

# Evaporative Cooling

1.0

Generated by Doxygen 1.7.4

Fri Jan 6 2012 19:38:13



# Contents

<b>1</b>	<b>Namespace Index</b>	<b>1</b>
1.1	Namespace List . . . . .	1
<b>2</b>	<b>Class Index</b>	<b>3</b>
2.1	Class Hierarchy . . . . .	3
<b>3</b>	<b>Class Index</b>	<b>5</b>
3.1	Class List . . . . .	5
<b>4</b>	<b>File Index</b>	<b>7</b>
4.1	File List . . . . .	7
<b>5</b>	<b>Namespace Documentation</b>	<b>9</b>
5.1	insilico Namespace Reference . . . . .	9
5.1.1	Function Documentation . . . . .	10
5.1.1.1	best_n . . . . .	10
5.1.1.2	get_bits . . . . .	11
5.1.1.3	join . . . . .	11
5.1.1.4	join . . . . .	11
5.1.1.5	join . . . . .	11
5.1.1.6	split . . . . .	11
5.1.1.7	split . . . . .	11
5.1.1.8	split . . . . .	12
5.1.1.9	split . . . . .	12
5.1.1.10	split . . . . .	12
5.1.1.11	split . . . . .	12
5.1.1.12	split . . . . .	12

5.1.1.13	<a href="#">split</a>	12
5.1.1.14	<a href="#">split</a>	12
5.1.1.15	<a href="#">split</a>	12
5.1.1.16	<a href="#">split_if</a>	12
5.1.1.17	<a href="#">split_if</a>	13
5.1.1.18	<a href="#">split_if</a>	13
5.1.1.19	<a href="#">to_lower</a>	13
5.1.1.20	<a href="#">to_lower</a>	13
5.1.1.21	<a href="#">to_lower</a>	13
5.1.1.22	<a href="#">to_upper</a>	13
5.1.1.23	<a href="#">to_upper</a>	13
5.1.1.24	<a href="#">to_upper</a>	13
5.1.1.25	<a href="#">trim</a>	13
5.1.1.26	<a href="#">trim</a>	14
5.1.1.27	<a href="#">trim</a>	14
5.1.1.28	<a href="#">trim_left</a>	14
5.1.1.29	<a href="#">trim_left</a>	14
5.1.1.30	<a href="#">trim_left</a>	14
5.1.1.31	<a href="#">trim_right</a>	14
5.1.1.32	<a href="#">trim_right</a>	14
5.1.1.33	<a href="#">trim_right</a>	14
5.1.1.34	<a href="#">zeroPadNumber</a>	14
<b>6</b>	<b>Class Documentation</b>	<b>15</b>
6.1	<a href="#">ArffDataset Class Reference</a>	15
6.1.1	<a href="#">Detailed Description</a>	18
6.1.2	<a href="#">Constructor &amp; Destructor Documentation</a>	19
6.1.2.1	<a href="#">ArffDataset</a>	19
6.1.2.2	<a href="#">~ArffDataset</a>	19
6.1.3	<a href="#">Member Function Documentation</a>	19
6.1.3.1	<a href="#">GetAttributeLevel</a>	19
6.1.3.2	<a href="#">GetDiscreteClassLevel</a>	19
6.1.3.3	<a href="#">GetNumericClassLevel</a>	20
6.1.3.4	<a href="#">GetTypeOf</a>	20

6.1.3.5	LoadSnps	20
6.1.3.6	PrintNominalsMapping	20
6.1.4	Member Data Documentation	21
6.1.4.1	attributeTypes	21
6.1.4.2	missingAttributeValuesToCheck	21
6.1.4.3	missingClassValuesToCheck	21
6.1.4.4	nominalValues	21
6.1.4.5	relationName	21
6.2	ChiSquared Class Reference	21
6.2.1	Detailed Description	24
6.2.2	Constructor & Destructor Documentation	24
6.2.2.1	ChiSquared	24
6.2.2.2	~ChiSquared	24
6.2.3	Member Function Documentation	24
6.2.3.1	ClearTables	24
6.2.3.2	ComputeScore	24
6.2.3.3	ComputeScores	25
6.2.3.4	GetFrequencyCounts	25
6.2.3.5	PrepareForAttribute	25
6.2.3.6	PrintScores	25
6.2.3.7	PrintTables	26
6.2.3.8	WriteScores	26
6.2.4	Member Data Documentation	26
6.2.4.1	chiSquaredValues	26
6.2.4.2	dataset	26
6.2.4.3	expectedContingencyTable	26
6.2.4.4	numClasses	26
6.2.4.5	numLevels	26
6.2.4.6	observedFreqTable	27
6.2.4.7	scores	27
6.3	CleanSnpDataDataset Class Reference	27
6.3.1	Detailed Description	30
6.3.2	Constructor & Destructor Documentation	30
6.3.2.1	CleanSnpDataDataset	30

