and `sub_initialize()`.

8.25.5.5 `INDEX sequential_TPD::m_truncationPoint` [protected]

Definition at line 51 of file `truncation_point_detection.h`.

Referenced by `getTruncationIndex()`, `printDistribution()`, `printResult()`, `printStatus()`, `process()`, `sub_compare()`, and `sub_initialize()`.

8.25.5.6 `INDEX sequential_TPD::m_ratio` [protected]

Definition at line 52 of file `truncation_point_detection.h`.

Referenced by `printDistribution()`, `printResult()`, `printSetting()`, `printStatus()`, `settings()`, `sub_collect()`, `sub_compare()`, and `sub_initialize()`.

8.25.5.7 INDEX sequential_TPD::m_ratio_min [protected]

Definition at line 53 of file truncation_point_detection.h.

Referenced by printResult(), printSetting(), printStatus(), settings(), and sub_initialize().

8.25.5.8 INDEX sequential_TPD::m_ratio_max [protected]

Definition at line 54 of file truncation_point_detection.h.

Referenced by printResult(), printSetting(), printStatus(), settings(), and sub_initialize().

8.25.5.9 INDEX sequential_TPD::m_noReplication [protected]

Definition at line 55 of file truncation_point_detection.h.

Referenced by homogeneityTest(), printResult(), printStatus(), process(), sequential_TPD(), settings(), sub_begin(), sub_collect(), sub_compare(), and sub_initialize().

8.25.5.10 INDEX sequential_TPD::m_limit [protected]

Definition at line 56 of file truncation_point_detection.h.

Referenced by printResult(), printSetting(), printStatus(), process(), and settings().

8.25.5.11 CONTINUOUS sequential_TPD::m_alpha [protected]

Definition at line 57 of file truncation_point_detection.h.

Referenced by homogeneityTest(), printResult(), printSetting(), printStatus(), and settings().

8.25.5.12 CONTINUOUS sequential_TPD::m_varianceHomogeneityTest [protected]

Definition at line 58 of file truncation_point_detection.h.

Referenced by homogeneityTest(), printResult(), and printStatus().

8.25.5.13 std::vector<CONTINUOUS> sequential_TPD::m_actSample [protected]

Definition at line 59 of file truncation_point_detection.h.

Referenced by process(), sequential_TPD(), sub_begin(), sub_collect(), and sub_initialize().

8.25.5.14 INDEX sequential_TPD::m_batchSize [protected]

Definition at line 73 of file truncation_point_detection.h.

Referenced by printResult(), printStatus(), sub_collect(), and sub_compare().

8.25.5.15 INDEX `sequential_TPD::m_actBatch` [protected]

Definition at line 74 of file `truncation_point_detection.h`.

Referenced by `printResult()`, `printStatus()`, `sub_collect()`, and `sub_compare()`.

8.25.5.16 INDEX `sequential_TPD::m_actNoInBatch` [protected]

Definition at line 75 of file `truncation_point_detection.h`.

Referenced by `printResult()`, `printStatus()`, `sub_collect()`, and `sub_compare()`.

8.25.5.17 INDEX `sequential_TPD::m_nextSelection` [protected]

Definition at line 76 of file `truncation_point_detection.h`.

Referenced by `printResult()`, `printStatus()`, `sub_collect()`, and `sub_compare()`.

8.25.5.18 `std::vector<CONTINUOUS> sequential_TPD::m_accu` [protected]

Definition at line 79 of file `truncation_point_detection.h`.

Referenced by `sequential_TPD()`, `sub_begin()`, `sub_collect()`, `sub_compare()`, and `sub_initialize()`.

8.25.5.19 `std::vector<CONTINUOUS> sequential_TPD::m_meanSample` [protected]

Definition at line 80 of file `truncation_point_detection.h`.

Referenced by `sequential_TPD()`, and `sub_compare()`.

8.25.5.20 INDEX `outputAnalyser::m_processedIndexes` [protected, inherited]

Definition at line 20 of file `basic.h`.

Referenced by `evolution::calculateQuantiles()`, `spectral_analysis_QE::checkQuantiles()`, `batch_mean_QE::checkQuantiles()`, `pooling_QE::checkQuantiles()`, `deterministic_TPD::isReady()`, `evolution::isReady()`, `printResult()`, `deterministic_TPD::printResult()`, `spectral_analysis_QE::printResult()`, `batch_mean_QE::printResult()`, `pooling_QE::printResult()`, `batching::printResult()`, `printStatus()`, `deterministic_TPD::printStatus()`, `evolution::printStatus()`, `spectral_analysis_QE::printStatus()`, `batch_mean_QE::printStatus()`, `pooling_QE::printStatus()`, `batching::printStatus()`, `process()`, `deterministic_TPD::process()`, `evolution::process()`, `spectral_analysis_QE::process()`, `batch_mean_QE::process()`, `pooling_QE::process()`, `batching::process()`, `outputAnalyser::process()`, `sub_collect()`, `sub_compare()`, and `sub_initialize()`.

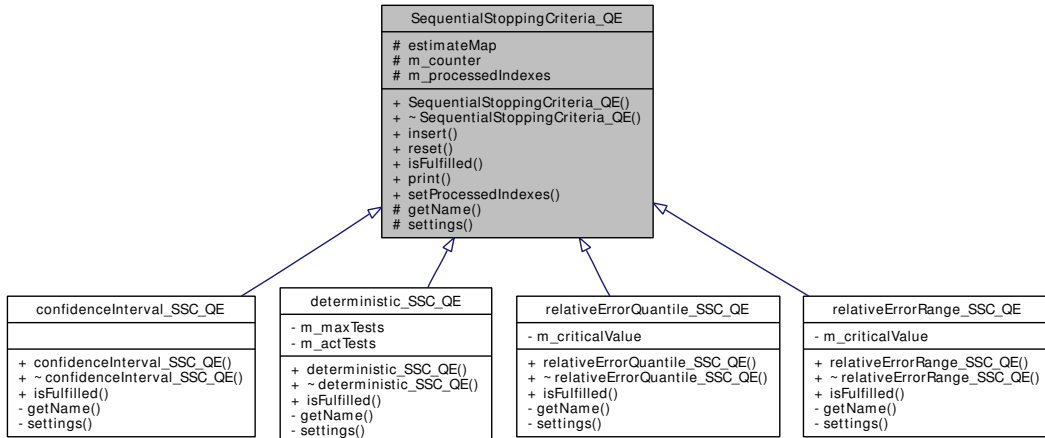
The documentation for this class was generated from the following files:

- `truncation_point_detection.h`
- `truncation_point_detection.cc`

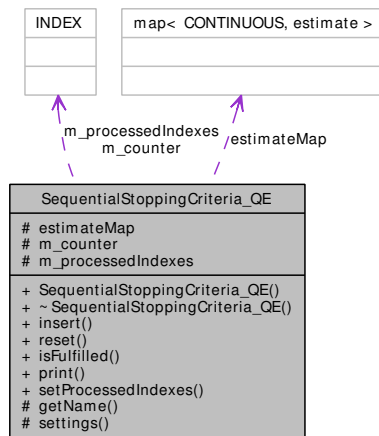
8.26 SequentialStoppingCriteria_QE Class Reference

```
#include <quantile_estimation.h>
```

Inheritance diagram for SequentialStoppingCriteria_QE:



Collaboration diagram for SequentialStoppingCriteria_QE:



Public Member Functions

- **SequentialStoppingCriteria_QE** (void)
- virtual **~SequentialStoppingCriteria_QE** (void)
- void **insert** (const CONTINUOUS &location, const CONTINUOUS &probability, const CONTINUOUS &absoluteErrorNeg, const CONTINUOUS &absoluteErrorPos)
- void **reset** (void)
- virtual bool **isFulfilled** (void)
- void **print** (bool isFinal=false)
- void **setProcessedIndexes** (INDEX i)

Protected Member Functions

- virtual std::string **getName** (void)
- virtual void **settings** (void)

Protected Attributes

- std::map< CONTINUOUS, estimate > **estimateMap**
- INDEX **m_counter**
- INDEX **m_processedIndexes**

Data Structures

- struct **estimate**

8.26.1 Detailed Description

Definition at line 106 of file `quantile_estimation.h`.

8.26.2 Constructor & Destructor Documentation

8.26.2.1 SequentialStoppingCriteria_QE::SequentialStoppingCriteria_QE (void)

Definition at line 619 of file `quantile_estimation.cc`.

8.26.2.2 SequentialStoppingCriteria_QE::~~SequentialStoppingCriteria_QE (void) [virtual]

Definition at line 624 of file `quantile_estimation.cc`.

8.26.3 Member Function Documentation

8.26.3.1 void SequentialStoppingCriteria_QE::insert (const CONTINUOUS & *location*, const CONTINUOUS & *probability*, const CONTINUOUS & *absoluteErrorNeg*, const CONTINUOUS & *absoluteErrorPos*)

Definition at line 627 of file `quantile_estimation.cc`.

References `SequentialStoppingCriteria_QE::estimate::absoluteErrorNeg`, `SequentialStoppingCriteria_QE::estimate::absoluteErrorPos`, `estimateMap`, `SequentialStoppingCriteria_QE::estimate::location`, and `SequentialStoppingCriteria_QE::estimate::probability`.

Referenced by `spectral_analysis_QE::checkQuantiles()`, `batch_mean_QE::checkQuantiles()`, and `pooling_QE::checkQuantiles()`.

8.26.3.2 void SequentialStoppingCriteria_QE::reset (void)

Definition at line 639 of file `quantile_estimation.cc`.

References `estimateMap`.

Referenced by `spectral_analysis_QE::checkQuantiles()`, `batch_mean_QE::checkQuantiles()`, and `pooling_QE::checkQuantiles()`.

8.26.3.3 `bool SequentialStoppingCriteria_QE::isFulfilled (void) [virtual]`

Reimplemented in `deterministic_SSC_QE` (p. 55), `confidenceInterval_SSC_QE` (p. 47), `relativeErrorQuantile_SSC_QE` (p. 116), and `relativeErrorRange_SSC_QE` (p. 121).

Definition at line 643 of file `quantile_estimation.cc`.

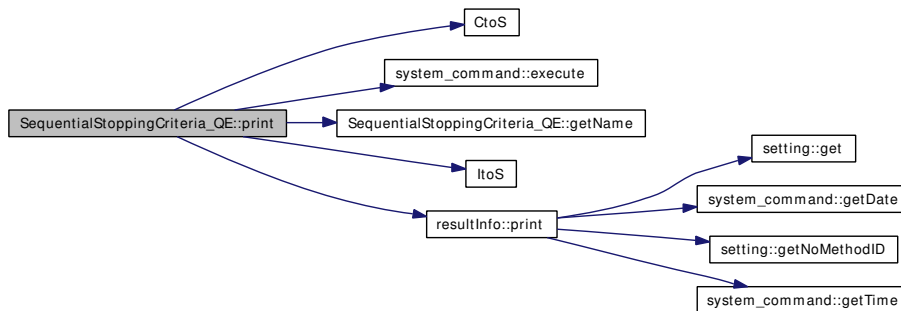
Referenced by `spectral_analysis_QE::checkQuantiles()`, `batch_mean_QE::checkQuantiles()`, and `pooling_QE::checkQuantiles()`.

8.26.3.4 `void SequentialStoppingCriteria_QE::print (bool isFinal = false)`

Definition at line 652 of file `quantile_estimation.cc`.

References `CONTINUOUS`, `CtoS()`, `estimateMap`, `system_command::execute()`, `getName()`, `ItoS()`, `lib_system`, `m_counter`, `m_processedIndexes`, `resultInfo::print()`, and `resultfile`.

Here is the call graph for this function:



8.26.3.5 `void SequentialStoppingCriteria_QE::setProcessedIndexes (INDEX i) [inline]`

Definition at line 118 of file `quantile_estimation.h`.

References `m_processedIndexes`.

Referenced by `spectral_analysis_QE::checkQuantiles()`, `batch_mean_QE::checkQuantiles()`, and `pooling_QE::checkQuantiles()`.

8.26.3.6 `virtual std::string SequentialStoppingCriteria_QE::getName (void) [inline, protected, virtual]`

Reimplemented in `deterministic_SSC_QE` (p. 56), `confidenceInterval_SSC_QE` (p. 47), `relativeErrorQuantile_SSC_QE` (p. 117), and `relativeErrorRange_SSC_QE` (p. 121).

Definition at line 128 of file `quantile_estimation.h`.

Referenced by `print()`.

8.26.3.7 void SequentialStoppingCriteria_QE::settings (void) [protected, virtual]

Reimplemented in `deterministic_SSC_QE` (p. 56), `confidenceInterval_SSC_QE` (p. 48), `relativeErrorQuantile_SSC_QE` (p. 117), and `relativeErrorRange_SSC_QE` (p. 122).

Definition at line 648 of file `quantile_estimation.cc`.

8.26.4 Field Documentation

8.26.4.1 std::map<CONTINUOUS,estimate> SequentialStoppingCriteria_QE::estimateMap [protected]

Definition at line 131 of file `quantile_estimation.h`.

Referenced by `insert()`, `print()`, and `reset()`.

8.26.4.2 INDEX SequentialStoppingCriteria_QE::m_counter [protected]

Definition at line 132 of file `quantile_estimation.h`.

Referenced by `print()`.

8.26.4.3 INDEX SequentialStoppingCriteria_QE::m_processedIndexes [protected]

Definition at line 133 of file `quantile_estimation.h`.

Referenced by `print()`, and `setProcessedIndexes()`.

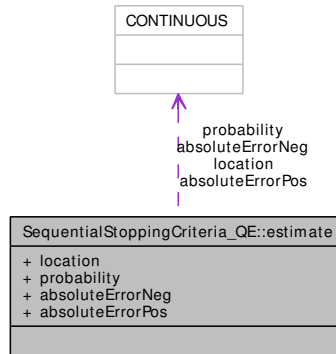
The documentation for this class was generated from the following files:

- `quantile_estimation.h`
- `quantile_estimation.cc`

8.27 SequentialStoppingCriteria_QE::estimate Struct Reference

```
#include <quantile_estimation.h>
```

Collaboration diagram for SequentialStoppingCriteria_QE::estimate:



Data Fields

- CONTINUOUS **location**
- CONTINUOUS **probability**
- CONTINUOUS **absoluteErrorNeg**
- CONTINUOUS **absoluteErrorPos**

8.27.1 Detailed Description

Definition at line 121 of file `quantile_estimation.h`.

8.27.2 Field Documentation

8.27.2.1 CONTINUOUS SequentialStoppingCriteria_QE::estimate::location

Definition at line 122 of file `quantile_estimation.h`.

Referenced by `SequentialStoppingCriteria_QE::insert()`.

8.27.2.2 CONTINUOUS SequentialStoppingCriteria_QE::estimate::probability

Definition at line 123 of file `quantile_estimation.h`.

Referenced by `SequentialStoppingCriteria_QE::insert()`.

8.27.2.3 CONTINUOUS SequentialStoppingCriteria_QE::estimate::absoluteErrorNeg

Definition at line 124 of file `quantile_estimation.h`.

Referenced by `SequentialStoppingCriteria_QE::insert()`.

8.27.2.4 CONTINUOUS SequentialStoppingCriteria_QE::estimate::absoluteError-Pos

Definition at line 125 of file `quantile_estimation.h`.

Referenced by `SequentialStoppingCriteria_QE::insert()`.

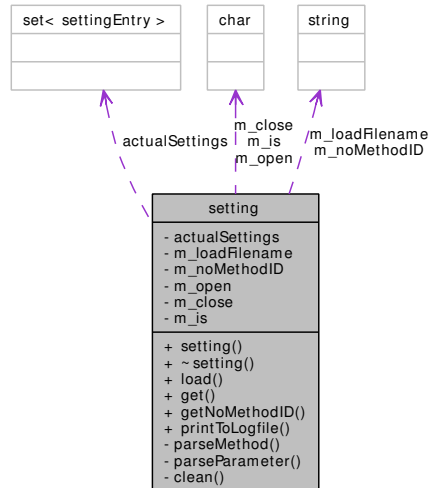
The documentation for this struct was generated from the following file:

- `quantile_estimation.h`

8.28 setting Class Reference

```
#include <setting.h>
```

Collaboration diagram for setting:



Public Member Functions

- **setting** (void)
- **~setting** (void)
- void **load** (const std::string &)
- bool **get** (settingEntry &)
- const std::string & **getNoMethodID** (void) const
- void **printToLogfile** (void)

Private Member Functions

- bool **parseMethod** (const std::string &, std::string &)
- bool **parseParameter** (const std::string &, std::string &, std::string &)
- void **clean** (std::string &)

Private Attributes

- std::set< settingEntry > **actualSettings**
- std::string **m_loadFilename**
- std::string **m_noMethodID**
- char **m_open**
- char **m_close**
- char **m_is**

8.28.1 Detailed Description

Definition at line 39 of file setting.h.

8.28.2 Constructor & Destructor Documentation

8.28.2.1 setting::setting (void)

Definition at line 95 of file setting.cc.

8.28.2.2 setting::~~setting (void)

Definition at line 103 of file setting.cc.

8.28.3 Member Function Documentation

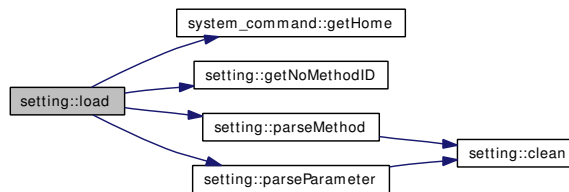
8.28.3.1 void setting::load (const std::string &)

Definition at line 106 of file setting.cc.

References `actualSettings`, `system_command::getHome()`, `getNoMethodID()`, `lib_system`, `m_loadFilename`, `parseMethod()`, and `parseParameter()`.

Referenced by `main()`.

Here is the call graph for this function:



8.28.3.2 bool setting::get (settingEntry &)

Definition at line 146 of file setting.cc.

References `actualSettings`.

Referenced by `deterministic_TPD::deterministic_TPD()`, `evolution::evolution()`, `main()`, `method_factory::method_factory()`, `resultInfo::print()`, `quantile_estimation::set_SSC()`, `sequential_TPD::settings()`, `spectral_analysis_QE::settings()`, `batch_mean_QE::settings()`, `pooling_QE::settings()`, and `batching::settings()`.

8.28.3.3 const std::string & setting::getNoMethodID (void) const

Definition at line 153 of file setting.cc.

References `m_noMethodID`.

Referenced by `evolution::evolution()`, `load()`, `main()`, `resultInfo::print()`, `sequential_TPD::settings()`, `spectral_analysis_QE::settings()`, `batch_mean_QE::settings()`, and `batching::settings()`.

8.28.3.4 void setting::printToLogfile (void)

Definition at line 157 of file setting.cc.

References actualSettings, logfile, m_close, m_is, m_loadFilename, and m_open.

Referenced by main().

8.28.3.5 bool setting::parseMethod (const std::string &, std::string &) [private]

Definition at line 179 of file setting.cc.

References clean(), m_close, and m_open.

Referenced by load().

Here is the call graph for this function:

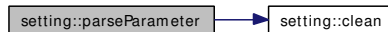
**8.28.3.6 bool setting::parseParameter (const std::string &, std::string &, std::string &) [private]**

Definition at line 192 of file setting.cc.

References clean(), and m_is.

Referenced by load().

Here is the call graph for this function:

**8.28.3.7 void setting::clean (std::string &) [private]**

Definition at line 208 of file setting.cc.

Referenced by parseMethod(), and parseParameter().

8.28.4 Field Documentation**8.28.4.1 std::set<settingEntry> setting::actualSettings [private]**

Definition at line 51 of file setting.h.

Referenced by get(), load(), and printToLogfile().

8.28.4.2 std::string setting::m_loadFilename [private]

Definition at line 53 of file setting.h.

Referenced by load(), and printToLogfile().

8.28.4.3 `std::string setting::m_noMethodID` [private]

Definition at line 54 of file setting.h.

Referenced by `getNoMethodID()`.

8.28.4.4 `char setting::m_open` [private]

Definition at line 55 of file setting.h.

Referenced by `parseMethod()`, and `printToLogfile()`.

8.28.4.5 `char setting::m_close` [private]

Definition at line 56 of file setting.h.

Referenced by `parseMethod()`, and `printToLogfile()`.

8.28.4.6 `char setting::m_is` [private]

Definition at line 57 of file setting.h.

Referenced by `parseParameter()`, and `printToLogfile()`.

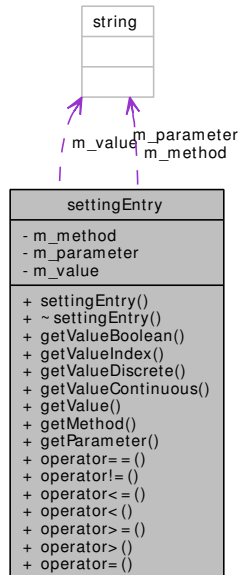
The documentation for this class was generated from the following files:

- `setting.h`
- `setting.cc`

8.29 settingEntry Class Reference

```
#include <setting.h>
```

Collaboration diagram for settingEntry:



Public Member Functions

- **settingEntry** (const std::string &, const std::string &, const std::string &)
- **~settingEntry** ()
- bool **getValueBoolean** (void) const
- INDEX **getValueIndex** (void) const
- DISCRETE **getValueDiscrete** (void) const
- CONTINUOUS **getValueContinuous** (void) const
- const std::string & **getValue** (void) const
- const std::string & **getMethod** (void) const
- const std::string & **getParameter** (void) const
- bool **operator==** (const **settingEntry** &) const
- bool **operator!=** (const **settingEntry** &) const
- bool **operator<=** (const **settingEntry** &) const
- bool **operator<** (const **settingEntry** &) const
- bool **operator>=** (const **settingEntry** &) const
- bool **operator>** (const **settingEntry** &) const
- const **settingEntry** & **operator=** (const **settingEntry** &)

Private Attributes

- const std::string **m_method**
- const std::string **m_parameter**
- std::string **m_value**

8.29.1 Detailed Description

Definition at line 11 of file setting.h.

8.29.2 Constructor & Destructor Documentation

8.29.2.1 settingEntry::settingEntry (const std::string &, const std::string &, const std::string &)

Definition at line 13 of file setting.cc.

8.29.2.2 settingEntry::~~settingEntry ()

Definition at line 21 of file setting.cc.

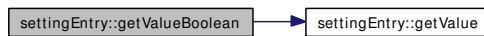
8.29.3 Member Function Documentation

8.29.3.1 bool settingEntry::getValueBoolean (void) const

Definition at line 24 of file setting.cc.

References `getValue()`.

Here is the call graph for this function:



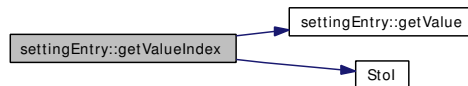
8.29.3.2 INDEX settingEntry::getValueIndex (void) const

Definition at line 29 of file setting.cc.

References `getValue()`, and `StoI()`.

Referenced by `deterministic_TPD::deterministic_TPD()`, `evolution::evolution()`, `sequential_TPD::settings()`, `spectral_analysis_QE::settings()`, `batch_mean_QE::settings()`, `pooling_QE::settings()`, and `batching::settings()`.

Here is the call graph for this function:

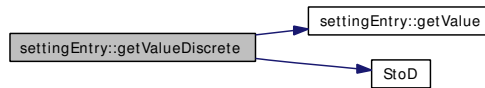


8.29.3.3 DISCRETE settingEntry::getValueDiscrete (void) const

Definition at line 33 of file setting.cc.

References `getValue()`, and `StoD()`.

Here is the call graph for this function:



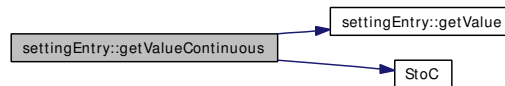
8.29.3.4 CONTINUOUS `settingEntry::getValueContinuous (void) const`

Definition at line 37 of file `setting.cc`.

References `getValue()`, and `StoC()`.

Referenced by `evolution::evolution()`, `spectral_analysis_QE::settings()`, `batch_mean_QE::settings()`, `pooling_QE::settings()`, and `batching::settings()`.

Here is the call graph for this function:



8.29.3.5 `const std::string & settingEntry::getValue (void) const`

Definition at line 41 of file `setting.cc`.

References `m_value`.

Referenced by `getValueBoolean()`, `getValueContinuous()`, `getValueDiscrete()`, `getValueIndex()`, `method_factory::method_factory()`, `quantile_estimation::set_SSC()`, `sequential_TPD::settings()`, and `batching::settings()`.

8.29.3.6 `const std::string & settingEntry::getMethod (void) const`

Definition at line 45 of file `setting.cc`.

References `m_method`.

Referenced by `method_factory::method_factory()`.

8.29.3.7 `const std::string & settingEntry::getParameter (void) const`

Definition at line 49 of file `setting.cc`.

References `m_parameter`.

8.29.3.8 `bool settingEntry::operator==(const settingEntry &) const`

Definition at line 53 of file `setting.cc`.

References `m_method`, and `m_parameter`.

8.29.3.9 `bool settingEntry::operator!=(const settingEntry &) const`

Definition at line 57 of file setting.cc.

References `m_method`, and `m_parameter`.

8.29.3.10 `bool settingEntry::operator<=(const settingEntry &) const`

Definition at line 61 of file setting.cc.

References `m_method`, and `m_parameter`.

8.29.3.11 `bool settingEntry::operator<(const settingEntry &) const`

Definition at line 68 of file setting.cc.

References `m_method`, and `m_parameter`.

8.29.3.12 `bool settingEntry::operator>=(const settingEntry &) const`

Definition at line 75 of file setting.cc.

References `m_method`, and `m_parameter`.

8.29.3.13 `bool settingEntry::operator>(const settingEntry &) const`

Definition at line 82 of file setting.cc.

References `m_method`, and `m_parameter`.

8.29.3.14 `const settingEntry & settingEntry::operator=(const settingEntry &)`

Definition at line 89 of file setting.cc.

References `m_value`.

8.29.4 Field Documentation**8.29.4.1** `const std::string settingEntry::m_method [private]`

Definition at line 34 of file setting.h.

Referenced by `getMethod()`, `operator!=()`, `operator<()`, `operator<=()`, `operator==()`, `operator>()`, and `operator>=()`.

8.29.4.2 `const std::string settingEntry::m_parameter [private]`

Definition at line 35 of file setting.h.

Referenced by `getParameter()`, `operator!=()`, `operator<()`, `operator<=()`, `operator==()`, `operator>()`, and `operator>=()`.

8.29.4.3 `std::string settingEntry::m_value` [private]

Definition at line 36 of file `setting.h`.

Referenced by `getValue()`, and `operator=()`.

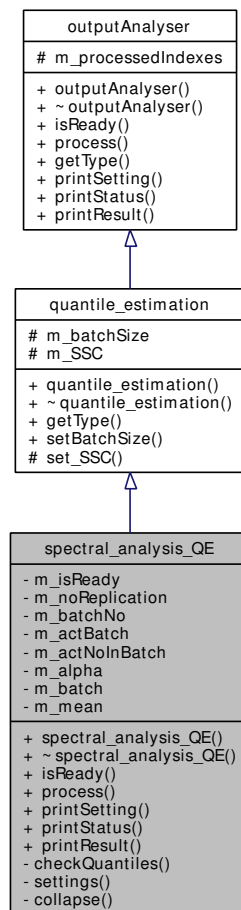
The documentation for this class was generated from the following files:

- `setting.h`
- `setting.cc`

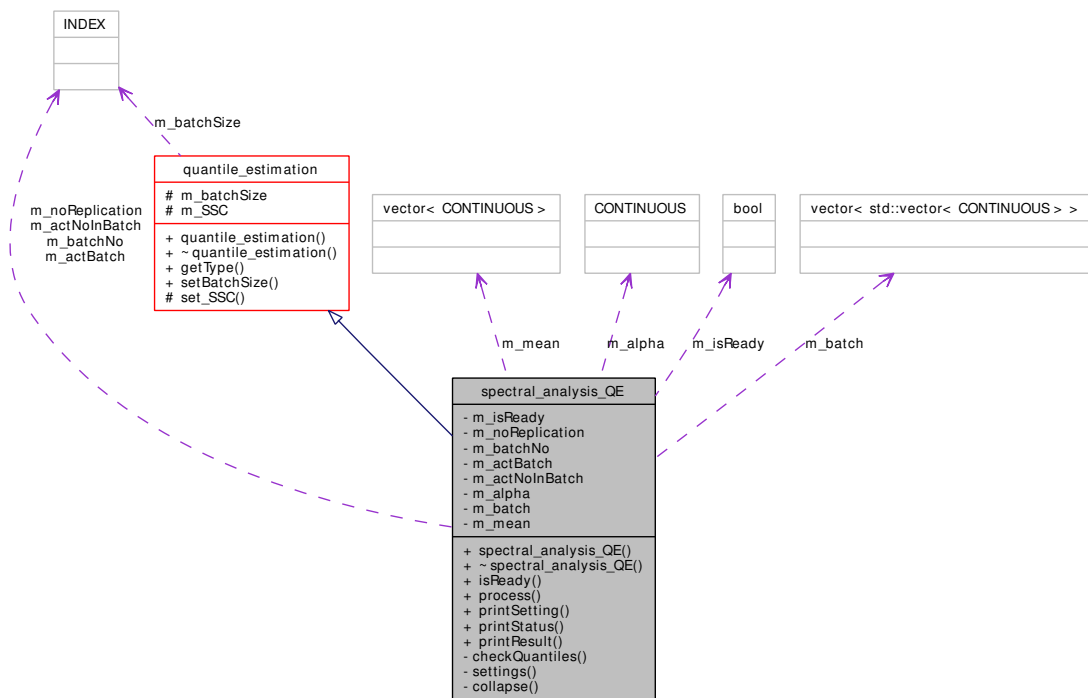
8.30 spectral_analysis_QE Class Reference

```
#include <quantile_estimation.h>
```

Inheritance diagram for spectral_analysis_QE:



Collaboration diagram for spectral_analysis_QE:



Public Member Functions

- **spectral_analysis_QE** (void)
- **~spectral_analysis_QE** (void)
- **bool isReady** (void) const
- **void process** (const std::list< CONTINUOUS > &)
- **void printSetting** (void)
- **void printStatus** (void)
- **void printResult** (void)
- **virtual TypeOfMethod getType** (void) const
- **void setBatchSize** (INDEX p)

Protected Member Functions

- **void set_SSC** (void)

Protected Attributes

- INDEX **m_batchSize**
- SequentialStoppingCriteria_QE * **m_SSC**
- INDEX **m_processedIndexes**

Private Member Functions

- **bool checkQuantiles** (void)
- **void settings** ()
- **void collapse** (void)

Private Attributes

- bool `m_isReady`
- INDEX `m_noReplication`
- INDEX `m_batchNo`
- INDEX `m_actBatch`
- INDEX `m_actNoInBatch`
- CONTINUOUS `m_alpha`
- `std::vector< std::vector< CONTINUOUS > >` `m_batch`
- `std::vector< CONTINUOUS >` `m_mean`

8.30.1 Detailed Description

Definition at line 80 of file `quantile_estimation.h`.

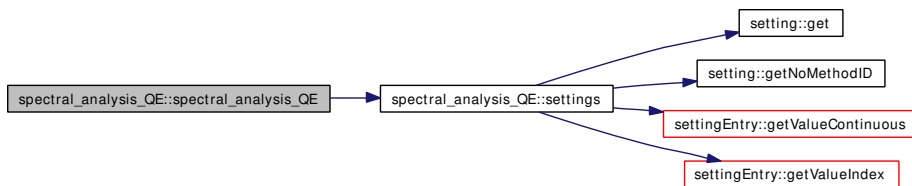
8.30.2 Constructor & Destructor Documentation

8.30.2.1 `spectral_analysis_QE::spectral_analysis_QE (void)`

Definition at line 420 of file `quantile_estimation.cc`.

References `settings()`.

Here is the call graph for this function:



8.30.2.2 `spectral_analysis_QE::~~spectral_analysis_QE (void)`

Definition at line 431 of file `quantile_estimation.cc`.

8.30.3 Member Function Documentation

8.30.3.1 `bool spectral_analysis_QE::isReady (void) const [virtual]`

Reimplemented from `outputAnalyser` (p. 183).

Definition at line 434 of file `quantile_estimation.cc`.

References `m_isReady`.

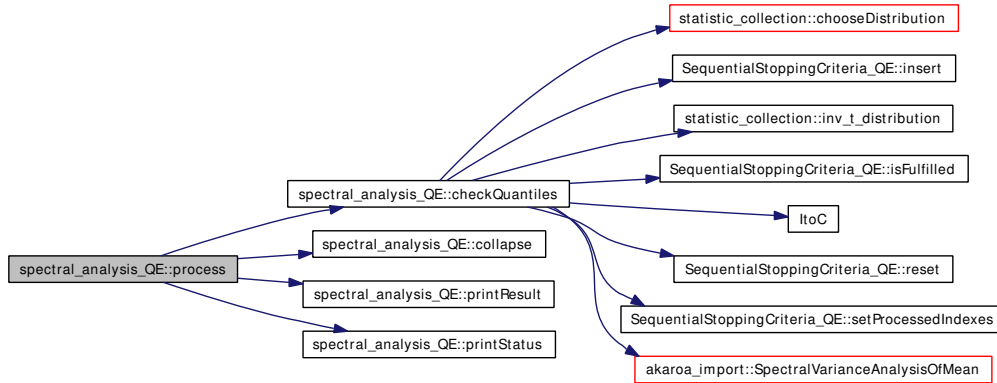
8.30.3.2 `void spectral_analysis_QE::process (const std::list< CONTINUOUS > &) [virtual]`

Reimplemented from `outputAnalyser` (p. 183).

Definition at line 438 of file `quantile_estimation.cc`.

References `checkQuantiles()`, `collapse()`, `INDEX`, `m_actBatch`, `m_actNoInBatch`, `m_batch`, `m_batchNo`, `quantile_estimation::m_batchSize`, `m_isReady`, `m_mean`, `m_noReplication`, `outputAnalyser::m_processedIndexes`, `printResult()`, and `printStatus()`.

Here is the call graph for this function:



8.30.3.3 void spectral_analysis_QE::printSetting (void) [virtual]

Reimplemented from `outputAnalyser` (p. 183).

Definition at line 488 of file `quantile_estimation.cc`.

References `logfile`, `m_batchNo`, `s_batches`, `s_execute`, `s_spectral_analysis_QE`, and `s_yes`.

8.30.3.4 void spectral_analysis_QE::printStatus (void) [virtual]

Reimplemented from `outputAnalyser` (p. 183).

Definition at line 499 of file `quantile_estimation.cc`.

References `logfile`, `m_actBatch`, `m_actNoInBatch`, `m_alpha`, `m_batchNo`, `quantile_estimation::m_batchSize`, `m_noReplication`, `outputAnalyser::m_processedIndexes`, and `s_spectral_analysis_QE`.

Referenced by `process()`.

8.30.3.5 void spectral_analysis_QE::printResult (void) [virtual]

Reimplemented from `outputAnalyser` (p. 183).

Definition at line 511 of file `quantile_estimation.cc`.

References `logfile`, `m_actBatch`, `m_actNoInBatch`, `m_alpha`, `m_batchNo`, `quantile_estimation::m_batchSize`, `m_noReplication`, `outputAnalyser::m_processedIndexes`, and `s_spectral_analysis_QE`.

Referenced by `process()`.

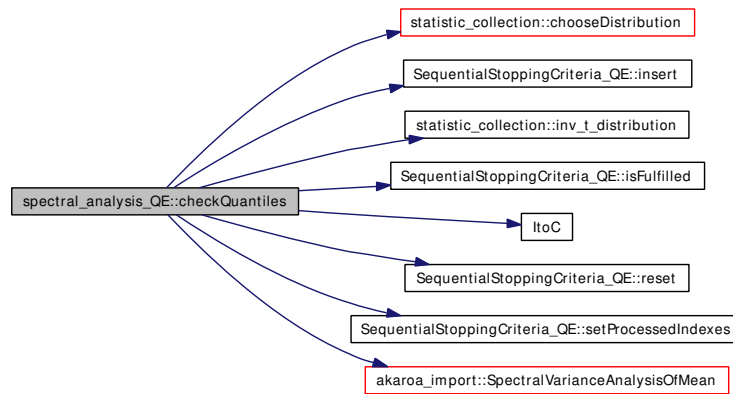
8.30.3.6 bool spectral_analysis_QE::checkQuantiles (void) [private]

Definition at line 523 of file quantile_estimation.cc.

References `statistic_collection::chooseDistribution()`, `CONTINUOUS`, `INDEX`, `SequentialStoppingCriteria_QE::insert()`, `statistic_collection::inv_t_distribution()`, `SequentialStoppingCriteria_QE::isFulfilled()`, `ItoC()`, `lib_akaroa`, `lib_statistic`, `m_alpha`, `m_batch`, `m_batchNo`, `quantile_estimation::m_batchSize`, `m_mean`, `m_noReplication`, `outputAnalyser::m_processedIndexes`, `quantile_estimation::m_SSC`, `SequentialStoppingCriteria_QE::reset()`, `SequentialStoppingCriteria_QE::setProcessedIndexes()`, and `akaroa_import::SpectralVarianceAnalysisOfMean()`.

Referenced by `process()`.

Here is the call graph for this function:



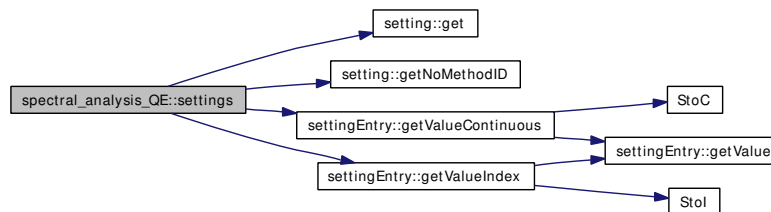
8.30.3.7 void spectral_analysis_QE::settings () [private]

Definition at line 574 of file quantile_estimation.cc.

References `setting::get()`, `setting::getNoMethodID()`, `settingEntry::getValueContinuous()`, `settingEntry::getValueIndex()`, `lib_setting`, `m_alpha`, `m_batchNo`, `m_noReplication`, `s_alpha`, `s_batches`, `s_replications`, and `s_spectral_analysis_QE`.

Referenced by `spectral_analysis_QE()`.

Here is the call graph for this function:



8.30.3.8 void spectral_analysis_QE::collapse (void) [private]

Definition at line 597 of file quantile_estimation.cc.

References INDEX, m_actBatch, m_actNoInBatch, m_batch, m_batchNo, quantile_estimation::m_batchSize, and m_noReplication.

Referenced by process().

8.30.3.9 TypeOfMethod quantile_estimation::getType (void) const [virtual, inherited]

Reimplemented from **outputAnalyser** (p. 87).

Definition at line 22 of file quantile_estimation.cc.

References ESTIMATOR.

8.30.3.10 void quantile_estimation::setBatchSize (INDEX p) [inherited]

Definition at line 26 of file quantile_estimation.cc.

References quantile_estimation::m_batchSize.

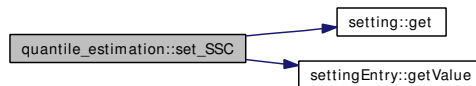
8.30.3.11 void quantile_estimation::set_SSC (void) [protected, inherited]

Definition at line 31 of file quantile_estimation.cc.

References setting::get(), settingEntry::getValue(), lib_setting, quantile_estimation::m_SSC, s_confidenceInterval_SSC_QE, s_deterministic_SSC_QE, s_execute, s_relativeErrorQuantile_SSC_QE, s_relativeErrorRange_SSC_QE, and s_yes.

Referenced by quantile_estimation::quantile_estimation().

Here is the call graph for this function:



8.30.4 Field Documentation

8.30.4.1 bool spectral_analysis_QE::m_isReady [private]

Definition at line 96 of file quantile_estimation.h.

Referenced by isReady(), and process().

8.30.4.2 INDEX spectral_analysis_QE::m_noReplication [private]

Definition at line 97 of file quantile_estimation.h.

Referenced by checkQuantiles(), collapse(), printResult(), printStatus(), process(), and settings().

8.30.4.3 INDEX spectral_analysis_QE::m_batchNo [private]

Definition at line 98 of file quantile_estimation.h.

Referenced by `checkQuantiles()`, `collapse()`, `printResult()`, `printSetting()`, `printStatus()`, `process()`, and `settings()`.

8.30.4.4 INDEX `spectral_analysis_QE::m_actBatch` [private]

Definition at line 99 of file `quantile_estimation.h`.

Referenced by `collapse()`, `printResult()`, `printStatus()`, and `process()`.

8.30.4.5 INDEX `spectral_analysis_QE::m_actNoInBatch` [private]

Definition at line 100 of file `quantile_estimation.h`.

Referenced by `collapse()`, `printResult()`, `printStatus()`, and `process()`.

8.30.4.6 CONTINUOUS `spectral_analysis_QE::m_alpha` [private]

Definition at line 101 of file `quantile_estimation.h`.

Referenced by `checkQuantiles()`, `printResult()`, `printStatus()`, and `settings()`.

8.30.4.7 `std::vector< std::vector<CONTINUOUS> > spectral_analysis_QE::m_batch` [private]

Definition at line 102 of file `quantile_estimation.h`.

Referenced by `checkQuantiles()`, `collapse()`, and `process()`.

8.30.4.8 `std::vector< CONTINUOUS > spectral_analysis_QE::m_mean` [private]

Definition at line 103 of file `quantile_estimation.h`.

Referenced by `checkQuantiles()`, and `process()`.

8.30.4.9 INDEX `quantile_estimation::m_batchSize` [protected, inherited]

Definition at line 26 of file `quantile_estimation.h`.

Referenced by `checkQuantiles()`, `batch_mean_QE::checkQuantiles()`, `collapse()`, `batch_mean_QE::collapse()`, `printResult()`, `batch_mean_QE::printResult()`, `pooling_QE::printResult()`, `printStatus()`, `batch_mean_QE::printStatus()`, `pooling_QE::printStatus()`, `process()`, `batch_mean_QE::process()`, `pooling_QE::process()`, and `quantile_estimation::setBatchSize()`.

8.30.4.10 `SequentialStoppingCriteria_QE* quantile_estimation::m_SSC` [protected, inherited]

Definition at line 27 of file `quantile_estimation.h`.

Referenced by `checkQuantiles()`, `batch_mean_QE::checkQuantiles()`, `pooling_QE::checkQuantiles()`, `quantile_estimation::set_SSC()`, and `quantile_estimation::~~quantile_estimation()`.

8.30.4.11 INDEX outputAnalyser::m_processedIndexes [protected, inherited]

Definition at line 20 of file basic.h.

Referenced by `evolution::calculateQuantiles()`, `checkQuantiles()`, `batch_mean_QE::checkQuantiles()`, `pooling_QE::checkQuantiles()`, `deterministic_TPD::isReady()`, `evolution::isReady()`, `sequential_TPD::printResult()`, `deterministic_TPD::printResult()`, `printResult()`, `batch_mean_QE::printResult()`, `pooling_QE::printResult()`, `batching::printResult()`, `sequential_TPD::printStatus()`, `deterministic_TPD::printStatus()`, `evolution::printStatus()`, `printStatus()`, `batch_mean_QE::printStatus()`, `pooling_QE::printStatus()`, `batching::printStatus()`, `sequential_TPD::process()`, `deterministic_TPD::process()`, `evolution::process()`, `process()`, `batch_mean_QE::process()`, `pooling_QE::process()`, `batching::process()`, `outputAnalyser::process()`, `sequential_TPD::sub_collect()`, `sequential_TPD::sub_compare()`, and `sequential_TPD::sub_initialize()`.

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

- `quantile_estimation.h`
- `quantile_estimation.cc`

8.31 statistic_collection Class Reference

```
#include <statistic.h>
```

Public Types

- enum **TypeOfIndependenceTest** {
RunsUpDown, **RunsAboveBelow**, **VonNeuman**, **PearsonStrelen**,
PearsonPermutation }

Public Member Functions

- **distribution chooseDistribution** (const std::vector< CONTINUOUS > &sample, CONTINUOUS alpha)
- CONTINUOUS **binomial** (CONTINUOUS pr_success, INDEX no_success, INDEX no_trials) const
- CONTINUOUS **inv_binomial** (CONTINUOUS cumulation, INDEX no_success, INDEX no_trials) const
- CONTINUOUS **normal** (CONTINUOUS X, CONTINUOUS mean, CONTINUOUS variance) const
- CONTINUOUS **inv_normal** (CONTINUOUS cumulation, CONTINUOUS mean, CONTINUOUS variance) const
- CONTINUOUS **f_distribution** (CONTINUOUS X, INDEX df_numerator, INDEX df_denominator) const
- CONTINUOUS **inv_f_distribution** (CONTINUOUS cumulation, INDEX df_numerator, INDEX df_denominator) const
- CONTINUOUS **t_distribution** (CONTINUOUS X, CONTINUOUS df) const
- CONTINUOUS **inv_t_distribution** (CONTINUOUS cumulation, CONTINUOUS df) const
- CONTINUOUS **uniform** (CONTINUOUS X, CONTINUOUS a, CONTINUOUS b) const
- CONTINUOUS **inv_uniform** (CONTINUOUS cumulation, CONTINUOUS a, CONTINUOUS b) const
- CONTINUOUS **exponential** (CONTINUOUS X, CONTINUOUS mean) const
- CONTINUOUS **inv_exponential** (CONTINUOUS cumulation, CONTINUOUS mean) const
- CONTINUOUS **M_M_1_response** (CONTINUOUS X, CONTINUOUS lambda, CONTINUOUS mu) const
- CONTINUOUS **inv_M_M_1_response** (CONTINUOUS cumulation, CONTINUOUS lambda, CONTINUOUS mu) const
- CONTINUOUS **M_E2_1_response** (CONTINUOUS X, CONTINUOUS lambda, CONTINUOUS mu_exp) const
- CONTINUOUS **inv_M_E2_1_response** (CONTINUOUS cumulation, CONTINUOUS lambda, CONTINUOUS mu_exp) const
- CONTINUOUS **M_H2_1_response** (CONTINUOUS X, CONTINUOUS lambda, CONTINUOUS mu1, CONTINUOUS mu2, CONTINUOUS p) const
- CONTINUOUS **inv_M_H2_1_response** (CONTINUOUS cumulation, CONTINUOUS lambda, CONTINUOUS mu1, CONTINUOUS mu2, CONTINUOUS p) const
- CONTINUOUS **sinh** (CONTINUOUS x) const
- CONTINUOUS **cosh** (CONTINUOUS x) const
- CONTINUOUS **tanh** (CONTINUOUS x) const

- CONTINUOUS **coth** (CONTINUOUS x) const
- CONTINUOUS **asinh** (CONTINUOUS x) const
- CONTINUOUS **acosh** (CONTINUOUS x) const
- CONTINUOUS **atanh** (CONTINUOUS x) const
- CONTINUOUS **acoth** (CONTINUOUS x) const
- CONTINUOUS **sq** (CONTINUOUS x) const
- void **chooseQuantiles_old** (const INDEX sampleSize, std::set< **quantile_rank** > &result) const
- void **chooseQuantiles** (const INDEX sampleSize, std::set< **quantile_rank** > &result, const CONTINUOUS alpha) const
- bool **independenceTest** (const CONTINUOUS alpha, const std::list< CONTINUOUS > &data, const **TypeOfIndependenceTest** whichTest) const
- bool **runsUpDown** (const CONTINUOUS alpha, const std::list< CONTINUOUS > &data) const
- void **runsUpDown_statistic** (const std::list< CONTINUOUS > &data, INDEX &pos, INDEX &neg, INDEX &run) const
- bool **runsUpDown_test** (const CONTINUOUS alpha, const INDEX pos, const INDEX neg, const INDEX run) const
- bool **runsAboveBelow** (const CONTINUOUS alpha, const std::list< CONTINUOUS > &data) const
- void **runsAboveBelow_statistic** (const std::list< CONTINUOUS > &data, INDEX &pos, INDEX &neg, INDEX &run) const
- bool **runsAboveBelow_test** (const CONTINUOUS alpha, const INDEX pos, const INDEX neg, const INDEX run) const
- bool **vonNeumann** (const CONTINUOUS alpha, const std::list< CONTINUOUS > &data) const
- void **vonNeumann_statistic** (const std::list< CONTINUOUS > &data, CONTINUOUS &statistic) const
- bool **vonNeumann_test** (const CONTINUOUS alpha, const CONTINUOUS statistic, CONTINUOUS &criticalValue) const
- bool **pearsonStrelen** (const CONTINUOUS alpha, const std::list< CONTINUOUS > &data) const
- void **pearsonStrelen_statistic** (const std::list< CONTINUOUS > &data, CONTINUOUS &statistic) const
- bool **pearsonStrelen_test** (const CONTINUOUS alpha, const std::list< CONTINUOUS > &data, const CONTINUOUS statistic, CONTINUOUS &criticalValue) const
- bool **pearsonPermutation** (const CONTINUOUS alpha, const std::list< CONTINUOUS > &data) const
- void **pearsonPermutation_statistic** (const std::list< CONTINUOUS > &data, CONTINUOUS &statistic) const
- bool **pearsonPermutation_test** (const CONTINUOUS alpha, const std::list< CONTINUOUS > &data, const CONTINUOUS statistic, CONTINUOUS &criticalValueLow, CONTINUOUS &criticalValueHigh) const
- CONTINUOUS **binomialCoefficient** (const INDEX n, const INDEX k) const
- CONTINUOUS **binomialCoefficient** (const CONTINUOUS n, const DISCRETE k) const
- CONTINUOUS **binomialCoefficient** (const CONTINUOUS n, const CONTINUOUS k) const
- CONTINUOUS **pearsonsCorrelationCoefficient** (const std::list< CONTINUOUS > &, const std::list< CONTINUOUS > &) const
- CONTINUOUS **spearmanCorrelationCoefficient** (const std::list< CONTINUOUS > &, const std::list< CONTINUOUS > &) const

- CONTINUOUS `vonNeumannsCorrelationCoefficient` (const std::list< CONTINUOUS > &) const
- CONTINUOUS `mean` (const std::list< CONTINUOUS > &) const
- void `ranks` (const std::list< CONTINUOUS > &, std::list< CONTINUOUS > &) const
- void `generateRandomPermutation` (const std::list< CONTINUOUS > &, std::list< CONTINUOUS > &) const
- void `permutationAll` (const std::list< CONTINUOUS > &, std::list< std::list< CONTINUOUS > > &) const
- bool `siegelsRunTest` (const INDEX n1, const INDEX n2, const INDEX r, CONTINUOUS alpha, bool &valid) const
- bool `siegelsRunTest_small` (const INDEX n1, const INDEX n2, const INDEX r, CONTINUOUS alpha, bool &valid) const
- bool `siegelsRunTest_large` (const INDEX n1, const INDEX n2, const INDEX r, CONTINUOUS alpha, bool &valid) const
- CONTINUOUS `infiniteSumCorrelationCoefficients_MM1` (const CONTINUOUS interarrivalRate, const CONTINUOUS serviceRate) const
- CONTINUOUS `finiteSumCorrelationCoefficients_MM1` (const CONTINUOUS interarrivalRate, const CONTINUOUS serviceRate, const INDEX n) const

8.31.1 Detailed Description

Definition at line 17 of file `statistic.h`.

8.31.2 Member Enumeration Documentation

8.31.2.1 enum `statistic_collection::TypeOfIndependenceTest`

Enumerator:

RunsUpDown

RunsAboveBelow

VonNeuman

PearsonStrelen

PearsonPermutation

Definition at line 100 of file `statistic.h`.

8.31.3 Member Function Documentation

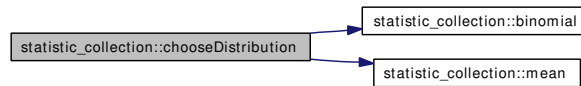
8.31.3.1 distribution `statistic_collection::chooseDistribution` (const std::vector< CONTINUOUS > & *sample*, CONTINUOUS *alpha*)

Definition at line 15 of file `statistic.cc`.

References `binomial()`, `CONTINUOUS`, `EXPONENTIAL`, `INDEX`, `mean()`, `NORMAL`, `UNIFORM`, and `UNSPECIFIED`.

Referenced by `spectral_analysis_QE::checkQuantiles()`, and `batch_mean_QE::checkQuantiles()`.

Here is the call graph for this function:



8.31.3.2 CONTINUOUS `statistic_collection::binomial` (CONTINUOUS *pr_success*, INDEX *no_success*, INDEX *no_trials*) const

Definition at line 70 of file `statistic.cc`.

References CONTINUOUS.

Referenced by `chooseDistribution()`, and `quantile_rank::quantileCDF()`.

8.31.3.3 CONTINUOUS `statistic_collection::inv_binomial` (CONTINUOUS *cumulation*, INDEX *no_success*, INDEX *no_trials*) const

Definition at line 100 of file `statistic.cc`.

References CONTINUOUS.

8.31.3.4 CONTINUOUS `statistic_collection::normal` (CONTINUOUS *X*, CONTINUOUS *mean*, CONTINUOUS *variance*) const

Definition at line 130 of file `statistic.cc`.

References CONTINUOUS.

8.31.3.5 CONTINUOUS `statistic_collection::inv_normal` (CONTINUOUS *cumulation*, CONTINUOUS *mean*, CONTINUOUS *variance*) const

Definition at line 156 of file `statistic.cc`.

References CONTINUOUS.

Referenced by `prng::draw_normal()`, `siegelsRunTest_large()`, and `vonNeumann_test()`.

8.31.3.6 CONTINUOUS `statistic_collection::f_distribution` (CONTINUOUS *X*, INDEX *df_numerator*, INDEX *df_denominator*) const

Definition at line 182 of file `statistic.cc`.

References CONTINUOUS.

8.31.3.7 CONTINUOUS `statistic_collection::inv_f_distribution` (CONTINUOUS *cumulation*, INDEX *df_numerator*, INDEX *df_denominator*) const

Definition at line 201 of file `statistic.cc`.

References CONTINUOUS.

8.31.3.8 CONTINUOUS `statistic_collection::t_distribution` (CONTINUOUS `X`, CONTINUOUS `df`) const

Definition at line 219 of file `statistic.cc`.

References CONTINUOUS.

8.31.3.9 CONTINUOUS `statistic_collection::inv_t_distribution` (CONTINUOUS `cumulation`, CONTINUOUS `df`) const

Definition at line 240 of file `statistic.cc`.

References CONTINUOUS.

Referenced by `spectral_analysis_QE::checkQuantiles()`, and `batch_mean_QE::checkQuantiles()`.

8.31.3.10 CONTINUOUS `statistic_collection::uniform` (CONTINUOUS `X`, CONTINUOUS `a`, CONTINUOUS `b`) const

Definition at line 258 of file `statistic.cc`.

References CONTINUOUS.

8.31.3.11 CONTINUOUS `statistic_collection::inv_uniform` (CONTINUOUS `cumulation`, CONTINUOUS `a`, CONTINUOUS `b`) const

Definition at line 267 of file `statistic.cc`.

8.31.3.12 CONTINUOUS `statistic_collection::exponential` (CONTINUOUS `X`, CONTINUOUS `mean`) const

Definition at line 273 of file `statistic.cc`.

Referenced by `M_M_1_response()`.

8.31.3.13 CONTINUOUS `statistic_collection::inv_exponential` (CONTINUOUS `cumulation`, CONTINUOUS `mean`) const

Definition at line 279 of file `statistic.cc`.

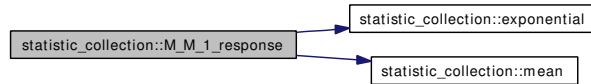
Referenced by `inv_M_M_1_response()`.

8.31.3.14 CONTINUOUS `statistic_collection::M_M_1_response` (CONTINUOUS `X`, CONTINUOUS `lambda`, CONTINUOUS `mu`) const

Definition at line 285 of file `statistic.cc`.

References CONTINUOUS, `exponential()`, and `mean()`.

Here is the call graph for this function:

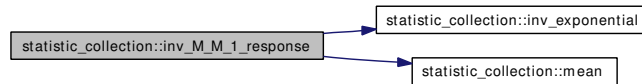


8.31.3.15 CONTINUOUS `statistic_collection::inv_M_M_1_response` (CONTINUOUS *cumulation*, CONTINUOUS *lambda*, CONTINUOUS *mu*) `const`

Definition at line 295 of file `statistic.cc`.

References CONTINUOUS, `inv_exponential()`, and `mean()`.

Here is the call graph for this function:



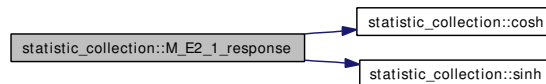
8.31.3.16 CONTINUOUS `statistic_collection::M_E2_1_response` (CONTINUOUS *X*, CONTINUOUS *lambda*, CONTINUOUS *mu_exp*) `const`

Definition at line 305 of file `statistic.cc`.

References CONTINUOUS, `cosh()`, and `sinh()`.

Referenced by `inv_M_E2_1_response()`.

Here is the call graph for this function:

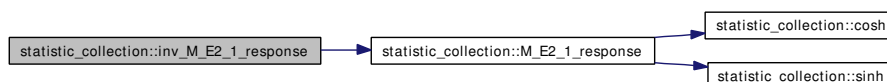


8.31.3.17 CONTINUOUS `statistic_collection::inv_M_E2_1_response` (CONTINUOUS *cumulation*, CONTINUOUS *lambda*, CONTINUOUS *mu_exp*) `const`

Definition at line 321 of file `statistic.cc`.

References CONTINUOUS, and `M_E2_1_response()`.

Here is the call graph for this function:



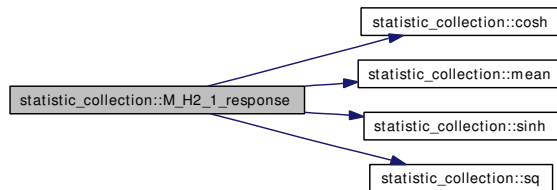
8.31.3.18 CONTINUOUS `statistic_collection::M_H2_1_response` (CONTINUOUS X , CONTINUOUS λ , CONTINUOUS μ_1 , CONTINUOUS μ_2 , CONTINUOUS p) const

Definition at line 359 of file `statistic.cc`.

References CONTINUOUS, `cosh()`, `mean()`, `sinh()`, and `sq()`.

Referenced by `inv_M_H2_1_response()`.

Here is the call graph for this function:

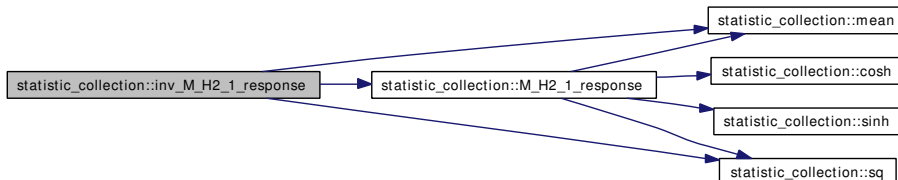


8.31.3.19 CONTINUOUS `statistic_collection::inv_M_H2_1_response` (CONTINUOUS $cumulation$, CONTINUOUS λ , CONTINUOUS μ_1 , CONTINUOUS μ_2 , CONTINUOUS p) const

Definition at line 382 of file `statistic.cc`.