6.3.2.2	<a href="#">~CleanSnpsDataset</a>	30
6.3.3	<a href="#">Member Function Documentation</a>	30
6.3.3.1	<a href="#">LoadSnps</a>	30
6.4	<a href="#">Dataset Class Reference</a>	31
6.4.1	<a href="#">Detailed Description</a>	40
6.4.2	<a href="#">Constructor &amp; Destructor Documentation</a>	40
6.4.2.1	<a href="#">Dataset</a>	40
6.4.2.2	<a href="#">~Dataset</a>	41
6.4.3	<a href="#">Member Function Documentation</a>	41
6.4.3.1	<a href="#">AttributeInteractionInformation</a>	41
6.4.3.2	<a href="#">CalculateGainMatrix</a>	41
6.4.3.3	<a href="#">CalculateInteractionInformation</a>	41
6.4.3.4	<a href="#">CheckHardyWeinbergEquilibrium</a>	42
6.4.3.5	<a href="#">ExtractAttributes</a>	42
6.4.3.6	<a href="#">GetAlternatePhenotypesFilename</a>	42
6.4.3.7	<a href="#">GetAttribute</a>	43
6.4.3.8	<a href="#">GetAttributeIndexFromName</a>	43
6.4.3.9	<a href="#">GetAttributeLevel</a>	43
6.4.3.10	<a href="#">GetAttributeMAF</a>	44
6.4.3.11	<a href="#">GetAttributeMutationType</a>	44
6.4.3.12	<a href="#">GetAttributeNames</a>	44
6.4.3.13	<a href="#">GetAttributeRowCol</a>	44
6.4.3.14	<a href="#">GetAttributeValues</a>	45
6.4.3.15	<a href="#">GetAttributeValues</a>	45
6.4.3.16	<a href="#">GetAttributeValueType</a>	46
6.4.3.17	<a href="#">GetClassColumn</a>	46
6.4.3.18	<a href="#">GetClassIndexes</a>	46
6.4.3.19	<a href="#">GetClassProbability</a>	46
6.4.3.20	<a href="#">GetClassValues</a>	47
6.4.3.21	<a href="#">GetClassValueType</a>	47
6.4.3.22	<a href="#">GetDiscreteClassLevel</a>	47
6.4.3.23	<a href="#">GetInstance</a>	48
6.4.3.24	<a href="#">GetInstanceIds</a>	48
6.4.3.25	<a href="#">GetInstanceIndexForID</a>	48

6.4.3.26	<a href="#">GetIntForGenotype</a>	48
6.4.3.27	<a href="#">GetMeanForNumeric</a>	49
6.4.3.28	<a href="#">GetMinMaxForContinuousPhenotype</a>	49
6.4.3.29	<a href="#">GetMinMaxForNumeric</a>	49
6.4.3.30	<a href="#">GetNumeric</a>	49
6.4.3.31	<a href="#">GetNumericClassLevel</a>	50
6.4.3.32	<a href="#">GetNumericIndexFromName</a>	50
6.4.3.33	<a href="#">GetNumericRowCol</a>	50
6.4.3.34	<a href="#">GetNumericsFilename</a>	51
6.4.3.35	<a href="#">GetNumericsNames</a>	51
6.4.3.36	<a href="#">GetNumericValues</a>	51
6.4.3.37	<a href="#">GetNumericValues</a>	51
6.4.3.38	<a href="#">GetProbabilityValueGivenClass</a>	52
6.4.3.39	<a href="#">GetRandomInstance</a>	52
6.4.3.40	<a href="#">GetSnpsFilename</a>	52
6.4.3.41	<a href="#">GetVariableNames</a>	52
6.4.3.42	<a href="#">HasAlternatePhenotypes</a>	53
6.4.3.43	<a href="#">HasContinuousPhenotypes</a>	53
6.4.3.44	<a href="#">HasGenotypes</a>	53
6.4.3.45	<a href="#">HasNumerics</a>	53
6.4.3.46	<a href="#">IsLoadableInstanceID</a>	53
6.4.3.47	<a href="#">LoadAlternatePhenotypes</a>	53
6.4.3.48	<a href="#">LoadDataset</a>	54
6.4.3.49	<a href="#">LoadNumerics</a>	54
6.4.3.50	<a href="#">LoadSnps</a>	54
6.4.3.51	<a href="#">MaskGetAllVariableNames</a>	55
6.4.3.52	<a href="#">MaskGetAttributeIndices</a>	55
6.4.3.53	<a href="#">MaskGetAttributeMask</a>	55
6.4.3.54	<a href="#">MaskGetInstanceIds</a>	56
6.4.3.55	<a href="#">MaskGetInstanceIndices</a>	56
6.4.3.56	<a href="#">MaskGetInstanceMask</a>	56
6.4.3.57	<a href="#">MaskIncludeAllAttributes</a>	56
6.4.3.58	<a href="#">MaskIncludeAllInstances</a>	56
6.4.3.59	<a href="#">MaskPopAll</a>	57

6.4.3.60	MaskPushAll	57
6.4.3.61	MaskRemoveAttribute	57
6.4.3.62	MaskRemoveInstance	57
6.4.3.63	MaskSearchAttribute	58
6.4.3.64	MaskSearchInstance	58
6.4.3.65	MaskWriteNewDataset	58
6.4.3.66	NumAttributes	59
6.4.3.67	NumClasses	59
6.4.3.68	NumInstances	59
6.4.3.69	NumLevels	59
6.4.3.70	NumNumerics	59
6.4.3.71	NumVariables	59
6.4.3.72	Print	60
6.4.3.73	PrintAttributeLevelsSeen	60
6.4.3.74	PrintClassIndexInfo	60
6.4.3.75	PrintLevelCounts	60
6.4.3.76	PrintMaskStats	60
6.4.3.77	PrintMissingValuesStats	60
6.4.3.78	PrintNumericsStats	60
6.4.3.79	PrintRecodeMap	61
6.4.3.80	PrintStats	61
6.4.3.81	PrintStatsSimple	61
6.4.3.82	RunSnpDiagnosticTests	61
6.4.3.83	SNPHWE	61
6.4.3.84	SwapAttributes	62
6.4.3.85	UpdateAllLevelCounts	62
6.4.3.86	UpdateLevelCounts	62
6.4.3.87	WriteLevelCounts	63
6.4.3.88	WriteNewDataset	63
6.4.4	Member Data Documentation	63
6.4.4.1	alternatePhenotypesFilename	63
6.4.4.2	attributeAlleleCounts	63
6.4.4.3	attributeAlleles	64
6.4.4.4	attributeLevelsSeen	64



6.4.4.5	<a href="#">attributeMinorAllele</a>	64
6.4.4.6	<a href="#">attributeMutationMap</a>	64
6.4.4.7	<a href="#">attributeMutationTypes</a>	64
6.4.4.8	<a href="#">attributeNames</a>	64
6.4.4.9	<a href="#">attributesMask</a>	64
6.4.4.10	<a href="#">attributesMaskPushed</a>	65
6.4.4.11	<a href="#">classColumn</a>	65
6.4.4.12	<a href="#">classIndexes</a>	65
6.4.4.13	<a href="#">continuousPhenotypeMinMax</a>	65
6.4.4.14	<a href="#">genotypeCounts</a>	65
6.4.4.15	<a href="#">hasAlternatePhenotypes</a>	65
6.4.4.16	<a href="#">hasContinuousPhenotypes</a>	65
6.4.4.17	<a href="#">hasGenotypes</a>	66
6.4.4.18	<a href="#">hasNumerics</a>	66
6.4.4.19	<a href="#">instanceIds</a>	66
6.4.4.20	<a href="#">instanceIdsToLoad</a>	66
6.4.4.21	<a href="#">instances</a>	66
6.4.4.22	<a href="#">instancesMask</a>	66
6.4.4.23	<a href="#">instancesMaskPushed</a>	66
6.4.4.24	<a href="#">levelCounts</a>	66
6.4.4.25	<a href="#">levelCountsByClass</a>	67
6.4.4.26	<a href="#">maskIsPushed</a>	67
6.4.4.27	<a href="#">missingNumericValues</a>	67
6.4.4.28	<a href="#">missingValues</a>	67
6.4.4.29	<a href="#">numericsFilename</a>	67
6.4.4.30	<a href="#">numericsIds</a>	67
6.4.4.31	<a href="#">numericsMask</a>	67
6.4.4.32	<a href="#">numericsMaskPushed</a>	67
6.4.4.33	<a href="#">numericsMinMax</a>	68
6.4.4.34	<a href="#">numericsNames</a>	68
6.4.4.35	<a href="#">phenotypesIds</a>	68
6.4.4.36	<a href="#">rng</a>	68
6.4.4.37	<a href="#">snpsFilename</a>	68
6.5	<a href="#">DatasetInstance Class Reference</a>	68