References CONTINUOUS, `M_H2_1_response()`, `mean()`, and `sq()`.

Here is the call graph for this function:



8.31.3.20 CONTINUOUS `statistic_collection::sinh` (CONTINUOUS x) const

Definition at line 427 of file `statistic.cc`.

Referenced by `M_E2_1_response()`, and `M_H2_1_response()`.

8.31.3.21 CONTINUOUS `statistic_collection::cosh` (CONTINUOUS x) const

Definition at line 431 of file `statistic.cc`.

Referenced by `M_E2_1_response()`, and `M_H2_1_response()`.

8.31.3.22 CONTINUOUS `statistic_collection::tanh` (CONTINUOUS x) const

Definition at line 435 of file `statistic.cc`.

References CONTINUOUS.

8.31.3.23 CONTINUOUS `statistic_collection::coth (CONTINUOUS x) const`

Definition at line 440 of file `statistic.cc`.

References CONTINUOUS.

8.31.3.24 CONTINUOUS `statistic_collection::asinh (CONTINUOUS x) const`

Definition at line 445 of file `statistic.cc`.

8.31.3.25 CONTINUOUS `statistic_collection::acosh (CONTINUOUS x) const`

Definition at line 449 of file `statistic.cc`.

8.31.3.26 CONTINUOUS `statistic_collection::atanh (CONTINUOUS x) const`

Definition at line 453 of file `statistic.cc`.

8.31.3.27 CONTINUOUS `statistic_collection::acoth (CONTINUOUS x) const`

Definition at line 457 of file `statistic.cc`.

8.31.3.28 CONTINUOUS `statistic_collection::sq (CONTINUOUS x) const` [inline]

Definition at line 90 of file `statistic.h`.

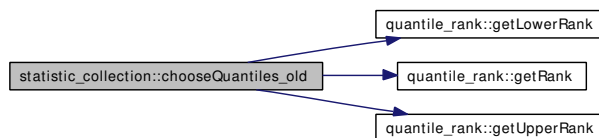
Referenced by `inv_M_H2_1_response()`, and `M_H2_1_response()`.

8.31.3.29 `void statistic_collection::chooseQuantiles_old (const INDEX sampleSize, std::set< quantile_rank > & result) const`

Definition at line 461 of file `statistic.cc`.

References DISCRETE, `quantile_rank::getLowerRank()`, `quantile_rank::getRank()`, `quantile_rank::getUpperRank()`, and INDEX.

Here is the call graph for this function:



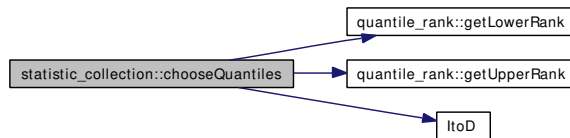
8.31.3.30 `void statistic_collection::chooseQuantiles (const INDEX sampleSize, std::set< quantile_rank > & result, const CONTINUOUS alpha) const`

Definition at line 520 of file `statistic.cc`.

References `CONTINUOUS`, `DISCRETE`, `quantile_rank::getLowerRank()`, `quantile_rank::getUpperRank()`, `INDEX`, and `ItoD()`.

Referenced by `pooling_QE::checkQuantiles()`, and `evolution::evolution()`.

Here is the call graph for this function:

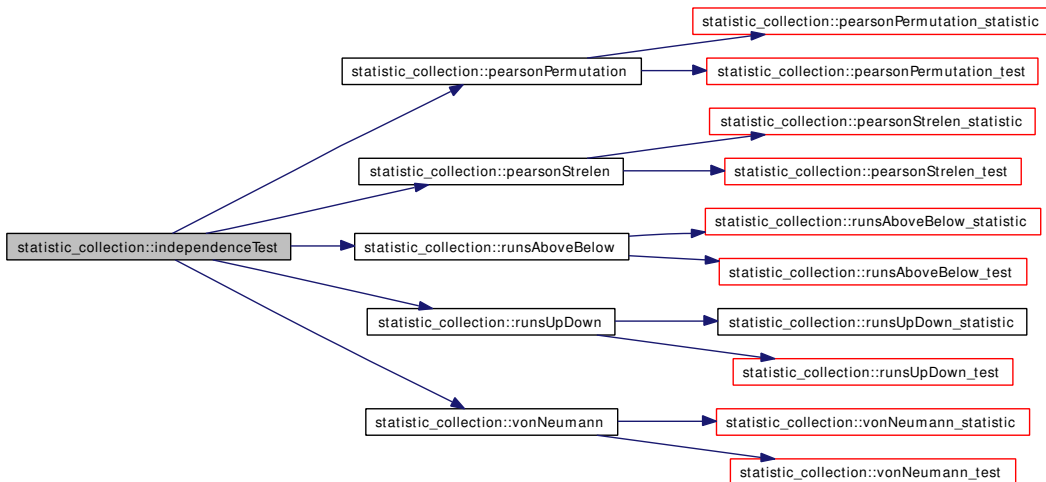


8.31.3.31 `bool statistic_collection::independenceTest (const CONTINUOUS alpha, const std::list< CONTINUOUS > & data, const TypeOfIndependenceTest whichTest) const`

Definition at line 591 of file `statistic.cc`.

References `pearsonPermutation()`, `PearsonPermutation`, `pearsonStrelen()`, `PearsonStrelen`, `runsAboveBelow()`, `RunsAboveBelow`, `runsUpDown()`, `RunsUpDown`, `VonNeuman`, and `vonNeumann()`.

Here is the call graph for this function:



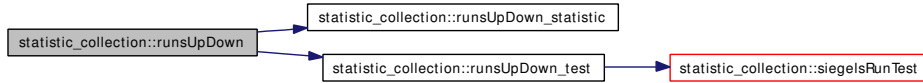
8.31.3.32 `bool statistic_collection::runsUpDown (const CONTINUOUS alpha, const std::list< CONTINUOUS > & data) const`

Definition at line 606 of file `statistic.cc`.

References `INDEX`, `runsUpDown_statistic()`, and `runsUpDown_test()`.

Referenced by `independenceTest()`.

Here is the call graph for this function:



8.31.3.33 `void statistic_collection::runsUpDown_statistic (const std::list< CONTINUOUS > & data, INDEX & pos, INDEX & neg, INDEX & run) const`

Definition at line 613 of file `statistic.cc`.

Referenced by `runsUpDown()`, and `batching::testBatchStatistic()`.

8.31.3.34 `bool statistic_collection::runsUpDown_test (const CONTINUOUS alpha, const INDEX pos, const INDEX neg, const INDEX run) const`

Definition at line 645 of file `statistic.cc`.

References `siegelsRunTest()`.

Referenced by `runsUpDown()`, and `batching::testBatchStatistic()`.

Here is the call graph for this function:



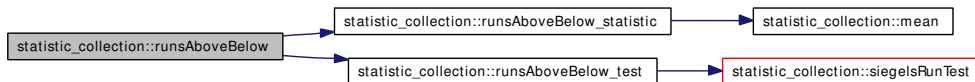
8.31.3.35 `bool statistic_collection::runsAboveBelow (const CONTINUOUS alpha, const std::list< CONTINUOUS > & data) const`

Definition at line 656 of file `statistic.cc`.

References `INDEX`, `runsAboveBelow_statistic()`, and `runsAboveBelow_test()`.

Referenced by `independenceTest()`.

Here is the call graph for this function:



8.31.3.36 `void statistic_collection::runsAboveBelow_statistic (const std::list< CONTINUOUS > & data, INDEX & pos, INDEX & neg, INDEX & run) const`

Definition at line 663 of file `statistic.cc`.

References `mean()`.

Referenced by `runsAboveBelow()`, and `batching::testBatchStatistic()`.

Here is the call graph for this function:



8.31.3.37 `bool statistic_collection::runsAboveBelow_test` (`const CONTINUOUS α` , `const INDEX pos` , `const INDEX neg` , `const INDEX run`) `const`

Definition at line 690 of file `statistic.cc`.

References `siegelsRunTest()`.

Referenced by `runsAboveBelow()`, and `batching::testBatchStatistic()`.

Here is the call graph for this function:



8.31.3.38 `bool statistic_collection::vonNeumann` (`const CONTINUOUS α` , `const std::list< CONTINUOUS > & $data$`) `const`

Definition at line 700 of file `statistic.cc`.

References `CONTINUOUS`, `vonNeumann_statistic()`, and `vonNeumann_test()`.

Referenced by `independenceTest()`.

Here is the call graph for this function:



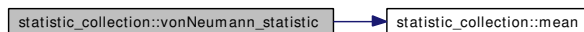
8.31.3.39 `void statistic_collection::vonNeumann_statistic` (`const std::list< CONTINUOUS > & $data$` , `CONTINUOUS & $statistic$`) `const`

Definition at line 707 of file `statistic.cc`.

References `CONTINUOUS`, `INDEX`, and `mean()`.

Referenced by `batching::testBatchStatistic()`, and `vonNeumann()`.

Here is the call graph for this function:



8.31.3.40 `bool statistic_collection::vonNeumann_test (const CONTINUOUS α , const CONTINUOUS \bar{C} statistic, CONTINUOUS & criticalValue) const`

Definition at line 745 of file `statistic.cc`.

References `inv_normal()`.

Referenced by `batching::testBatchStatistic()`, and `vonNeumann()`.

Here is the call graph for this function:



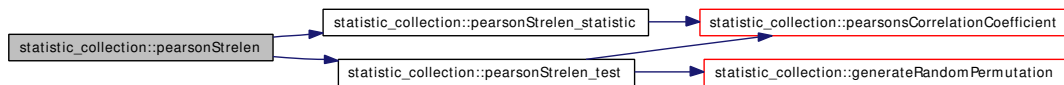
8.31.3.41 `bool statistic_collection::pearsonStrelen (const CONTINUOUS α , const std::list< CONTINUOUS > & data) const`

Definition at line 753 of file `statistic.cc`.

References `CONTINUOUS`, `pearsonStrelen_statistic()`, and `pearsonStrelen_test()`.

Referenced by `independenceTest()`.

Here is the call graph for this function:



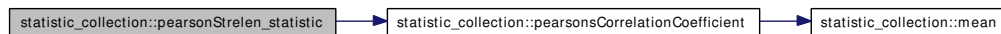
8.31.3.42 `void statistic_collection::pearsonStrelen_statistic (const std::list< CONTINUOUS > & data, CONTINUOUS & statistic) const`

Definition at line 760 of file `statistic.cc`.

References `pearsonsCorrelationCoefficient()`.

Referenced by `pearsonStrelen()`, and `batching::testBatchStatistic()`.

Here is the call graph for this function:



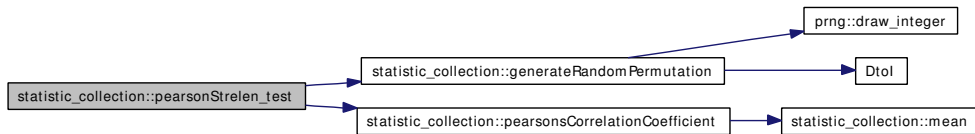
8.31.3.43 `bool statistic_collection::pearsonStrelen_test (const CONTINUOUS α , const std::list< CONTINUOUS > & data, const CONTINUOUS statistic, CONTINUOUS & criticalValue) const`

Definition at line 771 of file `statistic.cc`.

References `CONTINUOUS`, `generateRandomPermutation()`, `INDEX`, and `pearsonsCorrelationCoefficient()`.

Referenced by `pearsonStrelen()`, and `batching::testBatchStatistic()`.

Here is the call graph for this function:



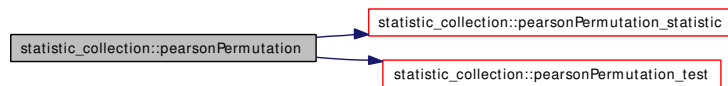
8.31.3.44 bool statistic_collection::pearsonPermutation (const CONTINUOUS *alpha*, const std::list< CONTINUOUS > & *data*) const

Definition at line 799 of file statistic.cc.

References CONTINUOUS, pearsonPermutation_statistic(), and pearsonPermutation_test().

Referenced by independenceTest().

Here is the call graph for this function:



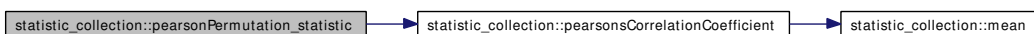
8.31.3.45 void statistic_collection::pearsonPermutation_statistic (const std::list< CONTINUOUS > & *data*, CONTINUOUS & *statistic*) const

Definition at line 806 of file statistic.cc.

References pearsonsCorrelationCoefficient().

Referenced by pearsonPermutation(), and batching::testBatchStatistic().

Here is the call graph for this function:



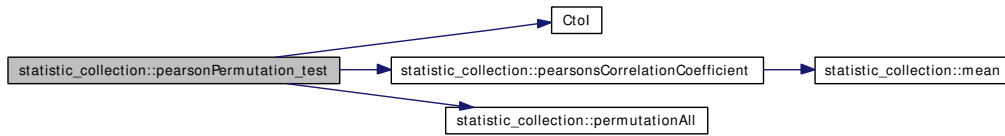
8.31.3.46 bool statistic_collection::pearsonPermutation_test (const CONTINUOUS *alpha*, const std::list< CONTINUOUS > & *data*, const CONTINUOUS *statistic*, CONTINUOUS & *criticalValueLow*, CONTINUOUS & *criticalValueHigh*) const

Definition at line 817 of file statistic.cc.

References CONTINUOUS, CtoI(), INDEX, pearsonsCorrelationCoefficient(), and permutation-All().

Referenced by pearsonPermutation(), and batching::testBatchStatistic().

Here is the call graph for this function:



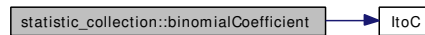
8.31.3.47 CONTINUOUS `statistic_collection::binomialCoefficient` (const INDEX n , const INDEX k) const

Definition at line 856 of file `statistic.cc`.

References CONTINUOUS, INDEX, and `ItoC()`.

Referenced by `finiteSumCorrelationCoefficients_MM1()`.

Here is the call graph for this function:

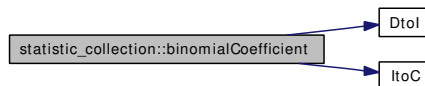


8.31.3.48 CONTINUOUS `statistic_collection::binomialCoefficient` (const CONTINUOUS n , const DISCRETE k) const

Definition at line 875 of file `statistic.cc`.

References CONTINUOUS, `DtoI()`, INDEX, and `ItoC()`.

Here is the call graph for this function:



8.31.3.49 CONTINUOUS `statistic_collection::binomialCoefficient` (const CONTINUOUS n , const CONTINUOUS k) const

Definition at line 891 of file `statistic.cc`.

8.31.3.50 CONTINUOUS `statistic_collection::pearsonsCorrelationCoefficient` (const `std::list< CONTINUOUS >` &, const `std::list< CONTINUOUS >` &) const

Definition at line 898 of file `statistic.cc`.

References CONTINUOUS, INDEX, and `mean()`.

Referenced by `pearsonPermutation_statistic()`, `pearsonPermutation_test()`, `pearsonStrelen_statistic()`, and `pearsonStrelen_test()`.

Here is the call graph for this function:



8.31.3.51 CONTINUOUS `statistic_collection::spearmansCorrelationCoefficient` (`const std::list< CONTINUOUS > &`, `const std::list< CONTINUOUS > &`) `const`

Definition at line 944 of file `statistic.cc`.

References `CONTINUOUS`, `INDEX`, and `ranks()`.

Here is the call graph for this function:



8.31.3.52 CONTINUOUS `statistic_collection::vonNeumannsCorrelationCoefficient` (`const std::list< CONTINUOUS > &`) `const`

Definition at line 976 of file `statistic.cc`.

References `CONTINUOUS`, `INDEX`, and `mean()`.

Here is the call graph for this function:



8.31.3.53 CONTINUOUS `statistic_collection::mean` (`const std::list< CONTINUOUS > &`) `const`

Definition at line 1013 of file `statistic.cc`.

References `CONTINUOUS`.

Referenced by `chooseDistribution()`, `inv_M_H2_1_response()`, `inv_M_M_1_response()`, `M_H2_1_response()`, `M_M_1_response()`, `pearsonsCorrelationCoefficient()`, `runsAboveBelow_statistic()`, `siegelsRunTest_large()`, `vonNeumann_statistic()`, and `vonNeumannsCorrelationCoefficient()`.

8.31.3.54 `void statistic_collection::ranks` (`const std::list< CONTINUOUS > &`, `std::list< CONTINUOUS > &`) `const`

Definition at line 1019 of file `statistic.cc`.

References `CONTINUOUS`, and `INDEX`.

Referenced by `spearmansCorrelationCoefficient()`.

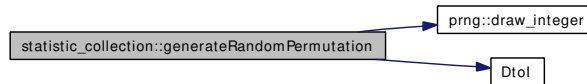
8.31.3.55 void statistic_collection::generateRandomPermutation (const std::list< CONTINUOUS > &, std::list< CONTINUOUS > &) const

Definition at line 1043 of file statistic.cc.

References CONTINUOUS, DISCRETE, prng::draw_integer(), DtoI(), INDEX, and lib_prng.

Referenced by pearsonStrelen_test().

Here is the call graph for this function:



8.31.3.56 void statistic_collection::permutationAll (const std::list< CONTINUOUS > &, std::list< std::list< CONTINUOUS > > &) const

Definition at line 1083 of file statistic.cc.

References CONTINUOUS, and INDEX.

Referenced by pearsonPermutation_test().

8.31.3.57 bool statistic_collection::siegersRunTest (const INDEX n1, const INDEX n2, const INDEX r, CONTINUOUS alpha, bool & valid) const

Definition at line 1112 of file statistic.cc.

References siegersRunTest_large(), and siegersRunTest_small().

Referenced by runsAboveBelow_test(), and runsUpDown_test().

Here is the call graph for this function:



8.31.3.58 bool statistic_collection::siegersRunTest_small (const INDEX n1, const INDEX n2, const INDEX r, CONTINUOUS alpha, bool & valid) const

Definition at line 1121 of file statistic.cc.

References DISCRETE, and INDEX.

Referenced by siegersRunTest().

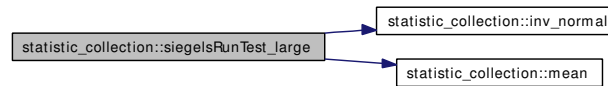
8.31.3.59 bool statistic_collection::siegersRunTest_large (const INDEX n1, const INDEX n2, const INDEX r, CONTINUOUS alpha, bool & valid) const

Definition at line 1429 of file statistic.cc.

References CONTINUOUS, inv_normal(), and mean().

Referenced by siegelsRunTest().

Here is the call graph for this function:



8.31.3.60 CONTINUOUS statistic_collection::infiniteSumCorrelationCoefficients_MM1 (const CONTINUOUS *interarrivalRate*, const CONTINUOUS *serviceRate*) const

Definition at line 1449 of file statistic.cc.

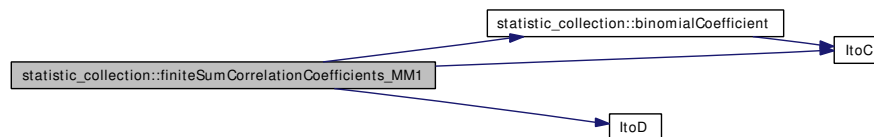
References CONTINUOUS.

8.31.3.61 CONTINUOUS statistic_collection::finiteSumCorrelationCoefficients_MM1 (const CONTINUOUS *interarrivalRate*, const CONTINUOUS *serviceRate*, const INDEX *n*) const

Definition at line 1475 of file statistic.cc.

References binomialCoefficient(), CONTINUOUS, INDEX, ItoC(), and ItoD().

Here is the call graph for this function:



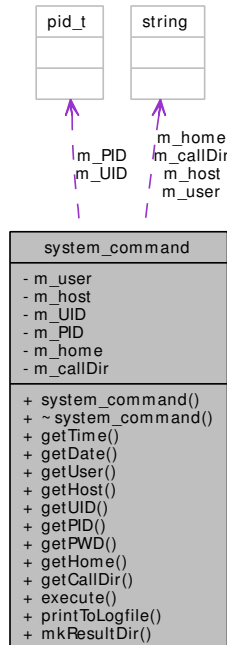
The documentation for this class was generated from the following files:

- statistic.h
- statistic.cc

8.32 system_command Class Reference

```
#include <system_command.h>
```

Collaboration diagram for system_command:



Public Member Functions

- `system_command` (void)
- `~system_command` (void)
- const std::string `getTime` (void)
- const std::string `getDate` (void)
- const std::string `getUser` (void)
- const std::string `getHost` (void)
- const pid_t `getUID` (void)
- const pid_t `getPID` (void)
- const std::string `getPWD` (void)
- const std::string `getHome` (void)
- const std::string `getCallDir` (void)
- void `execute` (const std::string &)
- void `printToLogfile` (void)
- void `mkResultDir` (const std::string &)

Private Attributes

- std::string `m_user`
- std::string `m_host`
- pid_t `m_UID`
- pid_t `m_PID`

- `std::string m_home`
- `std::string m_callDir`

8.32.1 Detailed Description

Definition at line 8 of file `system_command.h`.

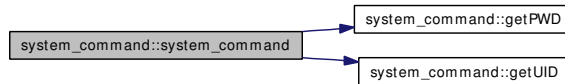
8.32.2 Constructor & Destructor Documentation

8.32.2.1 `system_command::system_command (void)`

Definition at line 14 of file `system_command.cc`.

References `getPWD()`, `getUID()`, `m_callDir`, `m_home`, and `m_user`.

Here is the call graph for this function:



8.32.2.2 `system_command::~~system_command (void)`

Definition at line 25 of file `system_command.cc`.

References `getCallDir()`.

Here is the call graph for this function:



8.32.3 Member Function Documentation

8.32.3.1 `const std::string system_command::getTime (void)`

Definition at line 29 of file `system_command.cc`.

Referenced by `mkResultDir()`, `resultInfo::print()`, and `printToLogfile()`.

8.32.3.2 `const std::string system_command::getDate (void)`

Definition at line 46 of file `system_command.cc`.

Referenced by `mkResultDir()`, `resultInfo::print()`, and `printToLogfile()`.

8.32.3.3 `const std::string system_command::getUser (void)`

Definition at line 61 of file `system_command.cc`.

References `m_user`.

Referenced by `mkResultDir()`, and `printToLogfile()`.

8.32.3.4 `const std::string system_command::getHost (void)`

Definition at line 65 of file `system_command.cc`.

References `m_host`.

Referenced by `mkResultDir()`, `printToLogfile()`, and `prng::prng()`.

8.32.3.5 `const pid_t system_command::getUID (void)`

Definition at line 75 of file `system_command.cc`.

References `m_UID`.

Referenced by `printToLogfile()`, and `system_command()`.

8.32.3.6 `const pid_t system_command::getPID (void)`

Definition at line 81 of file `system_command.cc`.

References `m_PID`.

Referenced by `mkResultDir()`, and `printToLogfile()`.

8.32.3.7 `const std::string system_command::getPWD (void)`

Definition at line 87 of file `system_command.cc`.

Referenced by `mkResultDir()`, and `system_command()`.

8.32.3.8 `const std::string system_command::getHome (void)`

Definition at line 94 of file `system_command.cc`.

References `m_home`.

Referenced by `setting::load()`, `printToLogfile()`, and `prng::prng()`.

8.32.3.9 `const std::string system_command::getCallDir (void)`

Definition at line 98 of file `system_command.cc`.

References `m_callDir`.

Referenced by `printToLogfile()`, and `~system_command()`.

8.32.3.10 `void system_command::execute (const std::string &)`

Definition at line 102 of file `system_command.cc`.

Referenced by `SequentialStoppingCriteria_QE::print()`, `sequential_TPD::printDistribution()`, and `evolution::printResult()`.

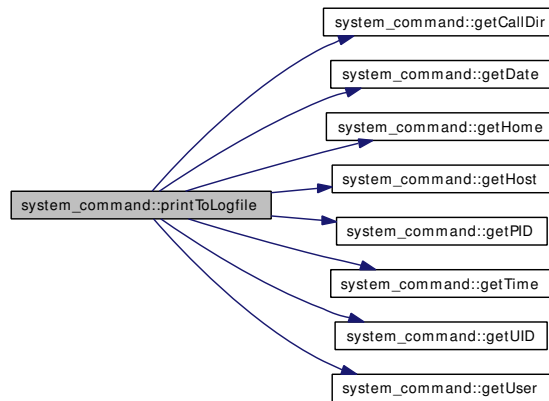
8.32.3.11 void system_command::printToLogfile (void)

Definition at line 119 of file system_command.cc.

References `getCallDir()`, `getDate()`, `getHome()`, `getHost()`, `getPID()`, `getTime()`, `getUID()`, `getUser()`, and `logfile`.

Referenced by `main()`.

Here is the call graph for this function:



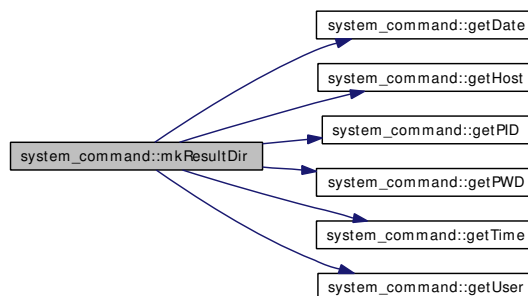
8.32.3.12 void system_command::mkResultDir (const std::string &)

Definition at line 107 of file system_command.cc.

References `getDate()`, `getHost()`, `getPID()`, `getPWD()`, `getTime()`, and `getUser()`.

Referenced by `main()`.

Here is the call graph for this function:



8.32.4 Field Documentation

8.32.4.1 std::string system_command::m_user [private]

Definition at line 29 of file system_command.h.

Referenced by `getUser()`, and `system_command()`.

8.32.4.2 `std::string system_command::m_host` [private]

Definition at line 30 of file `system_command.h`.

Referenced by `getHost()`.

8.32.4.3 `pid_t system_command::m_UID` [private]

Definition at line 31 of file `system_command.h`.

Referenced by `getUID()`.

8.32.4.4 `pid_t system_command::m_PID` [private]

Definition at line 32 of file `system_command.h`.

Referenced by `getPID()`.

8.32.4.5 `std::string system_command::m_home` [private]

Definition at line 34 of file `system_command.h`.

Referenced by `getHome()`, and `system_command()`.

8.32.4.6 `std::string system_command::m_callDir` [private]

Definition at line 35 of file `system_command.h`.

Referenced by `getCallDir()`, and `system_command()`.

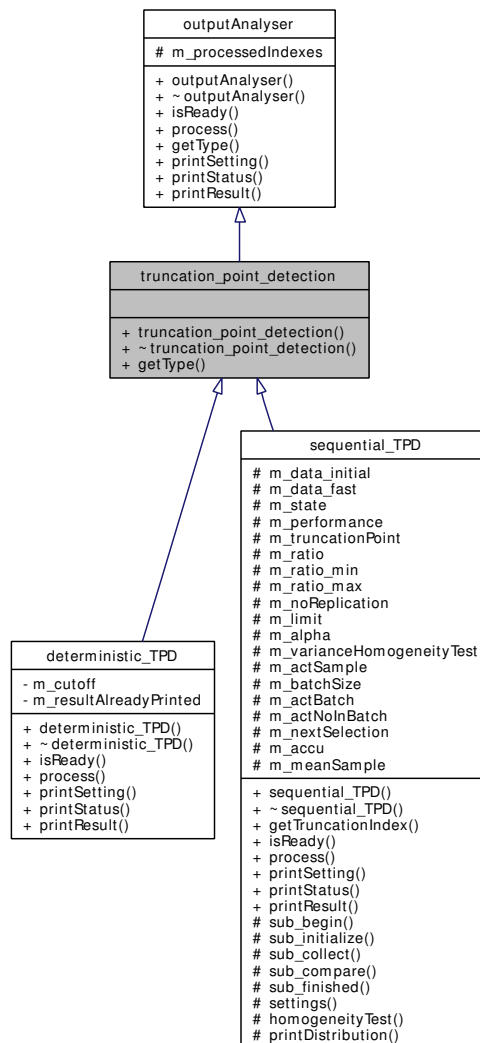
The documentation for this class was generated from the following files:

- `system_command.h`
- `system_command.cc`

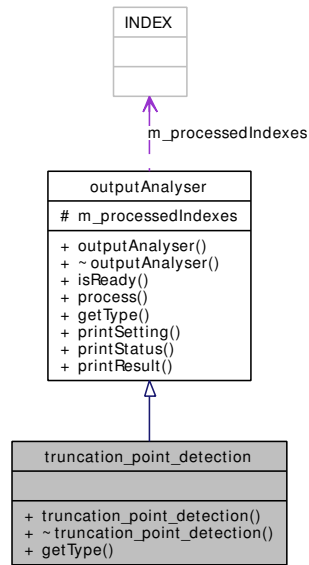
8.33 truncation_point_detection Class Reference

```
#include <truncation_point_detection.h>
```

Inheritance diagram for truncation_point_detection:



Collaboration diagram for truncation_point_detection:



Public Member Functions

- **truncation_point_detection** (void)
- virtual **~truncation_point_detection** (void)
- virtual **TypeOfMethod getType** (void) const
- virtual bool **isReady** (void) const
- virtual void **process** (const std::list< CONTINUOUS > &)
- virtual void **printSetting** (void)
- virtual void **printStatus** (void)
- virtual void **printResult** (void)

Protected Attributes

- INDEX **m_processedIndexes**

8.33.1 Detailed Description

Definition at line 6 of file `truncation_point_detection.h`.

8.33.2 Constructor & Destructor Documentation

8.33.2.1 `truncation_point_detection::truncation_point_detection` (void)

Definition at line 8 of file `truncation_point_detection.cc`.