6.5.1	Detailed Description	71
6.5.2	Constructor & Destructor Documentation	72
6.5.2.1	DatasetInstance	72
6.5.2.2	~DatasetInstance	72
6.5.3	Member Function Documentation	72
6.5.3.1	AddInfluenceFactorD	72
6.5.3.2	AddNumeric	72
6.5.3.3	ClearInfluenceFactors	73
6.5.3.4	GetAttribute	73
6.5.3.5	GetClass	73
6.5.3.6	GetDatasetPtr	73
6.5.3.7	GetInfluenceFactorD	73
6.5.3.8	GetNNearestInstances	73
6.5.3.9	GetNNearestInstances	74
6.5.3.10	GetNNearestInstances	74
6.5.3.11	GetNumeric	74
6.5.3.12	GetPredictedValueTau	75
6.5.3.13	LoadInstanceFromVector	75
6.5.3.14	NumAttributes	75
6.5.3.15	NumNumerics	75
6.5.3.16	Print	75
6.5.3.17	PrintDistancePairs	75
6.5.3.18	SetClass	76
6.5.3.19	SetDistanceSums	76
6.5.3.20	SetDistanceSums	76
6.5.3.21	SetPredictedValueTau	77
6.5.3.22	SwapAttributes	77
6.5.4	Member Data Documentation	77
6.5.4.1	attributes	77
6.5.4.2	bestNeighborIds	77
6.5.4.3	bestNeighborIdsDiffClass	77
6.5.4.4	bestNeighborIdsSameClass	77
6.5.4.5	classLabel	78
6.5.4.6	dataset	78

6.5.4.7	<a href="#">neighborInfluenceFactorDs</a>	78
6.5.4.8	<a href="#">numerics</a>	78
6.5.4.9	<a href="#">predictedValueTau</a>	78
6.6	<a href="#">deref_less Class Reference</a>	78
6.6.1	<a href="#">Detailed Description</a>	78
6.6.2	<a href="#">Member Function Documentation</a>	79
6.6.2.1	<a href="#">operator()</a>	79
6.7	<a href="#">deref_less_bcw Class Reference</a>	79
6.7.1	<a href="#">Detailed Description</a>	79
6.7.2	<a href="#">Member Function Documentation</a>	79
6.7.2.1	<a href="#">operator()</a>	79
6.8	<a href="#">insilico::do_to_lower&lt; charT &gt; Class Template Reference</a>	79
6.8.1	<a href="#">Detailed Description</a>	80
6.8.2	<a href="#">Constructor &amp; Destructor Documentation</a>	80
6.8.2.1	<a href="#">do_to_lower</a>	80
6.8.2.2	<a href="#">do_to_lower</a>	80
6.8.3	<a href="#">Member Function Documentation</a>	80
6.8.3.1	<a href="#">operator()</a>	80
6.8.4	<a href="#">Member Data Documentation</a>	80
6.8.4.1	<a href="#">m_ctype</a>	80
6.9	<a href="#">insilico::do_to_upper&lt; charT &gt; Class Template Reference</a>	80
6.9.1	<a href="#">Detailed Description</a>	81
6.9.2	<a href="#">Constructor &amp; Destructor Documentation</a>	81
6.9.2.1	<a href="#">do_to_upper</a>	81
6.9.2.2	<a href="#">do_to_upper</a>	81
6.9.3	<a href="#">Member Function Documentation</a>	81
6.9.3.1	<a href="#">operator()</a>	81
6.9.4	<a href="#">Member Data Documentation</a>	81
6.9.4.1	<a href="#">m_ctype</a>	81
6.10	<a href="#">EvaporativeCooling Class Reference</a>	82
6.10.1	<a href="#">Detailed Description</a>	85
6.10.2	<a href="#">Constructor &amp; Destructor Documentation</a>	86
6.10.2.1	<a href="#">EvaporativeCooling</a>	86
6.10.2.2	<a href="#">~EvaporativeCooling</a>	86

6.10.3	Member Function Documentation	86
6.10.3.1	ComputeECScores	86
6.10.3.2	ComputeFreeEnergy	86
6.10.3.3	GetAlgorithmType	86
6.10.3.4	GetECScores	87
6.10.3.5	GetRandomJungleScores	87
6.10.3.6	GetReliefFScores	87
6.10.3.7	PrintAllScoresTabular	87
6.10.3.8	PrintAttributeScores	87
6.10.3.9	PrintKendallTaus	87
6.10.3.10	RemoveWorstAttributes	87
6.10.3.11	RunReliefF	88
6.10.3.12	WriteAttributeScores	88
6.10.4	Member Data Documentation	88
6.10.4.1	algorithmType	88
6.10.4.2	analysisType	88
6.10.4.3	dataset	88
6.10.4.4	ecScores	88
6.10.4.5	evaporatedAttributes	89
6.10.4.6	freeEnergyScores	89
6.10.4.7	numRFThreads	89
6.10.4.8	numTargetAttributes	89
6.10.4.9	numToRemovePerIteration	89
6.10.4.10	outFilesPrefix	89
6.10.4.11	paramsMap	89
6.10.4.12	randomJungle	89
6.10.4.13	reliefF	90
6.10.4.14	rfScores	90
6.10.4.15	rjScores	90
6.11	GSLRandomBase Class Reference	90
6.11.1	Detailed Description	92
6.11.2	Constructor & Destructor Documentation	92
6.11.2.1	GSLRandomBase	92
6.11.2.2	GSLRandomBase	92

6.11.2.3	<a href="#">~GSLRandomBase</a>	92
6.11.3	<a href="#">Member Function Documentation</a>	92
6.11.3.1	<a href="#">nextRandVal</a>	93
6.11.3.2	<a href="#">state</a>	93
6.11.4	<a href="#">Member Data Documentation</a>	93
6.11.4.1	<a href="#">rStatePtr_</a>	93
6.12	<a href="#">GSLRandomFlat Class Reference</a>	93
6.12.1	<a href="#">Detailed Description</a>	95
6.12.2	<a href="#">Constructor &amp; Destructor Documentation</a>	96
6.12.2.1	<a href="#">GSLRandomFlat</a>	96
6.12.2.2	<a href="#">~GSLRandomFlat</a>	96
6.12.3	<a href="#">Member Function Documentation</a>	96
6.12.3.1	<a href="#">nextRandVal</a>	96
6.12.4	<a href="#">Member Data Documentation</a>	96
6.12.4.1	<a href="#">lower_</a>	96
6.12.4.2	<a href="#">upper_</a>	96
6.13	<a href="#">insilico::is_classified&lt; Type, charT &gt; Class Template Reference</a>	96
6.13.1	<a href="#">Detailed Description</a>	97
6.13.2	<a href="#">Constructor &amp; Destructor Documentation</a>	97
6.13.2.1	<a href="#">is_classified</a>	97
6.13.2.2	<a href="#">is_classified</a>	97
6.13.3	<a href="#">Member Function Documentation</a>	97
6.13.3.1	<a href="#">operator()</a>	97
6.13.4	<a href="#">Member Data Documentation</a>	97
6.13.4.1	<a href="#">m_ctype</a>	97
6.14	<a href="#">PlinkBinaryDataset Class Reference</a>	98
6.14.1	<a href="#">Detailed Description</a>	102
6.14.2	<a href="#">Constructor &amp; Destructor Documentation</a>	102
6.14.2.1	<a href="#">PlinkBinaryDataset</a>	102
6.14.2.2	<a href="#">~PlinkBinaryDataset</a>	102
6.14.3	<a href="#">Member Function Documentation</a>	102
6.14.3.1	<a href="#">GetAttributeLevel</a>	102
6.14.3.2	<a href="#">GetAttributeMAF</a>	103
6.14.3.3	<a href="#">GetAttributeMutationType</a>	103