8.33.2.2 `truncation_point_detection::~truncation_point_detection` (void) [virtual]

Definition at line 11 of file `truncation_point_detection.cc`.

8.33.3 Member Function Documentation

8.33.3.1 `TypeOfMethod truncation_point_detection::getType (void) const` [virtual]

Reimplemented from `outputAnalyser` (p. 87).

Definition at line 14 of file `truncation_point_detection.cc`.

References IDENTICAL.

8.33.3.2 `bool outputAnalyser::isReady (void) const` [virtual, inherited]

Reimplemented in `batching` (p. 40), `pooling_QE` (p. 92), `batch_mean_QE` (p. 31), `spectral_analysis_QE` (p. 153), `evolution` (p. 69), `deterministic_TPD` (p. 61), and `sequential_TPD` (p. 128).

Definition at line 11 of file `basic.cc`.

8.33.3.3 `void outputAnalyser::process (const std::list< CONTINUOUS > &)` [virtual, inherited]

Reimplemented in `batching` (p. 40), `pooling_QE` (p. 92), `batch_mean_QE` (p. 32), `spectral_analysis_QE` (p. 153), `evolution` (p. 69), `deterministic_TPD` (p. 61), and `sequential_TPD` (p. 128).

Definition at line 15 of file `basic.cc`.

References `outputAnalyser::m_processedIndexes`.

8.33.3.4 `void outputAnalyser::printSetting (void)` [virtual, inherited]

Reimplemented in `batching` (p. 41), `pooling_QE` (p. 92), `batch_mean_QE` (p. 32), `spectral_analysis_QE` (p. 154), `evolution` (p. 69), `deterministic_TPD` (p. 61), and `sequential_TPD` (p. 129).

Definition at line 23 of file `basic.cc`.

8.33.3.5 `void outputAnalyser::printStatus (void)` [virtual, inherited]

Reimplemented in `batching` (p. 41), `pooling_QE` (p. 92), `batch_mean_QE` (p. 32), `spectral_analysis_QE` (p. 154), `evolution` (p. 69), `deterministic_TPD` (p. 62), and `sequential_TPD` (p. 129).

Definition at line 26 of file `basic.cc`.

8.33.3.6 `void outputAnalyser::printResult (void)` [virtual, inherited]

Reimplemented in `batching` (p. 41), `pooling_QE` (p. 92), `batch_mean_QE` (p. 32), `spectral_analysis_QE` (p. 154), `evolution` (p. 69), `deterministic_TPD` (p. 62), and `sequential_TPD` (p. 129).

Definition at line 29 of file `basic.cc`.

8.33.4 Field Documentation

8.33.4.1 INDEX outputAnalyser::m_processedIndexes [protected, inherited]

Definition at line 20 of file basic.h.

Referenced by evolution::calculateQuantiles(), spectral_analysis_QE::checkQuantiles(), batch_mean_QE::checkQuantiles(), pooling_QE::checkQuantiles(), deterministic_TPD::isReady(), evolution::isReady(), sequential_TPD::printResult(), deterministic_TPD::printResult(), spectral_analysis_QE::printResult(), batch_mean_QE::printResult(), pooling_QE::printResult(), batching::printResult(), sequential_TPD::printStatus(), deterministic_TPD::printStatus(), evolution::printStatus(), spectral_analysis_QE::printStatus(), batch_mean_QE::printStatus(), pooling_QE::printStatus(), batching::printStatus(), sequential_TPD::process(), deterministic_TPD::process(), evolution::process(), spectral_analysis_QE::process(), batch_mean_QE::process(), pooling_QE::process(), batching::process(), outputAnalyser::process(), sequential_TPD::sub_collect(), sequential_TPD::sub_compare(), and sequential_TPD::sub_initialize().

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

- **truncation_point_detection.h**
- **truncation_point_detection.cc**

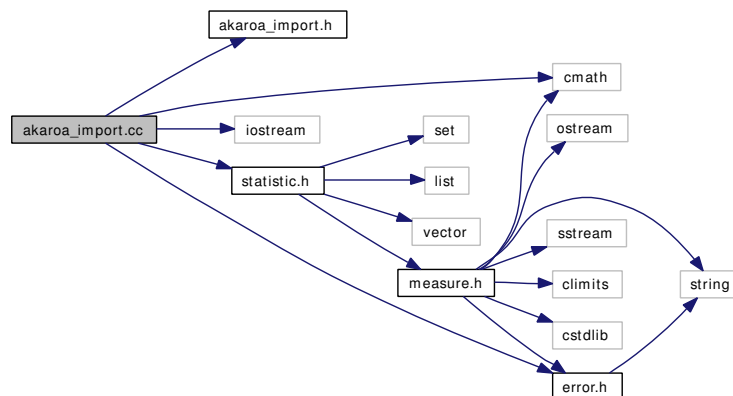
Chapter 9

Sequential Quantile Estimation File Documentation

9.1 akaroa_import.cc File Reference

```
#include "akaroa_import.h"  
#include <cmath>  
#include <iostream>  
#include "statistic.h"  
#include "error.h"
```

Include dependency graph for akaroa_import.cc:



Data Structures

- struct `K_d_entry`

Variables

- `akaroa_import lib_akaroa`

- static struct `K_d_entry K_d_table []`

9.1.1 Variable Documentation

9.1.1.1 struct `K_d_entry K_d_table[]` [static]

Referenced by `akaroa_import::LookUp_K_d()`.

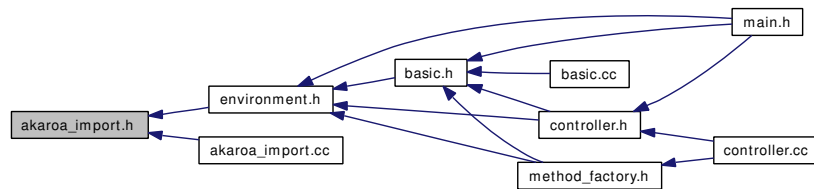
9.1.1.2 `akaroa_import lib_akaroa`

Definition at line 10 of file `akaroa_import.cc`.

Referenced by `spectral_analysis_QE::checkQuantiles()`.

9.2 akaroa_import.h File Reference

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



Data Structures

- class `akaroa_import`

Variables

- `akaroa_import lib_akaroa`

9.2.1 Variable Documentation

9.2.1.1 `akaroa_import lib_akaroa`

Definition at line 10 of file `akaroa_import.cc`.

Referenced by `spectral_analysis_QE::checkQuantiles()`.

9.3 basic.cc File Reference

```
#include "basic.h"
```

Include dependency graph for basic.cc:



Variables

- `std::string s_controller` = "controller"
- `std::string s_deterministic_TPD` = "deterministic_TPD"
- `std::string s_sequential_TPD` = "sequential_TPD"
- `std::string s_sequential_batching` = "sequential_batching"
- `std::string s_evolution` = "evolution"
- `std::string s_pooling_QE` = "pooling_QE"
- `std::string s_batch_mean_QE` = "batch_mean_QE"
- `std::string s_spectral_analysis_QE` = "spectral_analysis_QE"
- `std::string s_deterministic_SSC_QE` = "deterministic_SSC_QE"
- `std::string s_confidenceInterval_SSC_QE` = "confidenceInterval_SSC_QE"
- `std::string s_relativeErrorQuantile_SSC_QE` = "relativeErrorQuantile_SSC_QE"
- `std::string s_relativeErrorRange_SSC_QE` = "relativeErrorRange_SSC_QE"
- `std::string s_execute` = "execute"
- `std::string s_replications` = "replications"
- `std::string s_cutoff` = "cutoff"
- `std::string s_permanent` = "permanent"
- `std::string s_start` = "start"
- `std::string s_stop` = "stop"
- `std::string s_ratio` = "ratio"
- `std::string s_ratio_min` = "ratio_min"
- `std::string s_ratio_max` = "ratio_max"
- `std::string s_alpha` = "alpha"
- `std::string s_performance` = "performance"
- `std::string s_limit` = "limit"
- `std::string s_independence` = "independence"
- `std::string s_statistic` = "statistic"
- `std::string s_batch_max` = "batch_max"
- `std::string s_sort` = "sort"
- `std::string s_quantiles_min` = "quantiles_min"
- `std::string s_critical_value` = "critical_value"
- `std::string s_batches` = "batches"
- `std::string s_yes` = "yes"
- `std::string s_no` = "no"
- `std::string s_auto` = "auto"
- `std::string s_fast` = "fast"
- `std::string s_precise` = "precise"
- `std::string s_exact` = "exact"
- `std::string s_mean` = "mean"
- `std::string s_spacing` = "spacing"

- `std::string s_runsUpDown = "runsUpDown"`
- `std::string s_runsAboveBelow = "runsAboveBelow"`
- `std::string s_vonNeumann = "vonNeumann"`
- `std::string s_pearsonStrelen = "pearsonStrelen"`
- `std::string s_pearsonPermutation = "pearsonPermutation"`

9.3.1 Variable Documentation

9.3.1.1 `std::string s_alpha = "alpha"`

Definition at line 55 of file basic.cc.

Referenced by `evolution::evolution()`, `sequential_TPD::printSetting()`, `batching::printSetting()`, `sequential_TPD::settings()`, `spectral_analysis_QE::settings()`, `batch_mean_QE::settings()`, `pooling_QE::settings()`, and `batching::settings()`.

9.3.1.2 `std::string s_auto = "auto"`

Definition at line 68 of file basic.cc.

Referenced by `sequential_TPD::printSetting()`, `sequential_TPD::settings()`, and `batching::settings()`.

9.3.1.3 `std::string s_batch_max = "batch_max"`

Definition at line 60 of file basic.cc.

Referenced by `batching::printSetting()`, and `batching::settings()`.

9.3.1.4 `std::string s_batch_mean_QE = "batch_mean_QE"`

Definition at line 39 of file basic.cc.

Referenced by `method_factory::construct()`, `method_factory::method_factory()`, `batch_mean_QE::printResult()`, `batch_mean_QE::printSetting()`, `batch_mean_QE::printStatus()`, and `batch_mean_QE::settings()`.

9.3.1.5 `std::string s_batches = "batches"`

Definition at line 64 of file basic.cc.

Referenced by `spectral_analysis_QE::printSetting()`, `batch_mean_QE::printSetting()`, `spectral_analysis_QE::settings()`, and `batch_mean_QE::settings()`.

9.3.1.6 `std::string s_confidenceInterval_SSC_QE = "confidenceInterval_SSC_QE"`

Definition at line 42 of file basic.cc.

Referenced by `confidenceInterval_SSC_QE::getName()`, and `quantile_estimation::set_SSC()`.

9.3.1.7 `std::string s_controller = "controller"`

Definition at line 33 of file basic.cc.

9.3.1.8 `std::string s_critical_value = "critical_value"`

Definition at line 63 of file basic.cc.

9.3.1.9 `std::string s_cutoff = "cutoff"`

Definition at line 48 of file basic.cc.

Referenced by `deterministic_TPD::deterministic_TPD()`, and `deterministic_TPD::printSetting()`.

9.3.1.10 `std::string s_deterministic_SSC_QE = "deterministic_SSC_QE"`

Definition at line 41 of file basic.cc.

Referenced by `deterministic_SSC_QE::getName()`, and `quantile_estimation::set_SSC()`.

9.3.1.11 `std::string s_deterministic_TPD = "deterministic_TPD"`

Definition at line 34 of file basic.cc.

Referenced by `method_factory::construct()`, `deterministic_TPD::deterministic_TPD()`, `method_factory::method_factory()`, `deterministic_TPD::printResult()`, `deterministic_TPD::printSetting()`, and `deterministic_TPD::printStatus()`.

9.3.1.12 `std::string s_evolution = "evolution"`

Definition at line 37 of file basic.cc.

Referenced by `method_factory::construct()`, `evolution::evolution()`, `method_factory::method_factory()`, `evolution::printResult()`, `evolution::printSetting()`, and `evolution::printStatus()`.

9.3.1.13 `std::string s_exact = "exact"`

Definition at line 71 of file basic.cc.

Referenced by `sequential_TPD::printDistribution()`, `sequential_TPD::printResult()`, `sequential_TPD::printSetting()`, `sequential_TPD::printStatus()`, and `sequential_TPD::settings()`.

9.3.1.14 `std::string s_execute = "execute"`

Definition at line 46 of file basic.cc.

Referenced by `method_factory::method_factory()`, `sequential_TPD::printSetting()`, `deterministic_TPD::printSetting()`, `evolution::printSetting()`, `spectral_analysis_QE::printSetting()`, `batch_mean_QE::printSetting()`, `pooling_QE::printSetting()`, `batching::printSetting()`, and `quantile_estimation::set_SSC()`.

9.3.1.15 std::string s_fast = "fast"

Definition at line 69 of file basic.cc.

Referenced by sequential_TPD::printDistribution(), sequential_TPD::printResult(), sequential_TPD::printSetting(), and sequential_TPD::printStatus().

9.3.1.16 std::string s_independence = "independence"

Definition at line 58 of file basic.cc.

Referenced by batching::printSetting(), and batching::settings().

9.3.1.17 std::string s_limit = "limit"

Definition at line 57 of file basic.cc.

Referenced by sequential_TPD::printSetting(), and sequential_TPD::settings().

9.3.1.18 std::string s_mean = "mean"

Definition at line 72 of file basic.cc.

Referenced by batching::printSetting(), and batching::settings().

9.3.1.19 std::string s_no = "no"

Definition at line 67 of file basic.cc.

Referenced by evolution::printSetting(), batching::printSetting(), evolution::printStatus(), and batching::settings().

9.3.1.20 std::string s_pearsonPermutation = "pearsonPermutation"

Definition at line 78 of file basic.cc.

Referenced by batching::printSetting(), and batching::settings().

9.3.1.21 std::string s_pearsonStrelen = "pearsonStrelen"

Definition at line 77 of file basic.cc.

Referenced by batching::printSetting(), and batching::settings().

9.3.1.22 std::string s_performance = "performance"

Definition at line 56 of file basic.cc.

Referenced by sequential_TPD::printSetting(), and sequential_TPD::settings().

9.3.1.23 std::string s_permanent = "permanent"

Definition at line 49 of file basic.cc.

Referenced by `evolution::printSetting()`, and `evolution::printStatus()`.

9.3.1.24 `std::string s_pooling_QE = "pooling_QE"`

Definition at line 38 of file `basic.cc`.

Referenced by `method_factory::construct()`, `method_factory::method_factory()`, `pooling_QE::printResult()`, `pooling_QE::printSetting()`, `pooling_QE::printStatus()`, and `pooling_QE::settings()`.

9.3.1.25 `std::string s_precise = "precise"`

Definition at line 70 of file `basic.cc`.

Referenced by `sequential_TPD::printDistribution()`, `sequential_TPD::printResult()`, `sequential_TPD::printSetting()`, `sequential_TPD::printStatus()`, and `sequential_TPD::settings()`.

9.3.1.26 `std::string s_quantiles_min = "quantiles_min"`

Definition at line 62 of file `basic.cc`.

Referenced by `pooling_QE::printSetting()`, and `pooling_QE::settings()`.

9.3.1.27 `std::string s_ratio = "ratio"`

Definition at line 52 of file `basic.cc`.

Referenced by `sequential_TPD::printSetting()`, and `sequential_TPD::settings()`.

9.3.1.28 `std::string s_ratio_max = "ratio_max"`

Definition at line 54 of file `basic.cc`.

Referenced by `sequential_TPD::printSetting()`, and `sequential_TPD::settings()`.

9.3.1.29 `std::string s_ratio_min = "ratio_min"`

Definition at line 53 of file `basic.cc`.

Referenced by `sequential_TPD::printSetting()`, and `sequential_TPD::settings()`.

9.3.1.30 `std::string s_relativeErrorQuantile_SSC_QE = "relativeErrorQuantile_SSC_QE"`

Definition at line 43 of file `basic.cc`.

Referenced by `relativeErrorQuantile_SSC_QE::getName()`, and `quantile_estimation::set_SSC()`.

9.3.1.31 `std::string s_relativeErrorRange_SSC_QE = "relativeErrorRange_SSC_QE"`

Definition at line 44 of file `basic.cc`.

Referenced by `relativeErrorRange_SSC_QE::getName()`, and `quantile_estimation::set_SSC()`.

9.3.1.32 `std::string s_replications = "replications"`

Definition at line 47 of file `basic.cc`.

Referenced by `evolution::evolution()`, `main()`, `sequential_TPD::settings()`, `spectral_analysis_QE::settings()`, `batch_mean_QE::settings()`, and `batching::settings()`.

9.3.1.33 `std::string s_runsAboveBelow = "runsAboveBelow"`

Definition at line 75 of file `basic.cc`.

Referenced by `batching::printSetting()`, and `batching::settings()`.

9.3.1.34 `std::string s_runsUpDown = "runsUpDown"`

Definition at line 74 of file `basic.cc`.

Referenced by `batching::printSetting()`, and `batching::settings()`.

9.3.1.35 `std::string s_sequential_batching = "sequential_batching"`

Definition at line 36 of file `basic.cc`.

Referenced by `method_factory::construct()`, `method_factory::method_factory()`, `batching::printResult()`, `batching::printSetting()`, `batching::printStatus()`, and `batching::settings()`.

9.3.1.36 `std::string s_sequential_TPD = "sequential_TPD"`

Definition at line 35 of file `basic.cc`.

Referenced by `method_factory::construct()`, `method_factory::method_factory()`, `sequential_TPD::printDistribution()`, `sequential_TPD::printResult()`, `sequential_TPD::printSetting()`, `sequential_TPD::printStatus()`, `sequential_TPD::process()`, and `sequential_TPD::settings()`.

9.3.1.37 `std::string s_sort = "sort"`

Definition at line 61 of file `basic.cc`.

Referenced by `batching::printSetting()`, and `batching::settings()`.

9.3.1.38 `std::string s_spacing = "spacing"`

Definition at line 73 of file `basic.cc`.

Referenced by `batching::printSetting()`, and `batching::settings()`.

9.3.1.39 `std::string s_spectral_analysis_QE = "spectral_analysis_QE"`

Definition at line 40 of file `basic.cc`.

Referenced by `method_factory::construct()`, `method_factory::method_factory()`, `spectral_analysis_QE::printResult()`, `spectral_analysis_QE::printSetting()`, `spectral_analysis_QE::printStatus()`, and `spectral_analysis_QE::settings()`.

9.3.1.40 `std::string s_start = "start"`

Definition at line 50 of file `basic.cc`.

Referenced by `evolution::evolution()`, `evolution::printSetting()`, and `evolution::printStatus()`.

9.3.1.41 `std::string s_statistic = "statistic"`

Definition at line 59 of file `basic.cc`.

Referenced by `batching::printSetting()`, and `batching::settings()`.

9.3.1.42 `std::string s_stop = "stop"`

Definition at line 51 of file `basic.cc`.

Referenced by `evolution::evolution()`, `evolution::printSetting()`, and `evolution::printStatus()`.

9.3.1.43 `std::string s_vonNeumann = "vonNeumann"`

Definition at line 76 of file `basic.cc`.

Referenced by `batching::printSetting()`, and `batching::settings()`.

9.3.1.44 `std::string s_yes = "yes"`

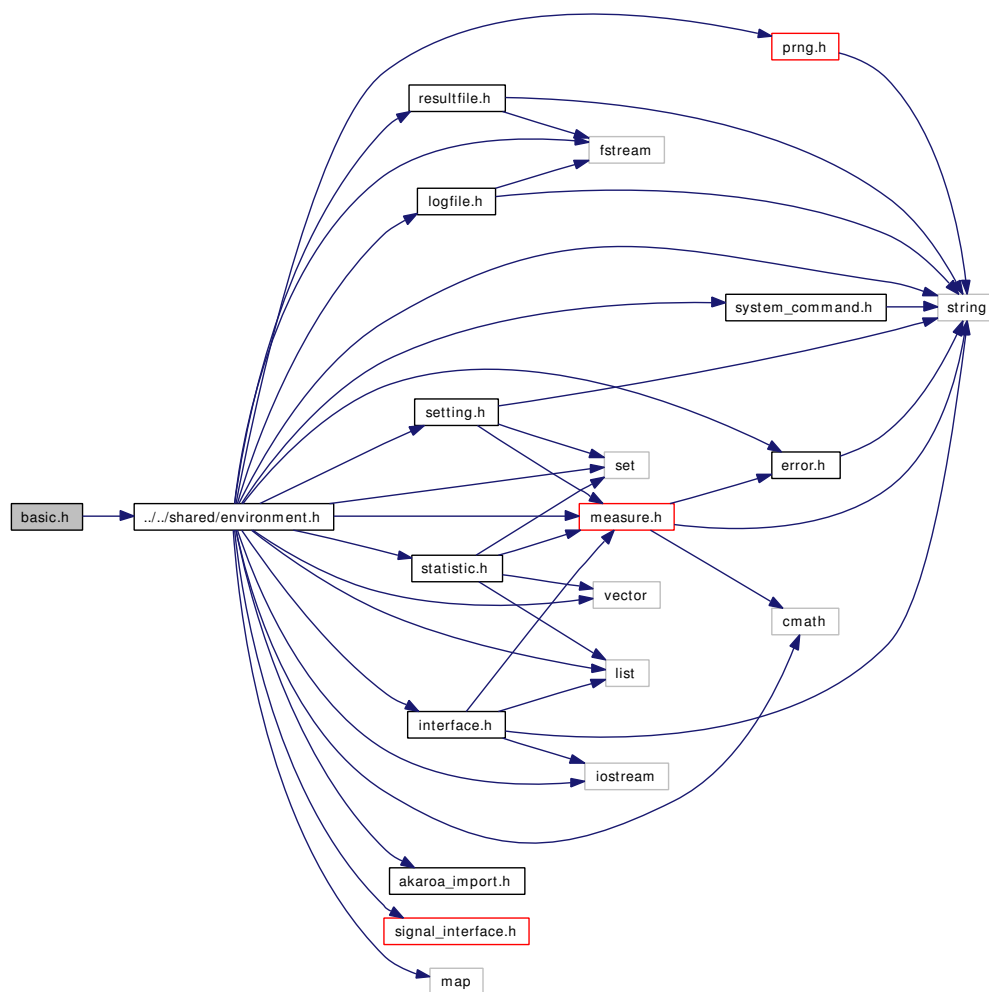
Definition at line 66 of file `basic.cc`.

Referenced by `method_factory::method_factory()`, `sequential_TPD::printSetting()`, `deterministic_TPD::printSetting()`, `evolution::printSetting()`, `spectral_analysis_QE::printSetting()`, `batch_mean_QE::printSetting()`, `pooling_QE::printSetting()`, `batching::printSetting()`, `evolution::printStatus()`, `quantile_estimation::set_SSC()`, and `batching::settings()`.

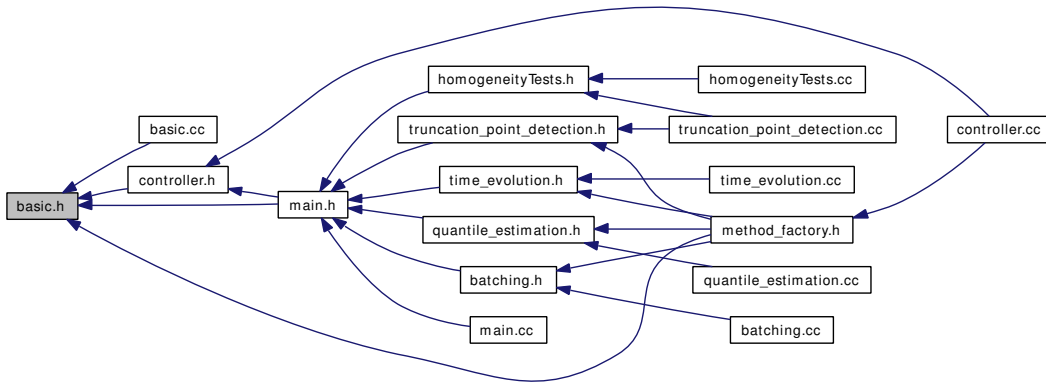
9.4 basic.h File Reference

```
#include "../../shared/environment.h"
```

Include dependency graph for basic.h:



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



Data Structures

- class `outputAnalyser`

Enumerations

- enum `TypeOfMethod` {
 EVOLUTION, IDENTICAL, INDEPENDENT, ESTIMATOR,
 NON }

Variables

- `std::string s_controller`
- `std::string s_deterministic_TPD`
- `std::string s_sequential_TPD`
- `std::string s_sequential_batching`
- `std::string s_evolution`
- `std::string s_pooling_QE`
- `std::string s_batch_mean_QE`
- `std::string s_spectral_analysis_QE`
- `std::string s_deterministic_SSC_QE`
- `std::string s_confidenceInterval_SSC_QE`
- `std::string s_relativeErrorQuantile_SSC_QE`
- `std::string s_relativeErrorRange_SSC_QE`
- `std::string s_execute`
- `std::string s_replications`
- `std::string s_cutoff`
- `std::string s_permanent`
- `std::string s_start`
- `std::string s_stop`
- `std::string s_ratio`
- `std::string s_ratio_min`
- `std::string s_ratio_max`
- `std::string s_alpha`

- std::string s_performance
- std::string s_limit
- std::string s_independence
- std::string s_statistic
- std::string s_batch_max
- std::string s_sort
- std::string s_quantiles_min
- std::string s_critical_value
- std::string s_batches
- std::string s_yes
- std::string s_no
- std::string s_auto
- std::string s_fast
- std::string s_precise
- std::string s_exact
- std::string s_mean
- std::string s_spacing
- std::string s_runsUpDown
- std::string s_runsAboveBelow
- std::string s_vonNeumann
- std::string s_pearsonStrelen
- std::string s_pearsonPermutation

9.4.1 Enumeration Type Documentation

9.4.1.1 enum TypeOfMethod

Enumerator:

EVOLUTION
IDENTICAL
INDEPENDENT
ESTIMATOR
NON

Definition at line 6 of file basic.h.

9.4.2 Variable Documentation

9.4.2.1 std::string s_alpha

Definition at line 55 of file basic.cc.

Referenced by evolution::evolution(), batching::printSetting(), sequential_TPD::printSetting(), batching::settings(), pooling_QE::settings(), batch_mean_QE::settings(), spectral_analysis_QE::settings(), and sequential_TPD::settings().

9.4.2.2 std::string s_auto

Definition at line 68 of file basic.cc.

Referenced by sequential_TPD::printSetting(), batching::settings(), and sequential_TPD::settings().

9.4.2.3 std::string s_batch_max

Definition at line 60 of file basic.cc.

Referenced by batching::printSetting(), and batching::settings().

9.4.2.4 std::string s_batch_mean_QE

Definition at line 39 of file basic.cc.

Referenced by method_factory::construct(), method_factory::method_factory(), batch_mean_QE::printResult(), batch_mean_QE::printSetting(), batch_mean_QE::printStatus(), and batch_mean_QE::settings().

9.4.2.5 std::string s_batches

Definition at line 64 of file basic.cc.

Referenced by batch_mean_QE::printSetting(), spectral_analysis_QE::printSetting(), batch_mean_QE::settings(), and spectral_analysis_QE::settings().

9.4.2.6 std::string s_confidenceInterval_SSC_QE

Definition at line 42 of file basic.cc.

Referenced by confidenceInterval_SSC_QE::getName(), and quantile_estimation::set_SSC().

9.4.2.7 std::string s_controller

Definition at line 33 of file basic.cc.

9.4.2.8 std::string s_critical_value

Definition at line 63 of file basic.cc.

9.4.2.9 std::string s_cutoff

Definition at line 48 of file basic.cc.

Referenced by deterministic_TPD::deterministic_TPD(), and deterministic_TPD::printSetting().

9.4.2.10 std::string s_deterministic_SSC_QE

Definition at line 41 of file basic.cc.

Referenced by `deterministic_SSC_QE::getName()`, and `quantile_estimation::set_SSC()`.

9.4.2.11 `std::string s_deterministic_TPD`

Definition at line 34 of file `basic.cc`.

Referenced by `method_factory::construct()`, `deterministic_TPD::deterministic_TPD()`, `method_factory::method_factory()`, `deterministic_TPD::printResult()`, `deterministic_TPD::printSetting()`, and `deterministic_TPD::printStatus()`.

9.4.2.12 `std::string s_evolution`

Definition at line 37 of file `basic.cc`.

Referenced by `method_factory::construct()`, `evolution::evolution()`, `method_factory::method_factory()`, `evolution::printResult()`, `evolution::printSetting()`, and `evolution::printStatus()`.

9.4.2.13 `std::string s_exact`

Definition at line 71 of file `basic.cc`.

Referenced by `sequential_TPD::printDistribution()`, `sequential_TPD::printResult()`, `sequential_TPD::printSetting()`, `sequential_TPD::printStatus()`, and `sequential_TPD::settings()`.

9.4.2.14 `std::string s_execute`

Definition at line 46 of file `basic.cc`.

Referenced by `method_factory::method_factory()`, `batching::printSetting()`, `pooling_QE::printSetting()`, `batch_mean_QE::printSetting()`, `spectral_analysis_QE::printSetting()`, `evolution::printSetting()`, `deterministic_TPD::printSetting()`, `sequential_TPD::printSetting()`, and `quantile_estimation::set_SSC()`.

9.4.2.15 `std::string s_fast`

Definition at line 69 of file `basic.cc`.

Referenced by `sequential_TPD::printDistribution()`, `sequential_TPD::printResult()`, `sequential_TPD::printSetting()`, and `sequential_TPD::printStatus()`.

9.4.2.16 `std::string s_independence`

Definition at line 58 of file `basic.cc`.

Referenced by `batching::printSetting()`, and `batching::settings()`.

9.4.2.17 `std::string s_limit`

Definition at line 57 of file `basic.cc`.

Referenced by `sequential_TPD::printSetting()`, and `sequential_TPD::settings()`.

9.4.2.18 std::string s_mean

Definition at line 72 of file basic.cc.

Referenced by `batching::printSetting()`, and `batching::settings()`.

9.4.2.19 std::string s_no

Definition at line 67 of file basic.cc.

Referenced by `batching::printSetting()`, `evolution::printSetting()`, `evolution::printStatus()`, and `batching::settings()`.

9.4.2.20 std::string s_pearsonPermutation

Definition at line 78 of file basic.cc.

Referenced by `batching::printSetting()`, and `batching::settings()`.

9.4.2.21 std::string s_pearsonStrelen

Definition at line 77 of file basic.cc.

Referenced by `batching::printSetting()`, and `batching::settings()`.

9.4.2.22 std::string s_performance

Definition at line 56 of file basic.cc.

Referenced by `sequential_TPD::printSetting()`, and `sequential_TPD::settings()`.

9.4.2.23 std::string s_permanent

Definition at line 49 of file basic.cc.

Referenced by `evolution::printSetting()`, and `evolution::printStatus()`.

9.4.2.24 std::string s_pooling_QE

Definition at line 38 of file basic.cc.

Referenced by `method_factory::construct()`, `method_factory::method_factory()`, `pooling_QE::printResult()`, `pooling_QE::printSetting()`, `pooling_QE::printStatus()`, and `pooling_QE::settings()`.

9.4.2.25 std::string s_precise

Definition at line 70 of file basic.cc.

Referenced by `sequential_TPD::printDistribution()`, `sequential_TPD::printResult()`, `sequential_TPD::printSetting()`, `sequential_TPD::printStatus()`, and `sequential_TPD::settings()`.

9.4.2.26 std::string s_quantiles_min

Definition at line 62 of file basic.cc.

Referenced by pooling_QE::printSetting(), and pooling_QE::settings().

9.4.2.27 std::string s_ratio

Definition at line 52 of file basic.cc.

Referenced by sequential_TPD::printSetting(), and sequential_TPD::settings().

9.4.2.28 std::string s_ratio_max

Definition at line 54 of file basic.cc.

Referenced by sequential_TPD::printSetting(), and sequential_TPD::settings().

9.4.2.29 std::string s_ratio_min

Definition at line 53 of file basic.cc.

Referenced by sequential_TPD::printSetting(), and sequential_TPD::settings().

9.4.2.30 std::string s_relativeErrorQuantile_SSC_QE

Definition at line 43 of file basic.cc.

Referenced by relativeErrorQuantile_SSC_QE::getName(), and quantile_estimation::set_SSC().

9.4.2.31 std::string s_relativeErrorRange_SSC_QE

Definition at line 44 of file basic.cc.

Referenced by relativeErrorRange_SSC_QE::getName(), and quantile_estimation::set_SSC().

9.4.2.32 std::string s_replications

Definition at line 47 of file basic.cc.

Referenced by evolution::evolution(), main(), batching::settings(), batch_mean_QE::settings(), spectral_analysis_QE::settings(), and sequential_TPD::settings().

9.4.2.33 std::string s_runsAboveBelow

Definition at line 75 of file basic.cc.

Referenced by batching::printSetting(), and batching::settings().

9.4.2.34 std::string s_runsUpDown

Definition at line 74 of file basic.cc.

Referenced by `batching::printSetting()`, and `batching::settings()`.

9.4.2.35 `std::string s_sequential_batching`

Definition at line 36 of file `basic.cc`.

Referenced by `method_factory::construct()`, `method_factory::method_factory()`, `batching::printResult()`, `batching::printSetting()`, `batching::printStatus()`, and `batching::settings()`.

9.4.2.36 `std::string s_sequential_TPD`

Definition at line 35 of file `basic.cc`.

Referenced by `method_factory::construct()`, `method_factory::method_factory()`, `sequential_TPD::printDistribution()`, `sequential_TPD::printResult()`, `sequential_TPD::printSetting()`, `sequential_TPD::printStatus()`, `sequential_TPD::process()`, and `sequential_TPD::settings()`.

9.4.2.37 `std::string s_sort`

Definition at line 61 of file `basic.cc`.

Referenced by `batching::printSetting()`, and `batching::settings()`.

9.4.2.38 `std::string s_spacing`

Definition at line 73 of file `basic.cc`.

Referenced by `batching::printSetting()`, and `batching::settings()`.

9.4.2.39 `std::string s_spectral_analysis_QE`

Definition at line 40 of file `basic.cc`.

Referenced by `method_factory::construct()`, `method_factory::method_factory()`, `spectral_analysis_QE::printResult()`, `spectral_analysis_QE::printSetting()`, `spectral_analysis_QE::printStatus()`, and `spectral_analysis_QE::settings()`.

9.4.2.40 `std::string s_start`

Definition at line 50 of file `basic.cc`.

Referenced by `evolution::evolution()`, `evolution::printSetting()`, and `evolution::printStatus()`.

9.4.2.41 `std::string s_statistic`

Definition at line 59 of file `basic.cc`.

Referenced by `batching::printSetting()`, and `batching::settings()`.

9.4.2.42 `std::string s_stop`

Definition at line 51 of file `basic.cc`.

Referenced by `evolution::evolution()`, `evolution::printSetting()`, and `evolution::printStatus()`.

9.4.2.43 `std::string s_vonNeumann`

Definition at line 76 of file `basic.cc`.

Referenced by `batching::printSetting()`, and `batching::settings()`.

9.4.2.44 `std::string s_yes`

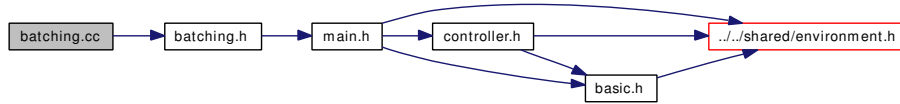
Definition at line 66 of file `basic.cc`.

Referenced by `method_factory::method_factory()`, `batching::printSetting()`, `pooling_QE::printSetting()`, `batch_mean_QE::printSetting()`, `spectral_analysis_QE::printSetting()`, `evolution::printSetting()`, `deterministic_TPD::printSetting()`, `sequential_TPD::printSetting()`, `evolution::printStatus()`, `quantile_estimation::set_SSC()`, and `batching::settings()`.

9.5 batching.cc File Reference

```
#include "batching.h"
```

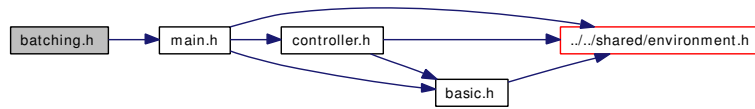
Include dependency graph for batching.cc:



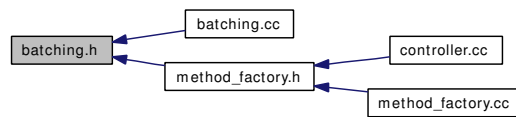
9.6 batching.h File Reference

```
#include "main.h"
```

Include dependency graph for batching.h:



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



Data Structures

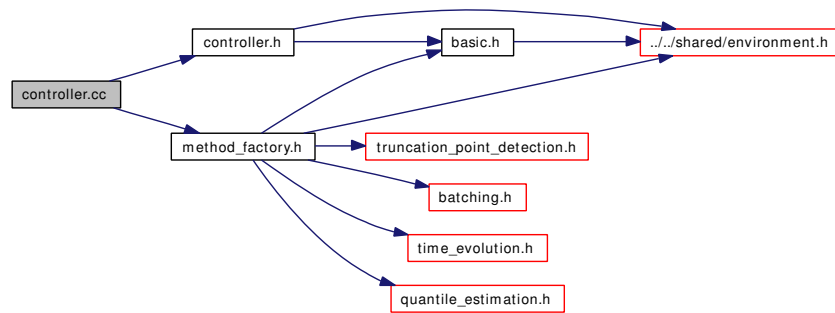
- class `batching`

9.7 controller.cc File Reference

```
#include "controller.h"
```

```
#include "method_factory.h"
```

Include dependency graph for controller.cc:



Variables

- controller lib_controller

9.7.1 Variable Documentation

9.7.1.1 controller lib_controller

Definition at line 6 of file controller.cc.

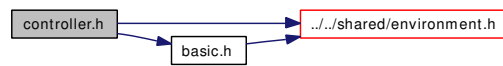
Referenced by main().

9.8 controller.h File Reference

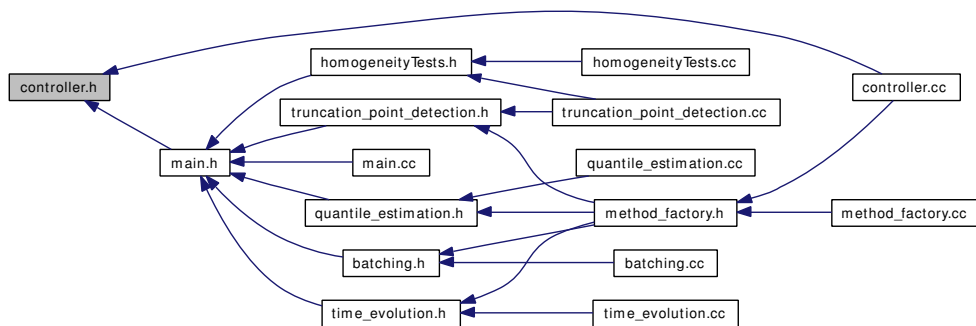
```
#include "../../shared/environment.h"
```

```
#include "basic.h"
```

Include dependency graph for controller.h:



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



Data Structures

- class `controller`

Variables

- `controller` `lib_controller`

9.8.1 Variable Documentation

9.8.1.1 `controller` `lib_controller`

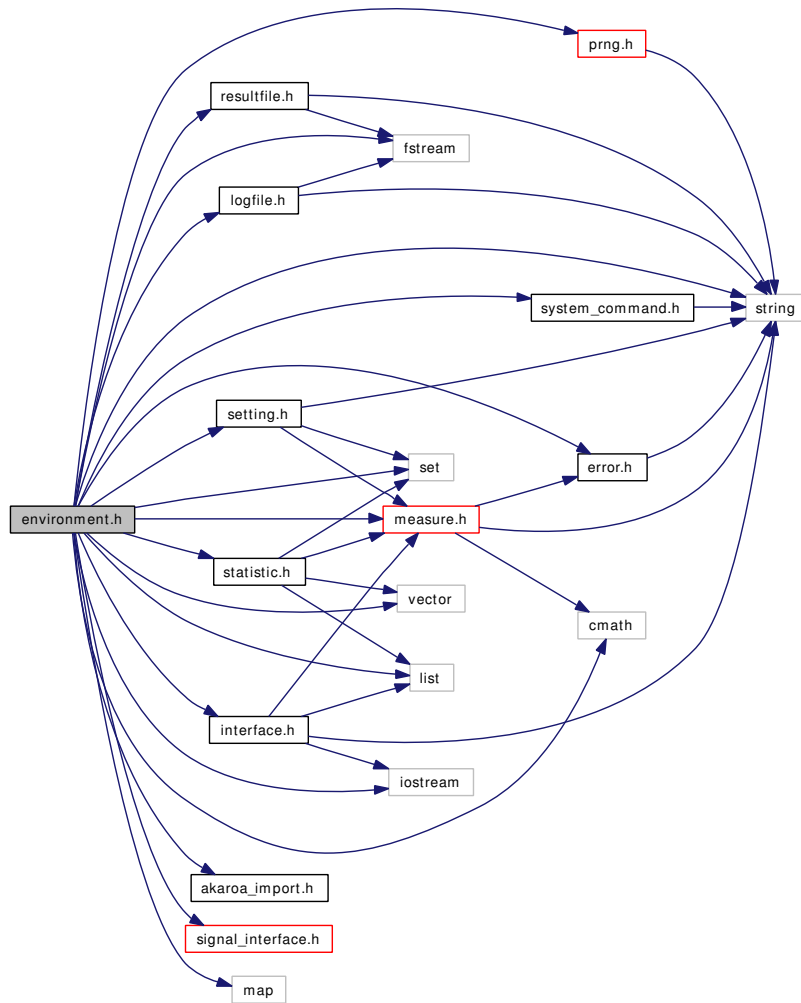
Definition at line 6 of file `controller.cc`.

Referenced by `main()`.

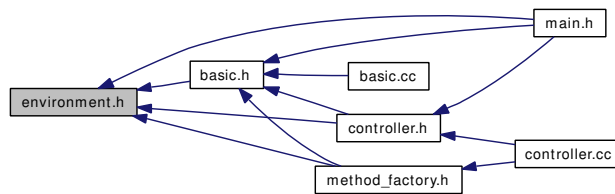
9.9 environment.h File Reference

```
#include "resultfile.h"
#include "akaroa_import.h"
#include "error.h"
#include "system_command.h"
#include "logfile.h"
#include "measure.h"
#include "statistic.h"
#include "interface.h"
#include "setting.h"
#include "signal_interface.h"
#include "prng.h"
#include <list>
#include <vector>
#include <set>
#include <map>
#include <string>
#include <cmath>
#include <iostream>
#include <fstream>
```

Include dependency graph for environment.h:



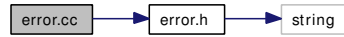
This graph shows which files directly or indirectly include this file:



9.10 error.cc File Reference

```
#include "error.h"
```

Include dependency graph for error.cc:



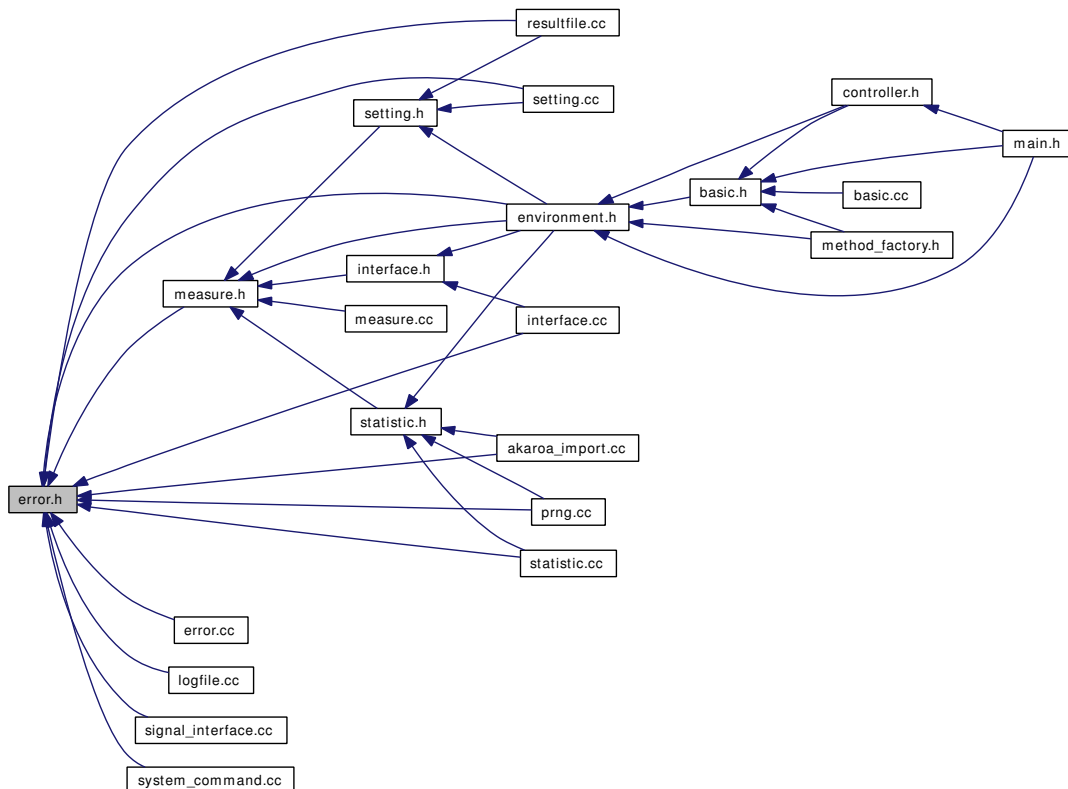
9.11 error.h File Reference

```
#include <string>
```

Include dependency graph for error.h:



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



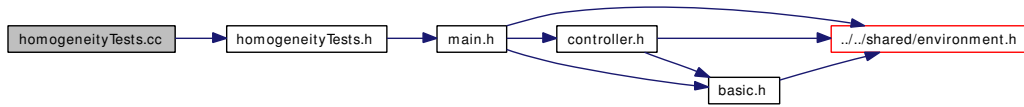
Data Structures

- class `error_in_FCM`

9.12 homogeneityTests.cc File Reference

```
#include "homogeneityTests.h"
```

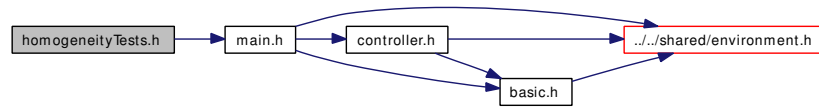
Include dependency graph for homogeneityTests.cc:



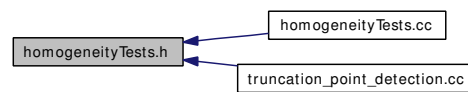
9.13 homogeneityTests.h File Reference

```
#include "main.h"
```

Include dependency graph for homogeneityTests.h:



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



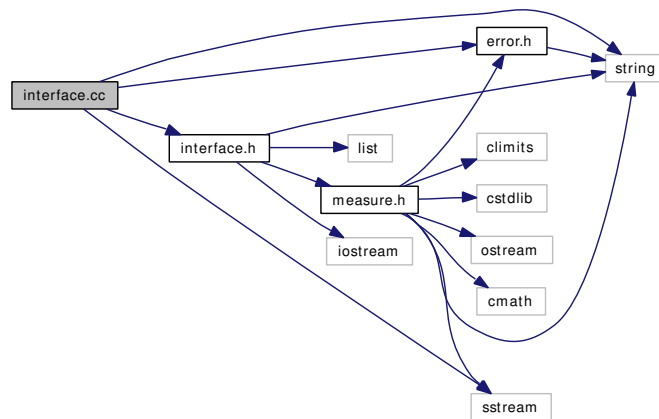
Data Structures

- class **homogeneityTest**
- class **AndersonDarlingKSampleTestEqualECDFSize**
- class **KolmogorovSmirnov2SampleTest**

9.14 interface.cc File Reference

```
#include "interface.h"  
#include <sstream>  
#include <string>  
#include "error.h"
```

Include dependency graph for interface.cc:



Variables

- `interface_singleRun` `lib_singleStream`
- `interface_multipleRuns` `lib_multipleStreams`

9.14.1 Variable Documentation

9.14.1.1 `interface_multipleRuns` `lib_multipleStreams`

Definition at line 64 of file `interface.cc`.

Referenced by `main()`.

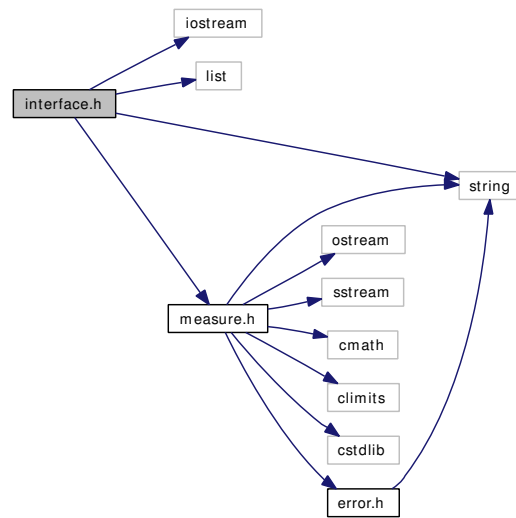
9.14.1.2 `interface_singleRun` `lib_singleStream`

Definition at line 11 of file `interface.cc`.

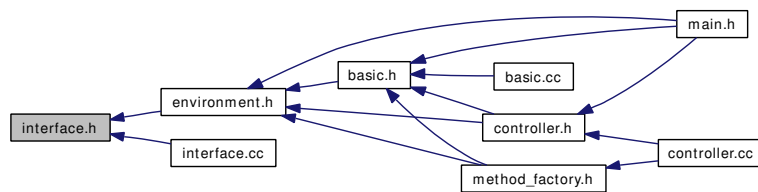
9.15 interface.h File Reference

```
#include <iostream>
#include <list>
#include <string>
#include "measure.h"
```

Include dependency graph for interface.h:



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



Data Structures

- class `interface_singleRun`
- class `interface_multipleRuns`

Variables

- `interface_singleRun` `lib_singleStream`
- `interface_multipleRuns` `lib_multipleStreams`

9.15.1 Variable Documentation

9.15.1.1 `interface _multipleRuns lib _multipleStreams`

Definition at line 64 of file `interface.cc`.

Referenced by `main()`.

9.15.1.2 `interface _singleRun lib _singleStream`

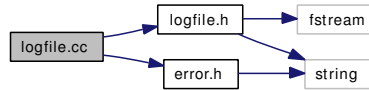
Definition at line 11 of file `interface.cc`.

9.16 logfile.cc File Reference

```
#include "logfile.h"
```

```
#include "error.h"
```

Include dependency graph for logfile.cc:



Functions

- void **logInfo::open** (void)
- void **logInfo::close** (void)

Variables

- `std::ofstream * logfile`

9.16.1 Variable Documentation

9.16.1.1 `std::ofstream* logfile`

Definition at line 6 of file logfile.cc.

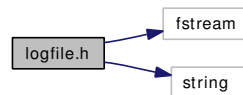
Referenced by `logInfo::close()`, `logInfo::open()`, `sequential_TPD::printResult()`, `deterministic_TPD::printResult()`, `spectral_analysis_QE::printResult()`, `batch_mean_QE::printResult()`, `pooling_QE::printResult()`, `batching::printResult()`, `sequential_TPD::printSetting()`, `deterministic_TPD::printSetting()`, `evolution::printSetting()`, `spectral_analysis_QE::printSetting()`, `batch_mean_QE::printSetting()`, `pooling_QE::printSetting()`, `batching::printSetting()`, `sequential_TPD::printStatus()`, `deterministic_TPD::printStatus()`, `evolution::printStatus()`, `spectral_analysis_QE::printStatus()`, `batch_mean_QE::printStatus()`, `pooling_QE::printStatus()`, `controller::printStatus()`, `batching::printStatus()`, `system_command::printToLogfile()`, `setting::printToLogfile()`, `prng::printToLogfile()`, `sequential_TPD::process()`, `controller::process()`, and `batching::testBatchStatistic()`.

9.17 logfile.h File Reference

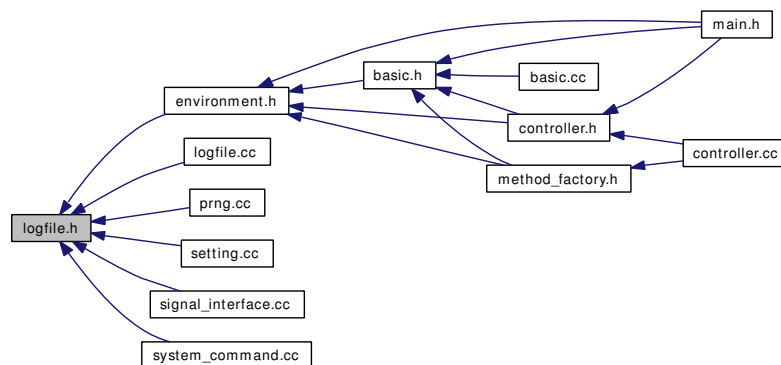
```
#include <fstream>
```

```
#include <string>
```

Include dependency graph for logfile.h:



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



Namespaces

- namespace **logInfo**

Functions

- void **logInfo::open** (void)
- void **logInfo::close** (void)

Variables

- `std::ofstream * logfile`

9.17.1 Variable Documentation

9.17.1.1 `std::ofstream* logfile`

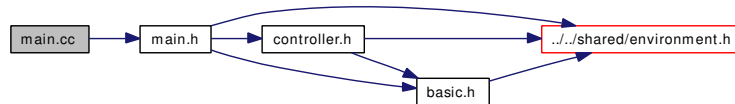
Definition at line 6 of file logfile.cc.

Referenced by `logInfo::close()`, `logInfo::open()`, `batching::printResult()`, `pooling_QE::printResult()`, `batch_mean_QE::printResult()`, `spectral_analysis_QE::printResult()`, `deterministic_TPD::printResult()`, `sequential_TPD::printResult()`, `batching::printSetting()`, `pooling_QE::printSetting()`, `batch_mean_QE::printSetting()`, `spectral_analysis_QE::printSetting()`, `evolution::printSetting()`, `deterministic_TPD::printSetting()`, `sequential_TPD::printSetting()`, `batching::printStatus()`, `controller::printStatus()`, `pooling_QE::printStatus()`, `batch_mean_QE::printStatus()`, `spectral_analysis_QE::printStatus()`, `evolution::printStatus()`, `deterministic_TPD::printStatus()`, `sequential_TPD::printStatus()`, `prng::printToLogfile()`, `setting::printToLogfile()`, `system_command::printToLogfile()`, `controller::process()`, `sequential_TPD::process()`, and `batching::testBatchStatistic()`.

9.18 main.cc File Reference

```
#include "main.h"
```

Include dependency graph for main.cc:



Functions

- `int main (int argc, char *argv[])`

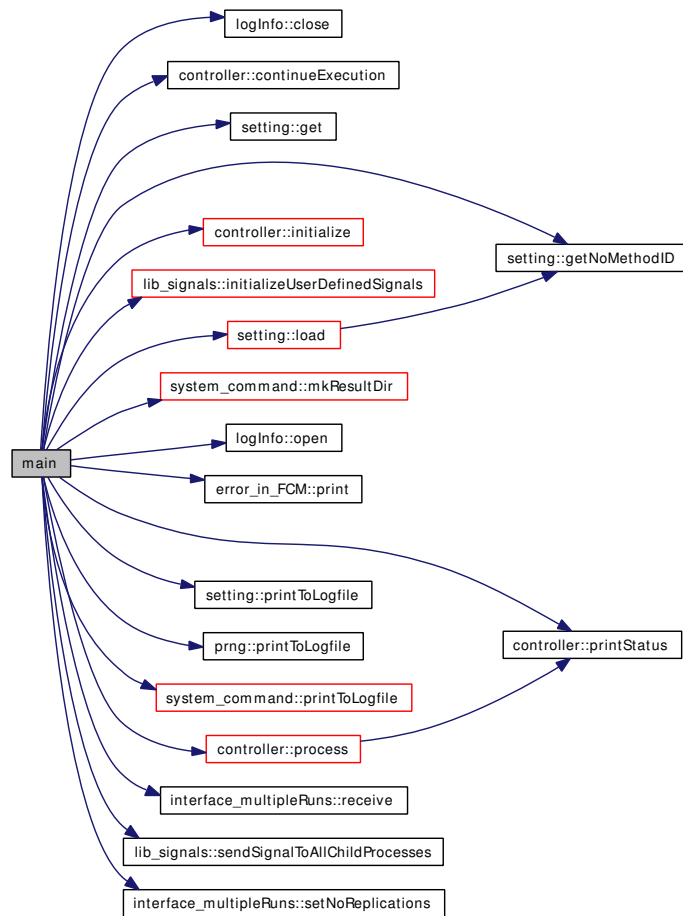
9.18.1 Function Documentation

9.18.1.1 `int main (int argc, char * argv[])`

Definition at line 4 of file main.cc.

References `logInfo::close()`, `controller::continueExecution()`, `lib_signals::continueExecution`, `CONTINUOUS`, `setting::get()`, `setting::getNoMethodID()`, `INDEX`, `controller::initialize()`, `lib_signals::initializeUserDefinedSignals()`, `lib_controller`, `lib_multipleStreams`, `lib_prng`, `lib_setting`, `lib_system`, `setting::load()`, `system_command::mkResultDir()`, `logInfo::open()`, `error_in_FCM::print()`, `controller::printStatus()`, `setting::printToLogfile()`, `prng::printToLogfile()`, `system_command::printToLogfile()`, `controller::process()`, `interface_multipleRuns::receive()`, `s_replications`, `lib_signals::sendSignalToAllChildProcesses()`, and `interface_multipleRuns::setNoReplications()`.