6.14.3.4	GetAttributeValueType . . . . .	103
6.14.3.5	GetClassValueType . . . . .	104
6.14.3.6	GetDiscreteClassLevel . . . . .	104
6.14.3.7	GetNumericClassLevel . . . . .	104
6.14.3.8	LoadSnps . . . . .	105
6.14.3.9	ReadBimFile . . . . .	105
6.14.3.10	ReadFamFile . . . . .	105
6.14.4	Member Data Documentation . . . . .	106
6.14.4.1	filenameBase . . . . .	106
6.14.4.2	missingAttributeValuesToCheck . . . . .	106
6.14.4.3	missingClassValuesToCheck . . . . .	106
6.14.4.4	numAttributesRead . . . . .	106
6.14.4.5	numClassesRead . . . . .	106
6.14.4.6	numInstancesRead . . . . .	106
6.14.4.7	validAttributeValues . . . . .	106
6.15	PlinkDataset Class Reference . . . . .	107
6.15.1	Detailed Description . . . . .	110
6.15.2	Constructor & Destructor Documentation . . . . .	111
6.15.2.1	PlinkDataset . . . . .	111
6.15.2.2	~PlinkDataset . . . . .	111
6.15.3	Member Function Documentation . . . . .	111
6.15.3.1	GetAttributeMAF . . . . .	111
6.15.3.2	GetAttributeMutationType . . . . .	111
6.15.3.3	GetClassValueType . . . . .	112
6.15.3.4	GetDiscreteClassLevel . . . . .	112
6.15.3.5	GetNumericClassLevel . . . . .	112
6.15.3.6	LoadSnps . . . . .	113
6.15.4	Member Data Documentation . . . . .	113
6.15.4.1	filenameBase . . . . .	113
6.15.4.2	missingClassValuesToCheck . . . . .	113
6.16	PlinkRawDataset Class Reference . . . . .	113
6.16.1	Detailed Description . . . . .	116
6.16.2	Constructor & Destructor Documentation . . . . .	116
6.16.2.1	PlinkRawDataset . . . . .	116

6.16.2.2	<a href="#">~PlinkRawDataset</a>	117
6.16.3	Member Function Documentation	117
6.16.3.1	<a href="#">GetClassValueType</a>	117
6.16.3.2	<a href="#">GetDiscreteClassLevel</a>	117
6.16.3.3	<a href="#">GetNumericClassLevel</a>	117
6.16.3.4	<a href="#">LoadSnps</a>	118
6.17	RandomJungle Class Reference	118
6.17.1	Detailed Description	120
6.17.2	Constructor & Destructor Documentation	121
6.17.2.1	<a href="#">RandomJungle</a>	121
6.17.2.2	<a href="#">~RandomJungle</a>	121
6.17.3	Member Function Documentation	121
6.17.3.1	<a href="#">ComputeAttributeScores</a>	121
6.17.3.2	<a href="#">GetScores</a>	121
6.17.3.3	<a href="#">ReadScores</a>	121
6.17.4	Member Data Documentation	121
6.17.4.1	<a href="#">dataset</a>	121
6.17.4.2	<a href="#">rjParams</a>	122
6.17.4.3	<a href="#">scores</a>	122
6.18	ReliefF Class Reference	122
6.18.1	Detailed Description	126
6.18.2	Constructor & Destructor Documentation	127
6.18.2.1	<a href="#">ReliefF</a>	127
6.18.2.2	<a href="#">ReliefF</a>	127
6.18.2.3	<a href="#">~ReliefF</a>	127
6.18.3	Member Function Documentation	127
6.18.3.1	<a href="#">ComputeAttributeScores</a>	127
6.18.3.2	<a href="#">ComputeAttributeScoresCleanSnps</a>	128
6.18.3.3	<a href="#">ComputeAttributeScoresIteratively</a>	128
6.18.3.4	<a href="#">ComputeInstanceToInstanceDistance</a>	128
6.18.3.5	<a href="#">ComputeWeightByDistanceFactors</a>	128
6.18.3.6	<a href="#">GetScores</a>	128
6.18.3.7	<a href="#">PreComputeDistances</a>	128
6.18.3.8	<a href="#">PreComputeDistancesByMap</a>	129