Here is the call graph for this function:



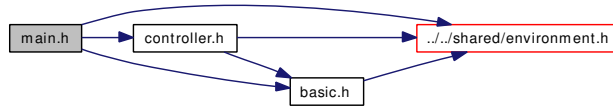
9.19 main.h File Reference

```
#include "../../shared/environment.h"
```

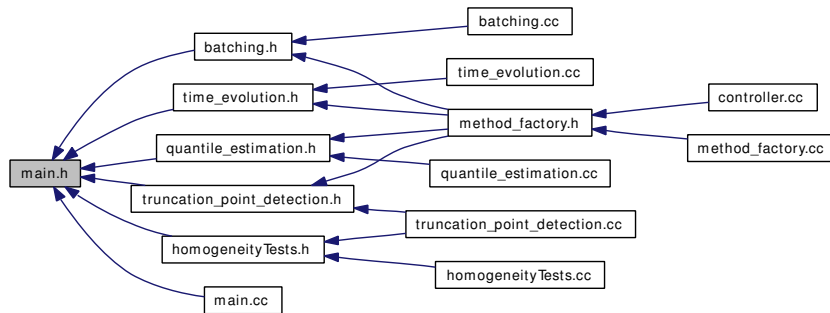
```
#include "controller.h"
```

```
#include "basic.h"
```

Include dependency graph for main.h:



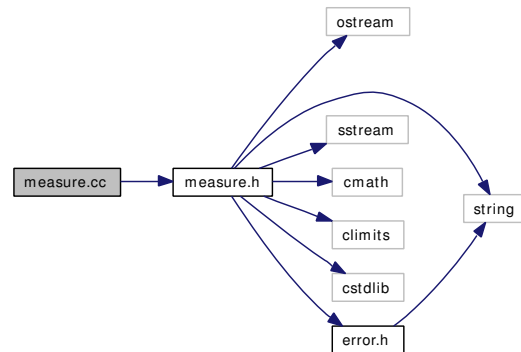
This graph shows which files directly or indirectly include this file:



9.20 measure.cc File Reference

```
#include "measure.h"
```

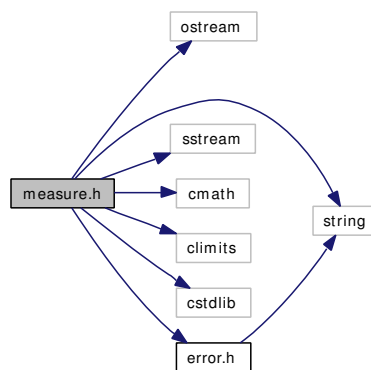
Include dependency graph for measure.cc:



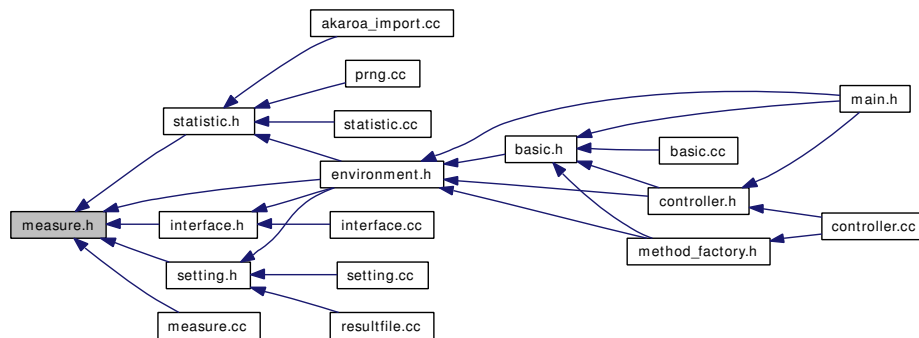
9.21 measure.h File Reference

```
#include <ostream>
#include <string>
#include <sstream>
#include <cmath>
#include <climits>
#include <cstdlib>
#include "error.h"
```

Include dependency graph for measure.h:



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



Defines

- #define **CONTINUOUS** long double
- #define **DISCRETE** long long
- #define **INDEX** unsigned long long

Functions

- CONTINUOUS **DtoC** (const DISCRETE &p)

- CONTINUOUS `ItoC` (const INDEX &p)
- DISCRETE `CtoD` (const CONTINUOUS &p)
- DISCRETE `ItoD` (const INDEX &p)
- INDEX `CtoI` (const CONTINUOUS &p)
- INDEX `DtoI` (const DISCRETE &p)
- `std::string CtoS` (const CONTINUOUS &p)
- `std::string DtoS` (const DISCRETE &p)
- `std::string ItoS` (const INDEX &p)
- CONTINUOUS `StoC` (const `std::string` &p)
- DISCRETE `StoD` (const `std::string` &p)
- INDEX `StoI` (const `std::string` &p)

9.21.1 Define Documentation

9.21.1.1 `#define CONTINUOUS long double`

Definition at line 15 of file `measure.h`.

Referenced by `statistic_collection::binomial()`, `statistic_collection::binomialCoefficient()`, `AndersonDarlingKSampleTestEqualECDFSize::calculateCriticalValue()`, `AndersonDarlingKSampleTestEqualECDFSize::calculateStatistic()`, `quantile_rank::calculateUnbiasedQuantile()`, `spectral_analysis_QE::checkQuantiles()`, `batch_mean_QE::checkQuantiles()`, `pooling_QE::checkQuantiles()`, `statistic_collection::chooseDistribution()`, `statistic_collection::chooseQuantiles()`, `statistic_collection::coth()`, `DtoC()`, `statistic_collection::f_distribution()`, `statistic_collection::finiteSumCorrelationCoefficients_MM1()`, `statistic_collection::generateRandomPermutation()`, `AndersonDarlingKSampleTestEqualECDFSize::getVariance()`, `sequential_TPD::homogeneityTest()`, `statistic_collection::infiniteSumCorrelationCoefficients_MM1()`, `statistic_collection::inv_binomial()`, `statistic_collection::inv_f_distribution()`, `statistic_collection::inv_M_E2_1_response()`, `statistic_collection::inv_M_H2_1_response()`, `statistic_collection::inv_M_M_1_response()`, `statistic_collection::inv_normal()`, `statistic_collection::inv_t_distribution()`, `relativeErrorQuantile_SSC_QE::isFulfilled()`, `ItoC()`, `statistic_collection::M_E2_1_response()`, `statistic_collection::M_H2_1_response()`, `statistic_collection::M_M_1_response()`, `main()`, `statistic_collection::mean()`, `statistic_collection::normal()`, `statistic_collection::pearsonPermutation()`, `statistic_collection::pearsonPermutation_test()`, `statistic_collection::pearsonsCorrelationCoefficient()`, `statistic_collection::pearsonStrelen()`, `statistic_collection::pearsonStrelen_test()`, `statistic_collection::permutationAll()`, `SequentialStoppingCriteria_QE::print()`, `quantile_rank::quantileCDF()`, `statistic_collection::ranks()`, `statistic_collection::siegelsRunTest_large()`, `statistic_collection::spearmanCorrelationCoefficient()`, `statistic_collection::t_distribution()`, `statistic_collection::tanh()`, `batching::testBatchStatistic()`, `statistic_collection::uniform()`, `statistic_collection::vonNeumann()`, `statistic_collection::vonNeumann_statistic()`, and `statistic_collection::vonNeumannsCorrelationCoefficient()`.

9.21.1.2 `#define DISCRETE long long`

Definition at line 16 of file `measure.h`.

Referenced by `statistic_collection::chooseQuantiles()`, `statistic_collection::chooseQuantiles_old()`, `CtoD()`, `statistic_collection::generateRandomPermutation()`, `ItoD()`, and `statistic_collection::siegelsRunTest_small()`.

9.21.1.3 `#define INDEX unsigned long long`

Definition at line 17 of file `measure.h`.

Referenced by `batching::batching()`, `statistic_collection::binomialCoefficient()`, `batching::calculateBatchStatistic()`, `quantile_rank::calculateLowerRank()`, `evolution::calculateQuantiles()`, `AndersonDarlingKSampleTestEqualECDFSize::calculateStatistic()`, `quantile_rank::calculateUpperRank()`, `quantile_rank::cdf()`, `spectral_analysis_QE::checkQuantiles()`, `batch_mean_QE::checkQuantiles()`, `pooling_QE::checkQuantiles()`, `statistic_collection::chooseDistribution()`, `statistic_collection::chooseQuantiles()`, `statistic_collection::chooseQuantiles_old()`, `spectral_analysis_QE::collapse()`, `batch_mean_QE::collapse()`, `batching::collapseBatchStatistic()`, `CtoI()`, `AndersonDarlingKSampleTestEqualECDFSize::debug_VecVec()`, `DtoI()`, `statistic_collection::finiteSumCorrelationCoefficients_MM1()`, `statistic_collection::generateRandomPermutation()`, `KolmogorovSmirnov2SampleTest::KolmogorovSmirnov2SampleTest()`, `main()`, `method_factory::method_factory()`, `statistic_collection::pearsonPermutation_test()`, `statistic_collection::pearsonsCorrelationCoefficient()`, `statistic_collection::pearsonStrelen_test()`, `statistic_collection::permutationAll()`, `sequential_TPD::printDistribution()`, `sequential_TPD::process()`, `spectral_analysis_QE::process()`, `batch_mean_QE::process()`, `statistic_collection::ranks()`, `statistic_collection::runsAboveBelow()`, `statistic_collection::runsUpDown()`, `sequential_TPD::sequential_TPD()`, `statistic_collection::siegelsRunTest_small()`, `AndersonDarlingKSampleTestEqualECDFSize::sortVector()`, `statistic_collection::spearmanCorrelationCoefficient()`, `sequential_TPD::sub_begin()`, `sequential_TPD::sub_collect()`, `sequential_TPD::sub_compare()`, `sequential_TPD::sub_initialize()`, `batching::testBatchStatistic()`, `batching::updateBatchStatistic()`, `statistic_collection::vonNeumann_statistic()`, and `statistic_collection::vonNeumannsCorrelationCoefficient()`.

9.21.2 Function Documentation

9.21.2.1 `DISCRETE CtoD (const CONTINUOUS & p)` [inline]

Definition at line 28 of file `measure.h`.

References `DISCRETE`.

9.21.2.2 `INDEX CtoI (const CONTINUOUS & p)` [inline]

Definition at line 39 of file `measure.h`.

References `INDEX`.

Referenced by `statistic_collection::pearsonPermutation_test()`, and `batching::testBatchStatistic()`.

9.21.2.3 `std::string CtoS (const CONTINUOUS & p)` [inline]

Definition at line 50 of file `measure.h`.

Referenced by `SequentialStoppingCriteria_QE::print()`.

9.21.2.4 `CONTINUOUS DtoC (const DISCRETE & p)` [inline]

Definition at line 20 of file `measure.h`.

References `CONTINUOUS`.

9.21.2.5 INDEX DtoI (const DISCRETE & p) [inline]

Definition at line 45 of file measure.h.

References INDEX.

Referenced by `statistic_collection::binomialCoefficient()`, and `statistic_collection::generateRandomPermutation()`.

9.21.2.6 std::string DtoS (const DISCRETE & p) [inline]

Definition at line 56 of file measure.h.

9.21.2.7 CONTINUOUS ItoC (const INDEX & p) [inline]

Definition at line 24 of file measure.h.

References CONTINUOUS.

Referenced by `statistic_collection::binomialCoefficient()`, `quantile_rank::calculateUnbiasedQuantile()`, `spectral_analysis_QE::checkQuantiles()`, `batch_mean_QE::checkQuantiles()`, `statistic_collection::finiteSumCorrelationCoefficients_MM1()`, `sequential_TPD::sub_compare()`, and `batching::testBatchStatistic()`.

9.21.2.8 DISCRETE ItoD (const INDEX & p) [inline]

Definition at line 34 of file measure.h.

References DISCRETE.

Referenced by `statistic_collection::chooseQuantiles()`, and `statistic_collection::finiteSumCorrelationCoefficients_MM1()`.

9.21.2.9 std::string ItoS (const INDEX & p) [inline]

Definition at line 62 of file measure.h.

Referenced by `SequentialStoppingCriteria_QE::print()`, `sequential_TPD::printResult()`, `batching::printResult()`, and `sequential_TPD::printSetting()`.

9.21.2.10 CONTINUOUS StoC (const std::string & p) [inline]

Definition at line 68 of file measure.h.

Referenced by `settingEntry::getValueContinuous()`.

9.21.2.11 DISCRETE StoD (const std::string & p) [inline]

Definition at line 72 of file measure.h.

Referenced by `settingEntry::getValueDiscrete()`.

9.21.2.12 INDEX StoI (const std::string & *p*) [inline]

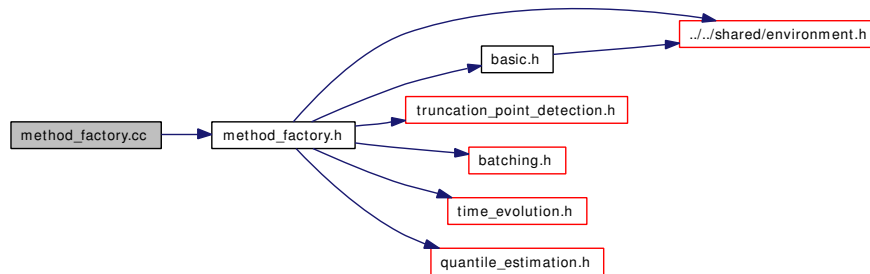
Definition at line 76 of file measure.h.

Referenced by settingEntry::getValueIndex().

9.22 method_factory.cc File Reference

```
#include "method_factory.h"
```

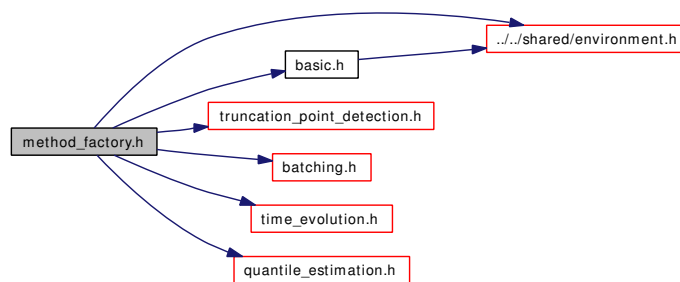
Include dependency graph for method_factory.cc:



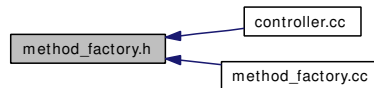
9.23 method_factory.h File Reference

```
#include "../../shared/environment.h"
#include "basic.h"
#include "truncation_point_detection.h"
#include "batching.h"
#include "time_evolution.h"
#include "quantile_estimation.h"
```

Include dependency graph for method_factory.h:



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



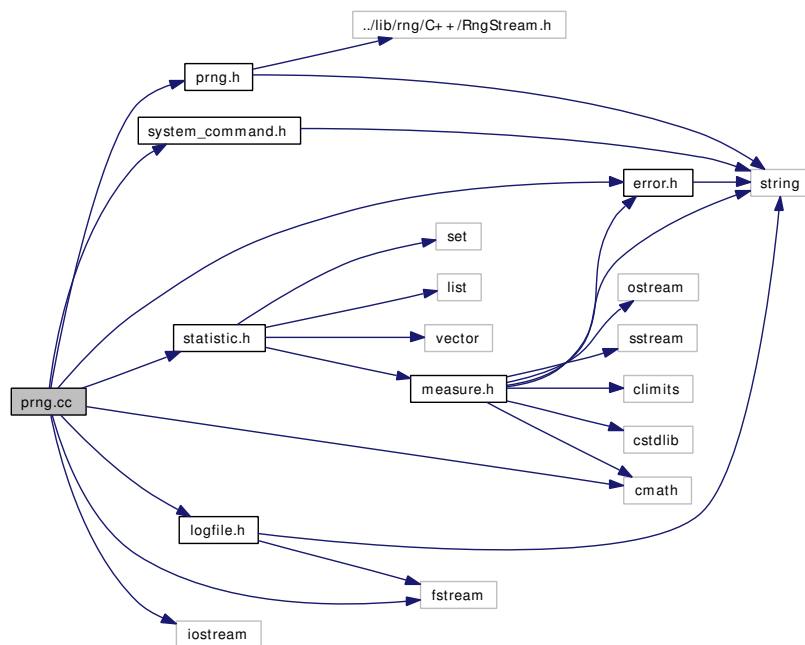
Data Structures

- class `method_factory`

9.24 prng.cc File Reference

```
#include "prng.h"
#include "system_command.h"
#include "error.h"
#include "statistic.h"
#include "logfile.h"
#include <cmath>
#include <fstream>
#include <iostream>
```

Include dependency graph for prng.cc:



Functions

- RngStream **LEcuyer** ("myStream")

Variables

- prng lib_prng

9.24.1 Function Documentation

9.24.1.1 RngStream LEcuyer ("myStream")

9.24.2 Variable Documentation

9.24.2.1 prng lib_prng

Definition at line 193 of file prng.cc.

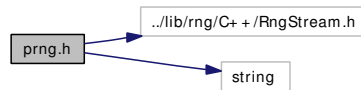
Referenced by `statistic_collection::generateRandomPermutation()`, `main()`, and `sequential_TPD::sub_compare()`.

9.25 prng.h File Reference

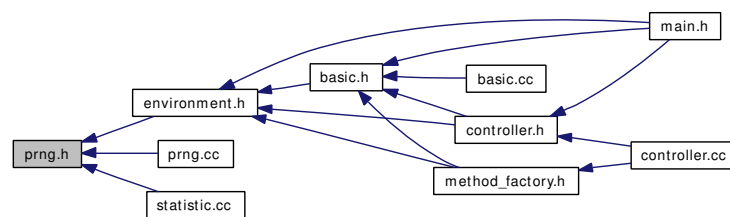
```
#include "../lib/rng/C++/RngStream.h"
```

```
#include <string>
```

Include dependency graph for prng.h:



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



Data Structures

- class `prng`

Variables

- `prng lib_prng`

9.25.1 Variable Documentation

9.25.1.1 `prng lib_prng`

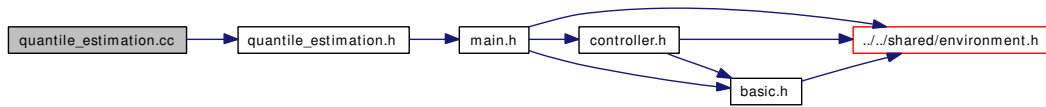
Definition at line 193 of file `prng.cc`.

Referenced by `statistic_collection::generateRandomPermutation()`, `main()`, and `sequential_TPD::sub_compare()`.

9.26 quantile_estimation.cc File Reference

```
#include "quantile_estimation.h"
```

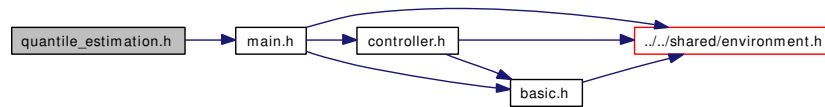
Include dependency graph for quantile_estimation.cc:



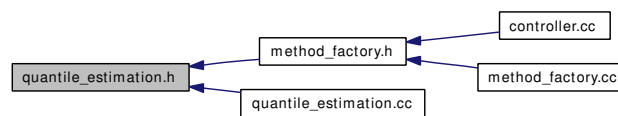
9.27 quantile_estimation.h File Reference

```
#include "main.h"
```

Include dependency graph for quantile_estimation.h:



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



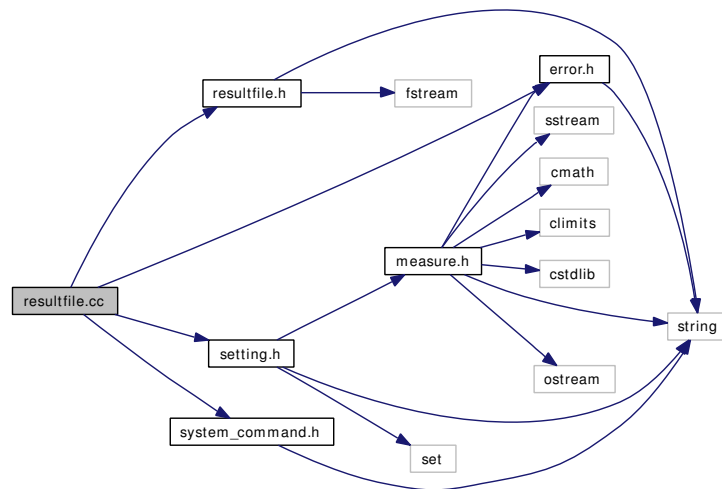
Data Structures

- class `quantile_estimation`
- class `pooling_QE`
- class `batch_mean_QE`
- class `spectral_analysis_QE`
- class `SequentialStoppingCriteria_QE`
- struct `SequentialStoppingCriteria_QE::estimate`
- class `deterministic_SSC_QE`
- class `confidenceInterval_SSC_QE`
- class `relativeErrorQuantile_SSC_QE`
- class `relativeErrorRange_SSC_QE`

9.28 resultfile.cc File Reference

```
#include "resultfile.h"
#include "error.h"
#include "setting.h"
#include "system_command.h"
```

Include dependency graph for resultfile.cc:



Variables

- **resultInfo** resultfile

9.28.1 Variable Documentation

9.28.1.1 resultInfo resultfile

Definition at line 9 of file resultfile.cc.

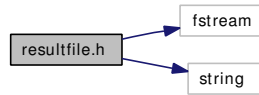
Referenced by SequentialStoppingCriteria_QE::print(), sequential_TPD::printResult(), and batching::printResult().

9.29 resultfile.h File Reference

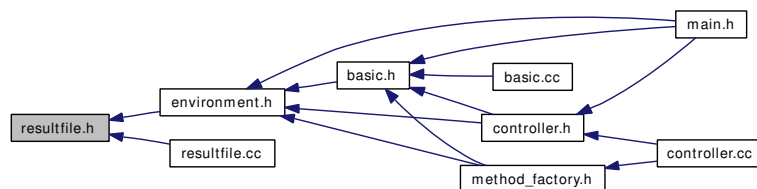
```
#include <fstream>
```

```
#include <string>
```

Include dependency graph for resultfile.h:



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



Data Structures

- class `resultInfo`

Variables

- `resultInfo resultfile`

9.29.1 Variable Documentation

9.29.1.1 `resultInfo resultfile`

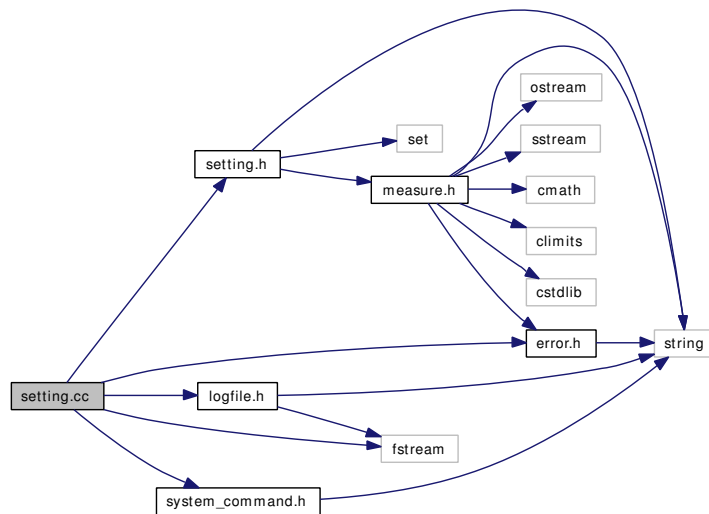
Definition at line 9 of file `resultfile.cc`.

Referenced by `SequentialStoppingCriteria_QE::print()`, `batching::printResult()`, and `sequential_TPD::printResult()`.

9.30 setting.cc File Reference

```
#include "setting.h"
#include "error.h"
#include "logfile.h"
#include "system_command.h"
#include <fstream>
```

Include dependency graph for setting.cc:



Variables

- setting lib_setting

9.30.1 Variable Documentation

9.30.1.1 setting lib_setting

Definition at line 10 of file setting.cc.

Referenced by `deterministic_TPD::deterministic_TPD()`, `evolution::evolution()`, `main()`, `method_factory::method_factory()`, `resultInfo::print()`, `quantile_estimation::set_SSC()`, `sequential_TPD::settings()`, `spectral_analysis_QE::settings()`, `batch_mean_QE::settings()`, `pooling_QE::settings()`, and `batching::settings()`.

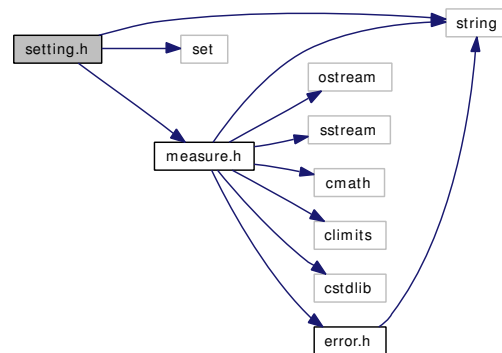
9.31 setting.h File Reference

```
#include <string>
```

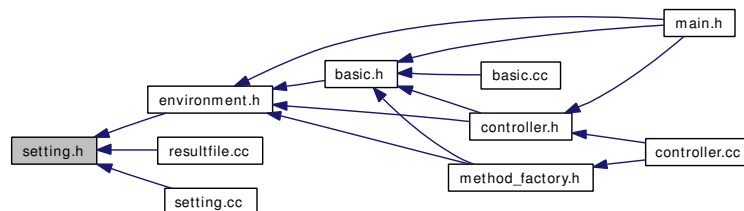
```
#include <set>
```

```
#include "measure.h"
```

Include dependency graph for setting.h:



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



Data Structures

- class `settingEntry`
- class `setting`

Variables

- `setting lib_setting`

9.31.1 Variable Documentation

9.31.1.1 setting lib_setting

Definition at line 10 of file setting.cc.

Referenced by `deterministic_TPD::deterministic_TPD()`, `evolution::evolution()`, `main()`, `method_factory::method_factory()`, `resultInfo::print()`, `quantile_estimation::set_SSC()`, `batching::settings()`, `pooling_QE::settings()`, `batch_mean_QE::settings()`, `spectral_analysis_QE::settings()`, and `sequential_TPD::settings()`.

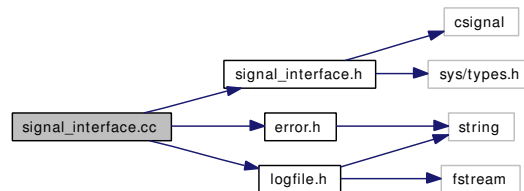
9.32 signal_interface.cc File Reference

```
#include "signal_interface.h"
```

```
#include "error.h"
```

```
#include "logfile.h"
```

Include dependency graph for signal_interface.cc:



Namespaces

- namespace `lib_signals`

Functions

- void `lib_signals::initializeUserDefinedSignals` (void)
- void `lib_signals::signal_stop` (int signr)
- void `lib_signals::signal_ignore` (int signr)
- void `lib_signals::registerChildProcess` (pid_t newProcess)
- void `lib_signals::sendSignalToAllChildProcesses` (int signr)

Variables

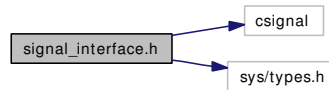
- bool `lib_signals::continueExecution` = true
- unsigned int `lib_signals::actNoChildProcesses` = 0
- const unsigned int `lib_signals::maxNoChildProcesses` = 1024
- pid_t `lib_signals::ChildProcessPIDs` [maxNoChildProcesses]

9.33 signal_interface.h File Reference

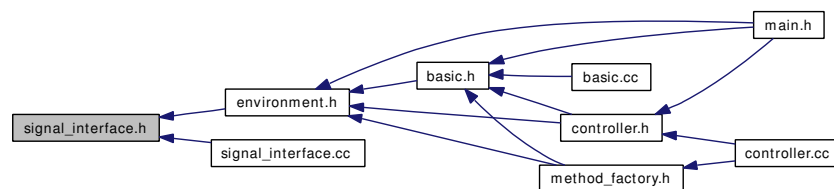
```
#include <csignal>
```

```
#include <sys/types.h>
```

Include dependency graph for signal_interface.h:



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



Namespaces

- namespace `lib_signals`

Functions

- void `lib_signals::initializeUserDefinedSignals` (void)
- void `lib_signals::signal_stop` (int signr)
- void `lib_signals::signal_ignore` (int signr)
- void `lib_signals::registerChildProcess` (pid_t newProcess)
- void `lib_signals::sendSignalToAllChildProcesses` (int signr)

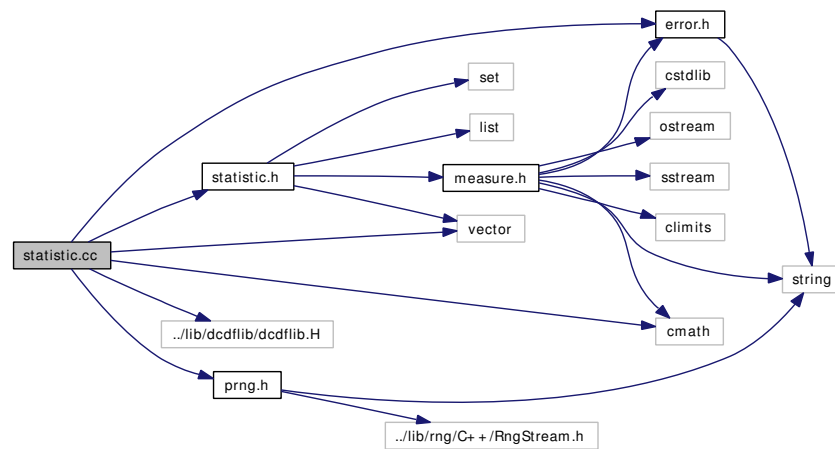
Variables

- bool `lib_signals::continueExecution`
- const unsigned int `lib_signals::maxNoChildProcesses`

9.34 statistic.cc File Reference

```
#include "statistic.h"
#include <cmath>
#include <vector>
#include "../lib/dcdfplib/dcdfplib.H"
#include "error.h"
#include "prng.h"
```

Include dependency graph for statistic.cc:



Variables

- `statistic_collection lib_statistic`

9.34.1 Variable Documentation

9.34.1.1 `statistic_collection lib_statistic`

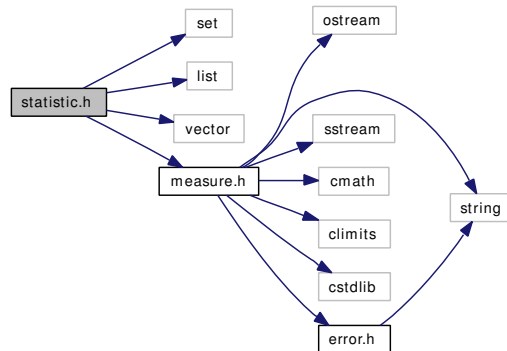
Definition at line 13 of file `statistic.cc`.

Referenced by `spectral_analysis_QE::checkQuantiles()`, `batch_mean_QE::checkQuantiles()`, `pooling_QE::checkQuantiles()`, `prng::draw_normal()`, `evolution::evolution()`, `quantile_rank::quantileCDF()`, and `batching::testBatchStatistic()`.

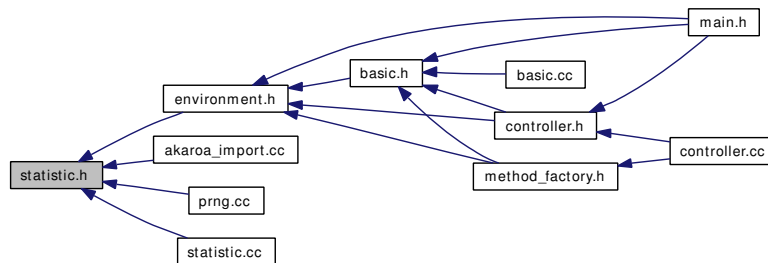
9.35 statistic.h File Reference

```
#include <set>
#include <list>
#include <vector>
#include "measure.h"
```

Include dependency graph for statistic.h:



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



Data Structures

- class `statistic_collection`
- class `quantile_rank`

Enumerations

- enum `distribution` { `UNSPECIFIED` = 0, `UNIFORM`, `EXPONENTIAL`, `NORMAL` }

Variables

- `statistic_collection` `lib_statistic`

9.35.1 Enumeration Type Documentation

9.35.1.1 enum distribution

Enumerator:

UNSPECIFIED

UNIFORM

EXPONENTIAL

NORMAL

Definition at line 13 of file statistic.h.

9.35.2 Variable Documentation

9.35.2.1 statistic_collection lib_statistic

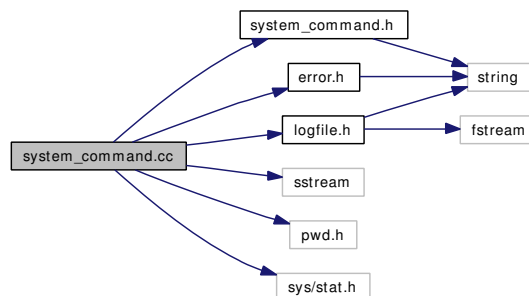
Definition at line 13 of file statistic.cc.

Referenced by pooling_QE::checkQuantiles(), batch_mean_QE::checkQuantiles(), spectral_analysis_QE::checkQuantiles(), prng::draw_normal(), evolution::evolution(), quantile_rank::quantileCDF(), and batching::testBatchStatistic().

9.36 system_command.cc File Reference

```
#include "system_command.h"  
#include "error.h"  
#include "logfile.h"  
#include <sstream>  
#include <pwd.h>  
#include <sys/stat.h>
```

Include dependency graph for system_command.cc:



Variables

- system_command lib_system

9.36.1 Variable Documentation

9.36.1.1 system_command lib_system

Definition at line 11 of file system_command.cc.

Referenced by setting::load(), main(), resultInfo::print(), SequentialStoppingCriteria_QE::print(), sequential_TPD::printDistribution(), evolution::printResult(), and prng::prng().

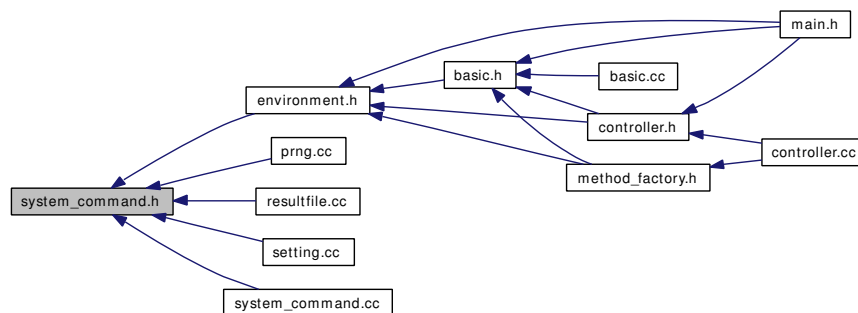
9.37 system_command.h File Reference

```
#include <string>
```

Include dependency graph for system_command.h:



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



Data Structures

- class `system_command`

Variables

- `system_command lib_system`

9.37.1 Variable Documentation

9.37.1.1 system_command lib_system

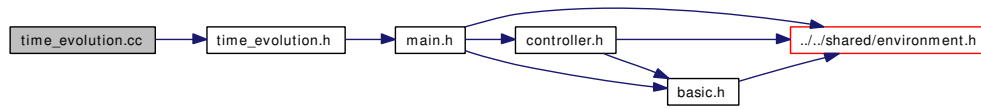
Definition at line 11 of file `system_command.cc`.

Referenced by `setting::load()`, `main()`, `SequentialStoppingCriteria_QE::print()`, `resultInfo::print()`, `sequential_TPD::printDistribution()`, `evolution::printResult()`, and `prng::prng()`.

9.38 time_evolution.cc File Reference

```
#include "time_evolution.h"
```

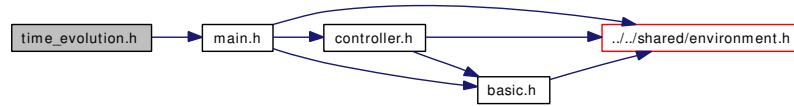
Include dependency graph for time_evolution.cc:



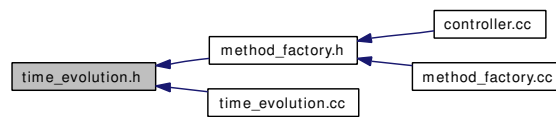
9.39 time_evolution.h File Reference

```
#include "main.h"
```

Include dependency graph for time_evolution.h:



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



Data Structures

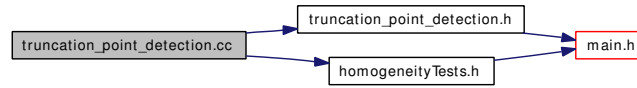
- class **evolution**

9.40 truncation_point_detection.cc File Reference

```
#include "truncation_point_detection.h"
```

```
#include "homogeneityTests.h"
```

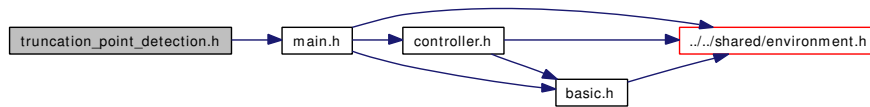
Include dependency graph for truncation_point_detection.cc:



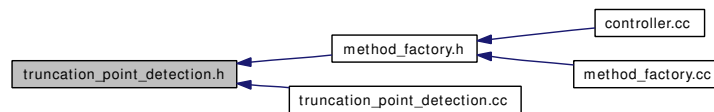
9.41 truncation_point_detection.h File Reference

```
#include "main.h"
```

Include dependency graph for truncation_point_detection.h:



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



Data Structures

- class `truncation_point_detection`
- class `deterministic_TPD`
- class `sequential_TPD`

Index

- `/home/cosc/student/mei16/archiv/projects/distributionalTBD/shared/`
- Directory Reference, 11
- `sequential_TPD`, 128
- `~AndersonDarlingKSampleTestEqualECDFSize` `~setting`
 - setting, 143
- `AndersonDarlingKSampleTestEqualECDFSize`, 24
- `~settingEntry`
 - settingEntry, 147
- `~KolmogorovSmirnov2SampleTest`
 - KolmogorovSmirnov2SampleTest, 81
- `~SequentialStoppingCriteria_QE`
 - spectral_analysis_QE, 153
- `SequentialStoppingCriteria_QE`, 137
- `~system_command`
 - system_command, 177
- `~batch_mean_QE`
 - batch_mean_QE, 31
- `~truncation_point_detection`
 - truncation_point_detection, 182
- `~batching`
 - batching, 39
- `~confidenceInterval_SSC_QE`
 - confidenceInterval_SSC_QE, 47
- `~controller`
 - controller, 51
- `~deterministic_SSC_QE`
 - deterministic_SSC_QE, 55
- `~deterministic_TPD`
 - deterministic_TPD, 61
- `~error_in_FCM`
 - error_in_FCM, 64
- `~evolution`
 - evolution, 68
- `~interface_multipleRuns`
 - interface_multipleRuns, 73
- `~interface_singleRun`
 - interface_singleRun, 76
- `~method_factory`
 - method_factory, 84
- `~outputAnalyser`
 - outputAnalyser, 87
- `~pooling_QE`
 - pooling_QE, 91
- `~prng`
 - prng, 98
- `~quantile_estimation`
 - quantile_estimation, 105
- `~quantile_rank`
 - quantile_rank, 109
- `~relativeErrorQuantile_SSC_QE`
 - relativeErrorQuantile_SSC_QE, 116
- `~relativeErrorRange_SSC_QE`
 - relativeErrorRange_SSC_QE, 121
- `absoluteErrorNeg`
 - SequentialStoppingCriteria_QE::estimate, 140
- `absoluteErrorPos`
 - SequentialStoppingCriteria_QE::estimate, 140
- `accuBatch`
 - batching, 45
- `acosh`
 - statistic_collection, 166
- `acoth`
 - statistic_collection, 166
- `actNoChildProcesses`
 - lib_signals, 16
- `actualSettings`
 - setting, 144
- `addSample`
 - AndersonDarlingKSampleTestEqualECDFSize, 25
- `akaroa_import`, 19
 - CalculatePeriodogram, 21
 - LeastSquaresPolyAt0, 21
 - LogAveragePairsAndOffset, 21
 - LookUp_K_d, 21
 - OrthogonalPolynomialTables, 21
 - OrthogonalPolynomialValues, 22
 - SchrubenStatistic, 20
 - SLOPE_PROTECTION_-CONDITIONAL, 20
 - SLOPE_PROTECTION_OFF, 20
 - SLOPE_PROTECTION_-UNCONDITIONAL, 20

- SpectralVarianceAnalysisOfMean, 20
- SpectralVarianceAnalysisOfProcess, 20
- sqr, 22
- t_distribution, 22
- Z, 22
- akaroa_import.cc, 185
 - K_d_table, 186
 - lib_akaroa, 186
- akaroa_import.h, 187
 - lib_akaroa, 187
- AndersonDarlingKSampleTestEqualECDFSize, 23
 - AndersonDarlingKSampleTestEqualECDFSize, 24
- AndersonDarlingKSampleTestEqualECDFSize
 - ~AndersonDarlingKSampleTestEqualECDFSize, 24
 - addSample, 25
 - AndersonDarlingKSampleTestEqualECDFSize, 24
 - areSamplesIdenticallyDistributed, 25
 - calculateCriticalValue, 26
 - calculateStatistic, 26
 - debug_Set, 26
 - debug_Vec, 26
 - debug_VecVec, 26
 - getVariance, 25
 - m_addedGroups, 27
 - m_addedValues, 27
 - m_alpha, 27
 - m_data, 27
 - m_groupSize, 27
 - m_predefinedVariance, 27
 - m_predefinedVarianceIsValid, 27
 - setAlpha, 25
 - setPredefinedVariance, 25
 - sortVector, 26
- areSamplesIdenticallyDistributed
 - AndersonDarlingKSampleTestEqualECDFSize, 25
- asinh
 - statistic_collection, 166
- atanh
 - statistic_collection, 166
- basic.cc, 188
 - s_alpha, 189
 - s_auto, 189
 - s_batch_max, 189
 - s_batch_mean_QE, 189
 - s_batches, 189
 - s_confidenceInterval_SSC_QE, 189
 - s_controller, 189
 - s_critical_value, 190
 - s_cutoff, 190
 - s_deterministic_SSC_QE, 190
 - s_deterministic_TPD, 190
 - s_evolution, 190
 - s_exact, 190
 - s_execute, 190
 - s_fast, 190
 - s_independence, 191
 - s_limit, 191
 - s_mean, 191
 - s_no, 191
 - s_pearsonPermutation, 191
 - s_pearsonStrelen, 191
 - s_performance, 191
 - s_permanent, 191
 - s_pooling_QE, 192
 - s_precise, 192
 - s_quantiles_min, 192
 - s_ratio, 192
 - s_ratio_max, 192
 - s_ratio_min, 192
 - s_relativeErrorQuantile_SSC_QE, 192
 - s_relativeErrorRange_SSC_QE, 192
 - s_replications, 193
 - s_runsAboveBelow, 193
 - s_runsUpDown, 193
 - s_sequential_batching, 193
 - s_sequential_TPD, 193
 - s_sort, 193
 - s_spacing, 193
 - s_spectral_analysis_QE, 193
 - s_start, 194
 - s_statistic, 194
 - s_stop, 194
 - s_vonNeumann, 194
 - s_yes, 194
- basic.h, 195
 - ESTIMATOR, 197
 - EVOLUTION, 197
 - IDENTICAL, 197
 - INDEPENDENT, 197
 - NON, 197
 - s_alpha, 197
 - s_auto, 197
 - s_batch_max, 198
 - s_batch_mean_QE, 198
 - s_batches, 198
 - s_confidenceInterval_SSC_QE, 198
 - s_controller, 198
 - s_critical_value, 198
 - s_cutoff, 198
 - s_deterministic_SSC_QE, 198
 - s_deterministic_TPD, 199
 - s_evolution, 199

- s_exact, 199
- s_execute, 199
- s_fast, 199
- s_independence, 199
- s_limit, 199
- s_mean, 199
- s_no, 200
- s_pearsonPermutation, 200
- s_pearsonStrelen, 200
- s_performance, 200
- s_permanent, 200
- s_pooling_QE, 200
- s_precise, 200
- s_quantiles_min, 200
- s_ratio, 201
- s_ratio_max, 201
- s_ratio_min, 201
- s_relativeErrorQuantile_SSC_QE, 201
- s_relativeErrorRange_SSC_QE, 201
- s_replications, 201
- s_runsAboveBelow, 201
- s_runsUpDown, 201
- s_sequential_batching, 202
- s_sequential_TPD, 202
- s_sort, 202
- s_spacing, 202
- s_spectral_analysis_QE, 202
- s_start, 202
- s_statistic, 202
- s_stop, 202
- s_vonNeumann, 203
- s_yes, 203
- TypeOfMethod, 197
- batch_mean_QE, 29
 - ~batch_mean_QE, 31
 - batch_mean_QE, 31
 - checkQuantiles, 32
 - collapse, 33
 - getType, 34
 - isReady, 31
 - m_actBatch, 35
 - m_actNoInBatch, 35
 - m_alpha, 35
 - m_batch, 35
 - m_batchNo, 34
 - m_batchSize, 35
 - m_isReady, 34
 - m_mean, 35
 - m_noReplication, 34
 - m_processedIndexes, 35
 - m_SSC, 35
 - printResult, 32
 - printSetting, 32
 - printStatus, 32
 - process, 31
 - set_SSC, 34
 - setBatchSize, 34
 - settings, 33
- batching, 37
 - ~batching, 39
 - accuBatch, 45
 - batching, 39
 - batchStatistic, 45
 - calculateBatchStatistic, 42
 - collapseBatchStatistic, 42
 - ecdf, 45
 - getBatchSize, 41
 - getType, 40
 - isReady, 40
 - m_alpha, 44
 - m_isReady, 44
 - m_maxBatch, 44
 - m_numBatch, 44
 - m_p, 45
 - m_posBatch, 44
 - m_processedIndexes, 45
 - m_sizeBatch, 44
 - m_sortedData, 44
 - m_typeOfBatchStatistic, 44
 - m_typeOfIndependenceTest, 44
 - Mean, 39
 - printResult, 41
 - printSetting, 41
 - printStatus, 41
 - process, 40
 - settings, 43
 - Spacing, 39
 - testBatchStatistic, 42
 - TypeOfBatching, 39
 - updateBatchStatistic, 41
- batching.cc, 204
- batching.h, 205
- batchStatistic
 - batching, 45
- begin
 - sequential_TPD, 127
- binomial
 - statistic_collection, 162
- binomialCoefficient
 - statistic_collection, 172
- C1
 - K_d_entry, 78
- C2
 - K_d_entry, 78
- calculateBatchStatistic
 - batching, 42
- calculateCriticalValue

- AndersonDarlingKSampleTestEqual-ECDFSize, 26
- calculateLowerRank
 - quantile_rank, 112
- CalculatePeriodogram
 - akaroa_import, 21
- calculateQuantiles
 - evolution, 69
- calculateStatistic
 - AndersonDarlingKSampleTestEqual-ECDFSize, 26
- calculateUnbiasedQuantile
 - quantile_rank, 111
- calculateUpperRank
 - quantile_rank, 112
- cdf
 - quantile_rank, 111
- checkQuantiles
 - batch_mean_QE, 32
 - pooling_QE, 93
 - spectral_analysis_QE, 154
- ChildProcessPIDs
 - lib_signals, 16
- chooseDistribution
 - statistic_collection, 161
- chooseQuantiles
 - statistic_collection, 166
- chooseQuantiles_old
 - statistic_collection, 166
- clean
 - setting, 144
- close
 - logInfo, 18
- collapse
 - batch_mean_QE, 33
 - spectral_analysis_QE, 155
- collapseBatchStatistic
 - batching, 42
- collect
 - sequential_TPD, 127
- confidenceInterval_SSC_QE, 46
 - confidenceInterval_SSC_QE, 47
- confidenceInterval_SSC_QE
 - ~confidenceInterval_SSC_QE, 47
 - confidenceInterval_SSC_QE, 47
 - estimateMap, 49
 - getName, 47
 - insert, 48
 - isFulfilled, 47
 - m_counter, 49
 - m_processedIndexes, 49
 - print, 48
 - reset, 48
 - setProcessedIndexes, 48
 - settings, 47
- construct
 - method_factory, 85
- continueExecution
 - controller, 51
 - lib_signals, 16
- CONTINUOUS
 - measure.h, 225
- controller, 50
 - ~controller, 51
 - continueExecution, 51
 - controller, 51
 - executeESTIMATOR, 52
 - executeIDENTICAL, 52
 - executeINDEPENDENT, 52
 - initialize, 51
 - m_batchSize, 52
 - m_batchSizeIsSet, 52
 - m_checkpoint, 52
 - m_methodList, 52
 - m_simulationHorizon, 52
 - printStatus, 51
 - process, 51
- controller.cc, 206
 - lib_controller, 206
- controller.h, 207
 - lib_controller, 207
- cosh
 - statistic_collection, 165
- coth
 - statistic_collection, 166
- CtoD
 - measure.h, 226
- CtoI
 - measure.h, 226
- CtoS
 - measure.h, 226
- d
 - K_d_entry, 78
- debug_Set
 - AndersonDarlingKSampleTestEqual-ECDFSize, 26
- debug_Vec
 - AndersonDarlingKSampleTestEqual-ECDFSize, 26
 - KolmogorovSmirnov2SampleTest, 82
- debug_VecVec
 - AndersonDarlingKSampleTestEqual-ECDFSize, 26
- deterministic_SSC_QE, 54
 - ~deterministic_SSC_QE, 55
 - deterministic_SSC_QE, 55
 - estimateMap, 57

- getName, 55
- insert, 56
- isFulfilled, 55
- m_actTests, 57
- m_counter, 57
- m_maxTests, 57
- m_processedIndexes, 57
- print, 56
- reset, 56
- setProcessedIndexes, 57
- settings, 56
- deterministic_TPD, 59
 - ~deterministic_TPD, 61
- deterministic_TPD, 61
 - getType, 62
 - isReady, 61
 - m_cutoff, 62
 - m_processedIndexes, 62
 - m_resultAlreadyPrinted, 62
 - printResult, 62
 - printSetting, 61
 - printStatus, 62
 - process, 61
- DISCRETE
 - measure.h, 225
- distribution
 - statistic.h, 245
- draw_bool
 - prng, 99
- draw_erlang
 - prng, 100
- draw_exponential
 - prng, 100
- draw_integer
 - prng, 99
- draw_normal
 - prng, 99
- draw_pareto
 - prng, 100
- draw_probability
 - prng, 99
- draw_std_normal
 - prng, 99
- DtoC
 - measure.h, 226
- DtoI
 - measure.h, 226
- DtoS
 - measure.h, 227
- ecdf
 - batching, 45
- environment.h, 208
- error.cc, 210
- error.h, 211
- error_in_FCM, 64
 - ~error_in_FCM, 64
- error_in_FCM, 64
 - m_class, 65
 - m_file, 65
 - m_information, 65
 - m_method, 65
 - print, 65
- estimateMap
 - confidenceInterval_SSC_QE, 49
 - deterministic_SSC_QE, 57
 - relativeErrorQuantile_SSC_QE, 118
 - relativeErrorRange_SSC_QE, 123
 - SequentialStoppingCriteria_QE, 139
- ESTIMATOR
 - basic.h, 197
- EVOLUTION
 - basic.h, 197
- evolution, 66
 - ~evolution, 68
 - calculateQuantiles, 69
 - evolution, 68
 - getType, 68
 - isReady, 68
 - m_actObservations, 70
 - m_alpha, 70
 - m_permanent, 70
 - m_processedIndexes, 71
 - m_replications, 70
 - m_resultfile, 70
 - m_start, 70
 - m_stop, 70
 - m_wantedQuantiles, 70
 - printResult, 69
 - printSetting, 69
 - printStatus, 69
 - process, 69
- exact
 - sequential_TPD, 127
- execute
 - KolmogorovSmirnov2SampleTest, 81
 - system_command, 178
- executeESTIMATOR
 - controller, 52
- executeIDENTICAL
 - controller, 52
- executeINDEPENDENT
 - controller, 52
- EXPONENTIAL
 - statistic.h, 245
- exponential
 - statistic_collection, 163

- f_distribution
 - statistic_collection, 162
- fast
 - sequential_TPD, 127
- finished
 - sequential_TPD, 127
- finiteSumCorrelationCoefficients_MM1
 - statistic_collection, 175
- generateRandomPermutation
 - statistic_collection, 173
- get
 - setting, 143
- getBatchSize
 - batching, 41
- getCallDir
 - system_command, 178
- getCriticalValue
 - KolmogorovSmirnov2SampleTest, 82
- getDate
 - system_command, 177
- getHome
 - system_command, 178
- getHost
 - system_command, 178
- getLowerMeasure
 - quantile_rank, 110
- getLowerRank
 - quantile_rank, 110
- getMaximumDifference
 - KolmogorovSmirnov2SampleTest, 81
- getMeasure
 - quantile_rank, 110
- getMethod
 - settingEntry, 148
- getName
 - confidenceInterval_SSC_QE, 47
 - deterministic_SSC_QE, 55
 - relativeErrorQuantile_SSC_QE, 116
 - relativeErrorRange_SSC_QE, 121
 - SequentialStoppingCriteria_QE, 138
- getNoMethodID
 - setting, 143
- getParameter
 - settingEntry, 148
- getPID
 - system_command, 178
- getPWD
 - system_command, 178
- getRank
 - quantile_rank, 110
- getStandardizedMaximumDifference
 - KolmogorovSmirnov2SampleTest, 81
- getTime
 - system_command, 177
- getTruncationIndex
 - sequential_TPD, 128
- getType
 - batch_mean_QE, 34
 - batching, 40
 - deterministic_TPD, 62
 - evolution, 68
 - outputAnalyser, 87
 - pooling_QE, 93
 - quantile_estimation, 105
 - sequential_TPD, 132
 - spectral_analysis_QE, 156
 - truncation_point_detection, 183
- getUID
 - system_command, 178
- getUnbiasedQuantile
 - quantile_rank, 110
- getUpperMeasure
 - quantile_rank, 110
- getUpperRank
 - quantile_rank, 110
- getUser
 - system_command, 177
- getValue
 - settingEntry, 148
- getValueBoolean
 - settingEntry, 147
- getValueContinuous
 - settingEntry, 148
- getValueDiscrete
 - settingEntry, 147
- getValueIndex
 - settingEntry, 147
- getVariance
 - AndersonDarlingKSSampleTestEqual-ECDFSize, 25
- homogeneityTest, 72
 - sequential_TPD, 132
- homogeneityTests.cc, 212
- homogeneityTests.h, 213
- IDENTICAL
 - basic.h, 197
- independenceTest
 - statistic_collection, 167
- INDEPENDENT
 - basic.h, 197
- INDEX
 - measure.h, 225
- infiniteSumCorrelationCoefficients_MM1
 - statistic_collection, 175
- initialize

- controller, 51
- sequential_TPD, 127
- initializeUserDefinedSignals
 - lib_signals, 15
- insert
 - confidenceInterval_SSC_QE, 48
 - deterministic_SSC_QE, 56
 - relativeErrorQuantile_SSC_QE, 117
 - relativeErrorRange_SSC_QE, 122
 - SequentialStoppingCriteria_QE, 137
- interface.cc, 214
 - lib_multipleStreams, 214
 - lib_singleStream, 214
- interface.h, 215
 - lib_multipleStreams, 216
 - lib_singleStream, 216
- interface_multipleRuns, 73
 - interface_multipleRuns, 73
- interface_multipleRuns
 - ~interface_multipleRuns, 73
 - interface_multipleRuns, 73
 - LDSize, 74
 - m_buffer, 74
 - m_maxPipeBufferSize, 74
 - m_noReplications, 75
 - m_ptr, 74
 - receive, 74
 - send, 74
 - setNoReplications, 74
- interface_singleRun, 76
 - interface_singleRun, 76
- interface_singleRun
 - ~interface_singleRun, 76
 - interface_singleRun, 76
 - LDSize, 77
 - m_buffer, 77
 - m_ptr, 77
 - receive, 77
 - receive_parse, 77
 - send, 77
- inv_binomial
 - statistic_collection, 162
- inv_exponential
 - statistic_collection, 163
- inv_f_distribution
 - statistic_collection, 162
- inv_M_E2_1_response
 - statistic_collection, 164
- inv_M_H2_1_response
 - statistic_collection, 165
- inv_M_M_1_response
 - statistic_collection, 164
- inv_normal
 - statistic_collection, 162
- inv_t_distribution
 - statistic_collection, 163
- inv_uniform
 - statistic_collection, 163
- isFirstGreaterThanSecond
 - KolmogorovSmirnov2SampleTest, 82
- isFulfilled
 - confidenceInterval_SSC_QE, 47
 - deterministic_SSC_QE, 55
 - relativeErrorQuantile_SSC_QE, 116
 - relativeErrorRange_SSC_QE, 121
 - SequentialStoppingCriteria_QE, 138
- isReady
 - batch_mean_QE, 31
 - batching, 40
 - deterministic_TPD, 61
 - evolution, 68
 - outputAnalyser, 87
 - pooling_QE, 92
 - quantile_estimation, 105
 - sequential_TPD, 128
 - spectral_analysis_QE, 153
 - truncation_point_detection, 183
- ItoC
 - measure.h, 227
- ItoD
 - measure.h, 227
- ItoS
 - measure.h, 227
- K
 - K_d_entry, 78
- K_d_entry, 78
 - C1, 78
 - C2, 78
 - d, 78
 - K, 78
- K_d_table
 - akaroa_import.cc, 186
- KolmogorovSmirnov2SampleTest, 80
 - KolmogorovSmirnov2SampleTest, 81
- KolmogorovSmirnov2SampleTest
 - ~KolmogorovSmirnov2SampleTest, 81
 - debug_Vec, 82
 - execute, 81
 - getCriticalValue, 82
 - getMaximumDifference, 81
 - getStandardizedMaximumDifference, 81
 - isFirstGreaterThanSecond, 82
 - KolmogorovSmirnov2SampleTest, 81
 - m_isFirstGreaterThanSecond, 83
 - m_MaximumDifference, 82
 - m_name, 83
 - m_sample1, 82

- m_sample2, 82
- m_StandardizedMaximumDifference, 82
- sortVector, 82
- LDSize
 - interface_multipleRuns, 74
 - interface_singleRun, 77
- LeastSquaresPolyAt0
 - akaroa_import, 21
- LEcuyer
 - prng, 102
 - prng.cc, 232
- lib_akaroa
 - akaroa_import.cc, 186
 - akaroa_import.h, 187
- lib_controller
 - controller.cc, 206
 - controller.h, 207
- lib_multipleStreams
 - interface.cc, 214
 - interface.h, 216
- lib_prng
 - prng.cc, 232
 - prng.h, 233
- lib_setting
 - setting.cc, 238
 - setting.h, 239
- lib_signals, 15
 - actNoChildProcesses, 16
 - ChildProcessPIDs, 16
 - continueExecution, 16
 - initializeUserDefinedSignals, 15
 - maxNoChildProcesses, 17
 - registerChildProcess, 16
 - sendSignalToAllChildProcesses, 16
 - signal_ignore, 16
 - signal_stop, 16
- lib_singleStream
 - interface.cc, 214
 - interface.h, 216
- lib_statistic
 - statistic.cc, 243
 - statistic.h, 245
- lib_system
 - system_command.cc, 246
 - system_command.h, 247
- load
 - setting, 143
- location
 - SequentialStoppingCriteria_QE::estimate, 140
- LogAveragePairsAndOffset
 - akaroa_import, 21
- logfile
 - logfile.cc, 217
 - logfile.h, 218
- logfile.cc, 217
 - logfile, 217
- logfile.h, 218
 - logfile, 218
- logInfo, 18
- logInfo
 - close, 18
 - open, 18
- LookUp_K_d
 - akaroa_import, 21
- m_accu
 - sequential_TPD, 135
- m_actBatch
 - batch_mean_QE, 35
 - sequential_TPD, 134
 - spectral_analysis_QE, 157
- m_actNoInBatch
 - batch_mean_QE, 35
 - sequential_TPD, 135
 - spectral_analysis_QE, 157
- m_actObservations
 - evolution, 70
- m_actSample
 - sequential_TPD, 134
- m_actTests
 - deterministic_SSC_QE, 57
- m_addedGroups
 - AndersonDarlingKSampleTestEqual-ECDFSize, 27
- m_addedValues
 - AndersonDarlingKSampleTestEqual-ECDFSize, 27
- m_alpha
 - AndersonDarlingKSampleTestEqual-ECDFSize, 27
 - batch_mean_QE, 35
 - batching, 44
 - evolution, 70
 - pooling_QE, 95
 - quantile_rank, 113
 - sequential_TPD, 134
 - spectral_analysis_QE, 157
- m_batch
 - batch_mean_QE, 35
 - spectral_analysis_QE, 157
- m_batchNo
 - batch_mean_QE, 34
 - spectral_analysis_QE, 156
- m_batchSize
 - batch_mean_QE, 35
 - controller, 52