6.18.3.9	<a href="#">PrintAttributeScores</a>	129
6.18.3.10	<a href="#">ProcessExclusionFile</a>	129
6.18.3.11	<a href="#">ResetForNextIteration</a>	129
6.18.3.12	<a href="#">WriteAttributeScores</a>	129
6.18.4	<a href="#">Member Data Documentation</a>	130
6.18.4.1	<a href="#">analysisType</a>	130
6.18.4.2	<a href="#">dataset</a>	130
6.18.4.3	<a href="#">doRemovePercent</a>	130
6.18.4.4	<a href="#">finalScores</a>	130
6.18.4.5	<a href="#">k</a>	130
6.18.4.6	<a href="#">m</a>	130
6.18.4.7	<a href="#">numDiff</a>	130
6.18.4.8	<a href="#">numMetric</a>	131
6.18.4.9	<a href="#">one_over_m_times_k</a>	131
6.18.4.10	<a href="#">randomlySelect</a>	131
6.18.4.11	<a href="#">removePercentage</a>	131
6.18.4.12	<a href="#">removePerIteration</a>	131
6.18.4.13	<a href="#">scoreNames</a>	131
6.18.4.14	<a href="#">snpDiff</a>	132
6.18.4.15	<a href="#">snpMetric</a>	132
6.18.4.16	<a href="#">W</a>	132
6.18.4.17	<a href="#">weightByDistanceMethod</a>	132
6.18.4.18	<a href="#">weightByDistanceSigma</a>	132
6.19	<a href="#">RReliefF Class Reference</a>	133
6.19.1	<a href="#">Detailed Description</a>	136
6.19.2	<a href="#">Constructor &amp; Destructor Documentation</a>	136
6.19.2.1	<a href="#">RReliefF</a>	136
6.19.2.2	<a href="#">RReliefF</a>	136
6.19.2.3	<a href="#">~RReliefF</a>	137
6.19.3	<a href="#">Member Function Documentation</a>	137
6.19.3.1	<a href="#">ComputeAttributeScores</a>	137
<b>7</b>	<b><a href="#">File Documentation</a></b>	<b>139</b>
7.1	<a href="#">src/library/ArffDataset.cpp File Reference</a>	139



7.2	src/library/ArffDataset.h File Reference . . . . .	140
7.2.1	Enumeration Type Documentation . . . . .	141
7.2.1.1	ArffAttributeType . . . . .	141
7.3	src/library/best_n.h File Reference . . . . .	141
7.3.1	Detailed Description . . . . .	143
7.4	src/library/ChiSquared.cpp File Reference . . . . .	143
7.5	src/library/ChiSquared.h File Reference . . . . .	144
7.6	src/library/CleanSnpDataset.cpp File Reference . . . . .	145
7.7	src/library/CleanSnpDataset.h File Reference . . . . .	145
7.8	src/library/Dataset.cpp File Reference . . . . .	146
7.9	src/library/Dataset.h File Reference . . . . .	148
7.9.1	Enumeration Type Documentation . . . . .	149
7.9.1.1	AttributeMutationType . . . . .	149
7.9.1.2	AttributeType . . . . .	150
7.9.1.3	OutputDatasetType . . . . .	150
7.9.1.4	ValueType . . . . .	150
7.9.2	Variable Documentation . . . . .	151
7.9.2.1	INVALID_ATTRIBUTE_VALUE . . . . .	151
7.9.2.2	INVALID_DISCRETE_CLASS_VALUE . . . . .	151
7.9.2.3	INVALID_DISTANCE . . . . .	151
7.9.2.4	INVALID_INDEX . . . . .	151
7.9.2.5	INVALID_NUMERIC_CLASS_VALUE . . . . .	151
7.9.2.6	INVALID_NUMERIC_VALUE . . . . .	151
7.9.2.7	MISSING_ATTRIBUTE_VALUE . . . . .	151
7.9.2.8	MISSING_DISCRETE_CLASS_VALUE . . . . .	152
7.9.2.9	MISSING_NUMERIC_CLASS_VALUE . . . . .	152
7.9.2.10	MISSING_NUMERIC_VALUE . . . . .	152
7.10	src/library/DatasetInstance.cpp File Reference . . . . .	152
7.10.1	Typedef Documentation . . . . .	153
7.10.1.1	T . . . . .	153
7.11	src/library/DatasetInstance.h File Reference . . . . .	153
7.11.1	Typedef Documentation . . . . .	154
7.11.1.1	AttributeLevel . . . . .	154
7.11.1.2	ClassLevel . . . . .	154