- pooling_QE, 95
- quantile_estimation, 106
- sequential_TPD, 134
- spectral_analysis_QE, 157
- m_batchSizeIsSet
 - controller, 52
- m_buffer
 - interface_multipleRuns, 74
 - interface_singleRun, 77
- m_callDir
 - system_command, 180
- m_checkpoint
 - controller, 52
- m_class
 - error_in_FCM, 65
- m_close
 - setting, 145
- m_counter
 - confidenceInterval_SSC_QE, 49
 - deterministic_SSC_QE, 57
 - relativeErrorQuantile_SSC_QE, 118
 - relativeErrorRange_SSC_QE, 123
 - SequentialStoppingCriteria_QE, 139
- m_criticalValue
 - relativeErrorQuantile_SSC_QE, 118
 - relativeErrorRange_SSC_QE, 123
- m_cutoff
 - deterministic_TPD, 62
- m_data
 - AndersonDarlingKSampleTestEqual-ECDFSize, 27
- m_data_fast
 - sequential_TPD, 133
- m_data_initial
 - sequential_TPD, 133
- M_E2_1_response
 - statistic_collection, 164
- m_file
 - error_in_FCM, 65
- m_form
 - quantile_rank, 114
- m_groupSize
 - AndersonDarlingKSampleTestEqual-ECDFSize, 27
- M_H2_1_response
 - statistic_collection, 164
- m_home
 - system_command, 180
- m_homeDir
 - prng, 101
- m_host
 - system_command, 179
- m_hostName
 - prng, 101
- m_indexSubstream
 - prng, 101
- m_information
 - error_in_FCM, 65
- m_is
 - setting, 145
- m_isFirstGreaterThenSecond
 - KolmogorovSmirnov2SampleTest, 83
- m_isReady
 - batch_mean_QE, 34
 - batching, 44
 - pooling_QE, 95
 - spectral_analysis_QE, 156
- m_isSingleStream
 - prng, 101
- m_limit
 - sequential_TPD, 134
- m_loadFilename
 - setting, 144
- M_M_1_response
 - statistic_collection, 163
- m_maxBatch
 - batching, 44
- m_MaximumDifference
 - KolmogorovSmirnov2SampleTest, 82
- m_maxPipeBufferSize
 - interface_multipleRuns, 74
- m_maxTests
 - deterministic_SSC_QE, 57
- m_mean
 - batch_mean_QE, 35
 - spectral_analysis_QE, 157
- m_meanSample
 - sequential_TPD, 135
- m_measure_location
 - quantile_rank, 113
- m_measure_lower
 - quantile_rank, 113
- m_measure_upper
 - quantile_rank, 113
- m_method
 - error_in_FCM, 65
 - settingEntry, 149
- m_methodList
 - controller, 52
- m_minQuantiles
 - pooling_QE, 94
- m_name
 - KolmogorovSmirnov2SampleTest, 83
- m_nextCheckpoint
 - pooling_QE, 94
- m_nextSelection
 - sequential_TPD, 135
- m_noMethodID

- setting, 144
- m_noReplication
 - batch_mean_QE, 34
 - sequential_TPD, 134
 - spectral_analysis_QE, 156
- m_noReplications
 - interface_multipleRuns, 75
- m_noSubstreams
 - prng, 101
- m_numBatch
 - batching, 44
- m_open
 - setting, 145
- m_p
 - batching, 45
- m_parameter
 - settingEntry, 149
- m_performance
 - sequential_TPD, 133
- m_permanent
 - evolution, 70
- m_PID
 - system_command, 180
- m_pool
 - pooling_QE, 95
- m_posBatch
 - batching, 44
- m_posInSpace
 - pooling_QE, 94
- m_predefinedVariance
 - AndersonDarlingKSampleTestEqual-ECDFSize, 27
- m_predefinedVarianceIsValid
 - AndersonDarlingKSampleTestEqual-ECDFSize, 27
- m_probability_location
 - quantile_rank, 113
- m_processedBatches
 - pooling_QE, 94
- m_processedIndexes
 - batch_mean_QE, 35
 - batching, 45
 - confidenceInterval_SSC_QE, 49
 - deterministic_SSC_QE, 57
 - deterministic_TPD, 62
 - evolution, 71
 - outputAnalyser, 88
 - pooling_QE, 95
 - quantile_estimation, 107
 - relativeErrorQuantile_SSC_QE, 118
 - relativeErrorRange_SSC_QE, 123
 - sequential_TPD, 135
 - SequentialStoppingCriteria_QE, 139
 - spectral_analysis_QE, 157
 - truncation_point_detection, 184
- m_ptr
 - interface_multipleRuns, 74
 - interface_singleRun, 77
- m_rank_location
 - quantile_rank, 113
- m_rank_lower
 - quantile_rank, 113
- m_rank_upper
 - quantile_rank, 113
- m_ratio
 - sequential_TPD, 133
- m_ratio_max
 - sequential_TPD, 134
- m_ratio_min
 - sequential_TPD, 133
- m_replications
 - evolution, 70
- m_resultAlreadyPrinted
 - deterministic_TPD, 62
- m_resultfile
 - evolution, 70
- m_sample1
 - KolmogorovSmirnov2SampleTest, 82
- m_sample2
 - KolmogorovSmirnov2SampleTest, 82
- m_sampleSize
 - quantile_rank, 113
- m_seed
 - prng, 102
- m_seedFileName
 - prng, 101
- m_simulationHorizon
 - controller, 52
- m_sizeBatch
 - batching, 44
- m_sortedData
 - batching, 44
- m_SSC
 - batch_mean_QE, 35
 - pooling_QE, 95
 - quantile_estimation, 107
 - spectral_analysis_QE, 157
- m_StandardizedMaximumDifference
 - KolmogorovSmirnov2SampleTest, 82
- m_start
 - evolution, 70
- m_state
 - sequential_TPD, 133
- m_stop
 - evolution, 70
- m_substreamSeed
 - prng, 102
- m_truncationPoint

- sequential_TPD, 133
- m_typeOfBatchStatistic
 - batching, 44
- m_typeOfIndependenceTest
 - batching, 44
- m_UID
 - system_command, 180
- m_user
 - system_command, 179
- m_value
 - settingEntry, 149
- m_varianceHomogeneityTest
 - sequential_TPD, 134
- m_wantedMethods
 - method_factory, 85
- m_wantedQuantiles
 - evolution, 70
- main
 - main.cc, 220
- main.cc, 220
 - main, 220
- main.h, 222
- maxNoChildProcesses
 - lib_signals, 17
- Mean
 - batching, 39
- mean
 - statistic_collection, 173
- measure.cc, 223
- measure.h, 224
 - CONTINUOUS, 225
 - CtoD, 226
 - CtoI, 226
 - CtoS, 226
 - DISCRETE, 225
 - DtoC, 226
 - DtoI, 226
 - DtoS, 227
 - INDEX, 225
 - ItoC, 227
 - ItoD, 227
 - ItoS, 227
 - StoC, 227
 - StoD, 227
 - StoI, 227
- method_factory, 84
 - ~method_factory, 84
 - construct, 85
 - m_wantedMethods, 85
 - method_factory, 84
 - moreAvailable, 85
- method_factory.cc, 229
- method_factory.h, 230
- mkResultDir
 - system_command, 179
- moreAvailable
 - method_factory, 85
- NON
 - basic.h, 197
- NORMAL
 - statistic.h, 245
- normal
 - statistic_collection, 162
- open
 - logInfo, 18
- operator!=
 - settingEntry, 148
- operator<
 - quantile_rank, 111
 - settingEntry, 149
- operator<=
 - settingEntry, 149
- operator=
 - quantile_rank, 111
 - settingEntry, 149
- operator==
 - quantile_rank, 111
 - settingEntry, 148
- operator>
 - settingEntry, 149
- operator>=
 - settingEntry, 149
- OrthogonalPolynomialTables
 - akaroa_import, 21
- OrthogonalPolynomialValues
 - akaroa_import, 22
- outputAnalyser, 86
 - outputAnalyser, 87
- outputAnalyser
 - ~outputAnalyser, 87
 - getType, 87
 - isReady, 87
 - m_processedIndexes, 88
 - outputAnalyser, 87
 - printResult, 88
 - printSetting, 87
 - printStatus, 88
 - process, 87
- parseMethod
 - setting, 144
- parseParameter
 - setting, 144
- PearsonPermutation
 - statistic_collection, 161
- pearsonPermutation

- statistic_collection, 171
- pearsonPermutation_statistic
 - statistic_collection, 171
- pearsonPermutation_test
 - statistic_collection, 171
- pearsonsCorrelationCoefficient
 - statistic_collection, 172
- PearsonStrelen
 - statistic_collection, 161
- pearsonStrelen
 - statistic_collection, 170
- pearsonStrelen_statistic
 - statistic_collection, 170
- pearsonStrelen_test
 - statistic_collection, 170
- performance
 - sequential_TPD, 127
- permutationAll
 - statistic_collection, 174
- pooling_QE, 89
 - ~pooling_QE, 91
 - checkQuantiles, 93
 - getType, 93
 - isReady, 92
 - m_alpha, 95
 - m_batchSize, 95
 - m_isReady, 95
 - m_minQuantiles, 94
 - m_nextCheckpoint, 94
 - m_pool, 95
 - m_posInSpace, 94
 - m_processedBatches, 94
 - m_processedIndexes, 95
 - m_SSC, 95
 - pooling_QE, 91
 - printResult, 92
 - printSetting, 92
 - printStatus, 92
 - process, 92
 - set_SSC, 94
 - setBatchSize, 94
 - settings, 93
- precise
 - sequential_TPD, 127
- print
 - confidenceInterval_SSC_QE, 48
 - deterministic_SSC_QE, 56
 - error_in_FCM, 65
 - relativeErrorQuantile_SSC_QE, 117
 - relativeErrorRange_SSC_QE, 122
 - resultInfo, 124
 - SequentialStoppingCriteria_QE, 138
- printDistribution
 - sequential_TPD, 132
- printResult
 - batch_mean_QE, 32
 - batching, 41
 - deterministic_TPD, 62
 - evolution, 69
 - outputAnalyser, 88
 - pooling_QE, 92
 - quantile_estimation, 106
 - sequential_TPD, 129
 - spectral_analysis_QE, 154
 - truncation_point_detection, 183
- printSetting
 - batch_mean_QE, 32
 - batching, 41
 - deterministic_TPD, 61
 - evolution, 69
 - outputAnalyser, 87
 - pooling_QE, 92
 - quantile_estimation, 106
 - sequential_TPD, 129
 - spectral_analysis_QE, 154
 - truncation_point_detection, 183
- printState
 - prng, 101
- printStatus
 - batch_mean_QE, 32
 - batching, 41
 - controller, 51
 - deterministic_TPD, 62
 - evolution, 69
 - outputAnalyser, 88
 - pooling_QE, 92
 - quantile_estimation, 106
 - sequential_TPD, 129
 - spectral_analysis_QE, 154
 - truncation_point_detection, 183
- printToLogfile
 - prng, 100
 - setting, 143
 - system_command, 178
- prng, 97
 - ~prng, 98
 - draw_bool, 99
 - draw_erlang, 100
 - draw_exponential, 100
 - draw_integer, 99
 - draw_normal, 99
 - draw_pareto, 100
 - draw_probability, 99
 - draw_std_normal, 99
 - LEcuyer, 102
 - m_homeDir, 101
 - m_hostName, 101
 - m_indexSubstream, 101

- m_isSingleStream, 101
- m_noSubstreams, 101
- m_seed, 102
- m_seedFileName, 101
- m_substreamSeed, 102
- printState, 101
- printToLogFile, 100
- prng, 98
- readSeedFile, 100
- updateSeedFile, 101
- prng.cc, 231
 - LEcuyer, 232
 - lib_prng, 232
- prng.h, 233
 - lib_prng, 233
- probability
 - SequentialStoppingCriteria_QE::estimate, 140
- process
 - batch_mean_QE, 31
 - batching, 40
 - controller, 51
 - deterministic_TPD, 61
 - evolution, 69
 - outputAnalyser, 87
 - pooling_QE, 92
 - quantile_estimation, 106
 - sequential_TPD, 128
 - spectral_analysis_QE, 153
 - truncation_point_detection, 183
- quantile_estimation, 103
 - ~quantile_estimation, 105
 - getType, 105
 - isReady, 105
 - m_batchSize, 106
 - m_processedIndexes, 107
 - m_SSC, 107
 - printResult, 106
 - printSetting, 106
 - printStatus, 106
 - process, 106
 - quantile_estimation, 105
 - set_SSC, 105
 - setBatchSize, 105
- quantile_estimation.cc, 234
- quantile_estimation.h, 235
- quantile_rank, 108
 - ~quantile_rank, 109
 - calculateLowerRank, 112
 - calculateUnbiasedQuantile, 111
 - calculateUpperRank, 112
 - cdf, 111
 - getLowerMeasure, 110
 - getLowerRank, 110
 - getMeasure, 110
 - getRank, 110
 - getUnbiasedQuantile, 110
 - getUpperMeasure, 110
 - getUpperRank, 110
 - m_alpha, 113
 - m_form, 114
 - m_measure_location, 113
 - m_measure_lower, 113
 - m_measure_upper, 113
 - m_probability_location, 113
 - m_rank_location, 113
 - m_rank_lower, 113
 - m_rank_upper, 113
 - m_sampleSize, 113
 - operator<, 111
 - operator=, 111
 - operator==, 111
 - quantile_rank, 109
 - quantileCDF, 112
 - setMeasure, 111
- quantileCDF
 - quantile_rank, 112
- ranks
 - statistic_collection, 173
- readSeedFile
 - prng, 100
- receive
 - interface_multipleRuns, 74
 - interface_singleRun, 77
- receive_parse
 - interface_singleRun, 77
- registerChildProcess
 - lib_signals, 16
- relativeErrorQuantile_SSC_QE, 115
 - relativeErrorQuantile_SSC_QE, 116
- relativeErrorQuantile_SSC_QE
 - ~relativeErrorQuantile_SSC_QE, 116
 - estimateMap, 118
 - getName, 116
 - insert, 117
 - isFulfilled, 116
 - m_counter, 118
 - m_criticalValue, 118
 - m_processedIndexes, 118
 - print, 117
 - relativeErrorQuantile_SSC_QE, 116
 - reset, 117
 - setProcessedIndexes, 118
 - settings, 117
- relativeErrorRange_SSC_QE, 120
 - relativeErrorRange_SSC_QE, 121

- relativeErrorRange_SSC_QE
 - ~relativeErrorRange_SSC_QE, 121
 - estimateMap, 123
 - getName, 121
 - insert, 122
 - isFulfilled, 121
 - m_counter, 123
 - m_criticalValue, 123
 - m_processedIndexes, 123
 - print, 122
 - relativeErrorRange_SSC_QE, 121
 - reset, 122
 - setProcessedIndexes, 123
 - settings, 122
- reset
 - confidenceInterval_SSC_QE, 48
 - deterministic_SSC_QE, 56
 - relativeErrorQuantile_SSC_QE, 117
 - relativeErrorRange_SSC_QE, 122
 - SequentialStoppingCriteria_QE, 137
- resultfile
 - resultfile.cc, 236
 - resultfile.h, 237
- resultfile.cc, 236
 - resultfile, 236
- resultfile.h, 237
 - resultfile, 237
- resultInfo, 124
- resultInfo
 - print, 124
- RunsAboveBelow
 - statistic_collection, 161
- runsAboveBelow
 - statistic_collection, 168
- runsAboveBelow_statistic
 - statistic_collection, 168
- runsAboveBelow_test
 - statistic_collection, 169
- RunsUpDown
 - statistic_collection, 161
- runsUpDown
 - statistic_collection, 167
- runsUpDown_statistic
 - statistic_collection, 168
- runsUpDown_test
 - statistic_collection, 168
- s_alpha
 - basic.cc, 189
 - basic.h, 197
- s_auto
 - basic.cc, 189
 - basic.h, 197
- s_batch_max
 - basic.cc, 189
 - basic.h, 198
- s_batch_mean_QE
 - basic.cc, 189
 - basic.h, 198
- s_batches
 - basic.cc, 189
 - basic.h, 198
- s_confidenceInterval_SSC_QE
 - basic.cc, 189
 - basic.h, 198
- s_controller
 - basic.cc, 189
 - basic.h, 198
- s_critical_value
 - basic.cc, 190
 - basic.h, 198
- s_cutoff
 - basic.cc, 190
 - basic.h, 198
- s_deterministic_SSC_QE
 - basic.cc, 190
 - basic.h, 198
- s_deterministic_TPD
 - basic.cc, 190
 - basic.h, 199
- s_evolution
 - basic.cc, 190
 - basic.h, 199
- s_exact
 - basic.cc, 190
 - basic.h, 199
- s_execute
 - basic.cc, 190
 - basic.h, 199
- s_fast
 - basic.cc, 190
 - basic.h, 199
- s_independence
 - basic.cc, 191
 - basic.h, 199
- s_limit
 - basic.cc, 191
 - basic.h, 199
- s_mean
 - basic.cc, 191
 - basic.h, 199
- s_no
 - basic.cc, 191
 - basic.h, 200
- s_pearsonPermutation
 - basic.cc, 191
 - basic.h, 200
- s_pearsonStrelen

- basic.cc, 191
- basic.h, 200
- s_performance
 - basic.cc, 191
 - basic.h, 200
- s_permanent
 - basic.cc, 191
 - basic.h, 200
- s_pooling_QE
 - basic.cc, 192
 - basic.h, 200
- s_precise
 - basic.cc, 192
 - basic.h, 200
- s_quantiles_min
 - basic.cc, 192
 - basic.h, 200
- s_ratio
 - basic.cc, 192
 - basic.h, 201
- s_ratio_max
 - basic.cc, 192
 - basic.h, 201
- s_ratio_min
 - basic.cc, 192
 - basic.h, 201
- s_relativeErrorQuantile_SSC_QE
 - basic.cc, 192
 - basic.h, 201
- s_relativeErrorRange_SSC_QE
 - basic.cc, 192
 - basic.h, 201
- s_replications
 - basic.cc, 193
 - basic.h, 201
- s_runsAboveBelow
 - basic.cc, 193
 - basic.h, 201
- s_runsUpDown
 - basic.cc, 193
 - basic.h, 201
- s_sequential_batching
 - basic.cc, 193
 - basic.h, 202
- s_sequential_TPD
 - basic.cc, 193
 - basic.h, 202
- s_sort
 - basic.cc, 193
 - basic.h, 202
- s_spacing
 - basic.cc, 193
 - basic.h, 202
- s_spectral_analysis_QE
 - basic.cc, 193
 - basic.h, 202
- s_start
 - basic.cc, 194
 - basic.h, 202
- s_statistic
 - basic.cc, 194
 - basic.h, 202
- s_stop
 - basic.cc, 194
 - basic.h, 202
- s_vonNeumann
 - basic.cc, 194
 - basic.h, 203
- s_yes
 - basic.cc, 194
 - basic.h, 203
- SchrubenStatistic
 - akaroa_import, 20
- send
 - interface_multipleRuns, 74
 - interface_singleRun, 77
- sendSignalToAllChildProcesses
 - lib_signals, 16
- sequential_TPD, 125
 - ~sequential_TPD, 128
 - begin, 127
 - collect, 127
 - exact, 127
 - fast, 127
 - finished, 127
 - getTruncationIndex, 128
 - getType, 132
 - homogeneityTest, 132
 - initialize, 127
 - isReady, 128
 - m_accu, 135
 - m_actBatch, 134
 - m_actNoInBatch, 135
 - m_actSample, 134
 - m_alpha, 134
 - m_batchSize, 134
 - m_data_fast, 133
 - m_data_initial, 133
 - m_limit, 134
 - m_meanSample, 135
 - m_nextSelection, 135
 - m_noReplication, 134
 - m_performance, 133
 - m_processedIndexes, 135
 - m_ratio, 133
 - m_ratio_max, 134
 - m_ratio_min, 133
 - m_state, 133

- m_truncationPoint, 133
- m_varianceHomogeneityTest, 134
- performance, 127
- precise, 127
- printDistribution, 132
- printResult, 129
- printSetting, 129
- printStatus, 129
- process, 128
- sequential_TPD, 128
- settings, 131
- state, 127
- sub_begin, 130
- sub_collect, 130
- sub_compare, 131
- sub_finished, 131
- sub_initialize, 130
- SequentialStoppingCriteria_QE, 136
 - SequentialStoppingCriteria_QE, 137
- SequentialStoppingCriteria_QE
 - ~SequentialStoppingCriteria_QE, 137
- estimateMap, 139
- getName, 138
- insert, 137
- isFulfilled, 138
- m_counter, 139
- m_processedIndexes, 139
- print, 138
- reset, 137
- SequentialStoppingCriteria_QE, 137
- setProcessedIndexes, 138
- settings, 138
- SequentialStoppingCriteria_QE::estimate, 140
- SequentialStoppingCriteria_QE::estimate
 - absoluteErrorNeg, 140
 - absoluteErrorPos, 140
 - location, 140
 - probability, 140
- set_SSC
 - batch_mean_QE, 34
 - pooling_QE, 94
 - quantile_estimation, 105
 - spectral_analysis_QE, 156
- setAlpha
 - AndersonDarlingKSampleTestEqual-ECDFSize, 25
- setBatchSize
 - batch_mean_QE, 34
 - pooling_QE, 94
 - quantile_estimation, 105
 - spectral_analysis_QE, 156
- setMeasure
 - quantile_rank, 111
- setNoReplications
 - interface_multipleRuns, 74
- setPredefinedVariance
 - AndersonDarlingKSampleTestEqual-ECDFSize, 25
- setProcessedIndexes
 - confidenceInterval_SSC_QE, 48
 - deterministic_SSC_QE, 57
 - relativeErrorQuantile_SSC_QE, 118
 - relativeErrorRange_SSC_QE, 123
 - SequentialStoppingCriteria_QE, 138
- setting, 142
 - ~setting, 143
 - actualSettings, 144
 - clean, 144
 - get, 143
 - getNoMethodID, 143
 - load, 143
 - m_close, 145
 - m_is, 145
 - m_loadFilename, 144
 - m_noMethodID, 144
 - m_open, 145
 - parseMethod, 144
 - parseParameter, 144
 - printToLogfile, 143
 - setting, 143
- setting.cc, 238
 - lib_setting, 238
- setting.h, 239
 - lib_setting, 239
- settingEntry, 146
 - settingEntry, 147
- settingEntry
 - ~settingEntry, 147
 - getMethod, 148
 - getParameter, 148
 - getValue, 148
 - getValueBoolean, 147
 - getValueContinuous, 148
 - getValueDiscrete, 147
 - getValueIndex, 147
 - m_method, 149
 - m_parameter, 149
 - m_value, 149
 - operator!=, 148
 - operator<, 149
 - operator<=, 149
 - operator=, 149
 - operator==, 148
 - operator>, 149
 - operator>=, 149
 - settingEntry, 147
- settings
 - batch_mean_QE, 33

- batching, 43
- confidenceInterval_SSC_QE, 47
- deterministic_SSC_QE, 56
- pooling_QE, 93
- relativeErrorQuantile_SSC_QE, 117
- relativeErrorRange_SSC_QE, 122
- sequential_TPD, 131
- SequentialStoppingCriteria_QE, 138
- spectral_analysis_QE, 155
- siegelsRunTest
 - statistic_collection, 174
- siegelsRunTest_large
 - statistic_collection, 174
- siegelsRunTest_small
 - statistic_collection, 174
- signal_ignore
 - lib_signals, 16
- signal_interface.cc, 241
- signal_interface.h, 242
- signal_stop
 - lib_signals, 16
- sinh
 - statistic_collection, 165
- SLOPE_PROTECTION_CONDITIONAL
 - akaroa_import, 20
- SLOPE_PROTECTION_OFF
 - akaroa_import, 20
- SLOPE_PROTECTION_UNCONDITIONAL
 - akaroa_import, 20
- sortVector
 - AndersonDarlingKSampleTestEqual-ECDFSize, 26
 - KolmogorovSmirnov2SampleTest, 82
- source/ Directory Reference, 13
- Spacing
 - batching, 39
- spearmanCorrelationCoefficient
 - statistic_collection, 173
- spectral_analysis_QE, 151
 - ~spectral_analysis_QE, 153
 - checkQuantiles, 154
 - collapse, 155
 - getType, 156
 - isReady, 153
 - m_actBatch, 157
 - m_actNoInBatch, 157
 - m_alpha, 157
 - m_batch, 157
 - m_batchNo, 156
 - m_batchSize, 157
 - m_isReady, 156
 - m_mean, 157
 - m_noReplication, 156
 - m_processedIndexes, 157
 - m_SSC, 157
 - printResult, 154
 - printSetting, 154
 - printStatus, 154
 - process, 153
 - set_SSC, 156
 - setBatchSize, 156
 - settings, 155
 - spectral_analysis_QE, 153
- SpectralVarianceAnalysisOfMean
 - akaroa_import, 20
- SpectralVarianceAnalysisOfProcess
 - akaroa_import, 20
- sq
 - statistic_collection, 166
- sqr
 - akaroa_import, 22
- state
 - sequential_TPD, 127
- statistic.cc, 243
 - lib_statistic, 243
- statistic.h, 244
 - distribution, 245
 - EXPONENTIAL, 245
 - lib_statistic, 245
 - NORMAL, 245
 - UNIFORM, 245
 - UNSPECIFIED, 245
- statistic_collection, 159
 - acosh, 166
 - acoth, 166
 - asinh, 166
 - atanh, 166
 - binomial, 162
 - binomialCoefficient, 172
 - chooseDistribution, 161
 - chooseQuantiles, 166
 - chooseQuantiles_old, 166
 - cosh, 165
 - coth, 166
 - exponential, 163
 - f_distribution, 162
 - finiteSumCorrelationCoefficients_MM1, 175
 - generateRandomPermutation, 173
 - independenceTest, 167
 - infiniteSumCorrelationCoefficients_MM1, 175
 - inv_binomial, 162
 - inv_exponential, 163
 - inv_f_distribution, 162
 - inv_M_E2_1_response, 164
 - inv_M_H2_1_response, 165
 - inv_M_M_1_response, 164

- inv_normal, 162
- inv_t_distribution, 163
- inv_uniform, 163
- M_E2_1_response, 164
- M_H2_1_response, 164
- M_M_1_response, 163
- mean, 173
- normal, 162
- PearsonPermutation, 161
- pearsonPermutation, 171
- pearsonPermutation_statistic, 171
- pearsonPermutation_test, 171
- pearsonsCorrelationCoefficient, 172
- PearsonStrelen, 161
- pearsonStrelen, 170
- pearsonStrelen_statistic, 170
- pearsonStrelen_test, 170
- permutationAll, 174
- ranks, 173
- RunsAboveBelow, 161
- runsAboveBelow, 168
- runsAboveBelow_statistic, 168
- runsAboveBelow_test, 169
- RunsUpDown, 161
- runsUpDown, 167
- runsUpDown_statistic, 168
- runsUpDown_test, 168
- siegelsRunTest, 174
- siegelsRunTest_large, 174
- siegelsRunTest_small, 174
- sinh, 165
- spearmanCorrelationCoefficient, 173
- sq, 166
- t_distribution, 162
- tanh, 165
- TypeOfIndependenceTest, 161
- uniform, 163
- VonNeuman, 161
- vonNeumann, 169
- vonNeumann_statistic, 169
- vonNeumann_test, 169
- vonNeumannsCorrelationCoefficient, 173
- StoC
 - measure.h, 227
- StoD
 - measure.h, 227
- StoI
 - measure.h, 227
- sub_begin
 - sequential_TPD, 130
- sub_collect
 - sequential_TPD, 130
- sub_compare
 - sequential_TPD, 131
- sub_finished
 - sequential_TPD, 131
- sub_initialize
 - sequential_TPD, 130
- system_command, 176
 - ~system_command, 177
 - execute, 178
 - getCallDir, 178
 - getDate, 177
 - getHome, 178
 - getHost, 178
 - getPID, 178
 - getPWD, 178
 - getTime, 177
 - getUID, 178
 - getUser, 177
 - m_callDir, 180
 - m_home, 180
 - m_host, 179
 - m_PID, 180
 - m_UID, 180
 - m_user, 179
 - mkResultDir, 179
 - printToLogfile, 178
 - system_command, 177
- system_command.cc, 246
 - lib_system, 246
- system_command.h, 247
 - lib_system, 247
- t_distribution
 - akaroa_import, 22
 - statistic_collection, 162
- tanh
 - statistic_collection, 165
- testBatchStatistic
 - batching, 42
- time_evolution.cc, 248
- time_evolution.h, 249
- truncation_point_detection, 181
 - ~truncation_point_detection, 182
 - getType, 183
 - isReady, 183
 - m_processedIndexes, 184
 - printResult, 183
 - printSetting, 183
 - printStatus, 183
 - process, 183
 - truncation_point_detection, 182
- truncation_point_detection.cc, 250
- truncation_point_detection.h, 251
- TypeOfBatching
 - batching, 39
- TypeOfIndependenceTest

- statistic_collection, 161
- TypeOfMethod
 - basic.h, 197
- UNIFORM
 - statistic.h, 245
- uniform
 - statistic_collection, 163
- UNSPECIFIED
 - statistic.h, 245
- updateBatchStatistic
 - batching, 41
- updateSeedFile
 - prng, 101
- VonNeuman
 - statistic_collection, 161
- vonNeumann
 - statistic_collection, 169
- vonNeumann_statistic
 - statistic_collection, 169
- vonNeumann_test
 - statistic_collection, 169
- vonNeumannsCorrelationCoefficient
 - statistic_collection, 173
- Z
 - akaroa_import, 22