7.11.1.3	<a href="#">DistancePair</a>	155
7.11.1.4	<a href="#">DistancePairs</a>	155
7.11.1.5	<a href="#">DistancePairsIt</a>	155
7.11.1.6	<a href="#">NumericLevel</a>	155
7.12	<a href="#">src/library/Debugging.h File Reference</a>	155
7.12.1	<a href="#">Detailed Description</a>	156
7.12.2	<a href="#">Function Documentation</a>	156
7.12.2.1	<a href="#">PrintVector</a>	156
7.12.2.2	<a href="#">PrintVector</a>	157
7.13	<a href="#">src/library/DistanceMetrics.cpp File Reference</a>	157
7.13.1	<a href="#">Function Documentation</a>	158
7.13.1.1	<a href="#">CheckMissing</a>	158
7.13.1.2	<a href="#">CheckMissingNumeric</a>	158
7.13.1.3	<a href="#">diffAMM</a>	159
7.13.1.4	<a href="#">diffGMM</a>	159
7.13.1.5	<a href="#">diffManhattan</a>	159
7.13.1.6	<a href="#">diffPredictedValueTau</a>	160
7.13.1.7	<a href="#">norm</a>	160
7.14	<a href="#">src/library/DistanceMetrics.h File Reference</a>	161
7.14.1	<a href="#">Detailed Description</a>	161
7.14.2	<a href="#">Function Documentation</a>	162
7.14.2.1	<a href="#">CheckMissing</a>	162
7.14.2.2	<a href="#">CheckMissingNumeric</a>	162
7.14.2.3	<a href="#">diffAMM</a>	162
7.14.2.4	<a href="#">diffGMM</a>	163
7.14.2.5	<a href="#">diffManhattan</a>	163
7.14.2.6	<a href="#">diffPredictedValueTau</a>	164
7.14.2.7	<a href="#">norm</a>	164
7.15	<a href="#">src/library/EvaporativeCooling.cpp File Reference</a>	164
7.15.1	<a href="#">Function Documentation</a>	165
7.15.1.1	<a href="#">scoresSortAsc</a>	165
7.15.1.2	<a href="#">scoresSortAscByName</a>	165
7.15.1.3	<a href="#">scoresSortDesc</a>	165
7.16	<a href="#">src/library/EvaporativeCooling.h File Reference</a>	166

7.16.1	Typedef Documentation	167
7.16.1.1	EcScores	167
7.16.1.2	EcScoresClt	167
7.16.1.3	EcScoresIt	167
7.16.2	Enumeration Type Documentation	167
7.16.2.1	EcAlgorithmType	167
7.17	src/library/FileSystemUtils.cpp File Reference	168
7.17.1	Function Documentation	168
7.17.1.1	GetFileBasename	168
7.17.1.2	GetFileExtension	168
7.18	src/library/FileSystemUtils.h File Reference	169
7.18.1	Detailed Description	169
7.18.2	Function Documentation	170
7.18.2.1	GetFileBasename	170
7.18.2.2	GetFileExtension	170
7.19	src/library/GSLRandomBase.h File Reference	170
7.20	src/library/GSLRandomFlat.h File Reference	171
7.21	src/library/Insilico.cpp File Reference	172
7.21.1	Function Documentation	173
7.21.1.1	ChooseSnpsDatasetByExtension	173
7.21.1.2	GetMatchingIds	173
7.21.1.3	LoadIndividualIds	174
7.21.1.4	Timestamp	174
7.22	src/library/Insilico.h File Reference	174
7.22.1	Detailed Description	175
7.22.2	Function Documentation	175
7.22.2.1	ChooseSnpsDatasetByExtension	175
7.22.2.2	GetMatchingIds	176
7.22.2.3	LoadIndividualIds	176
7.22.2.4	Timestamp	176
7.22.3	Variable Documentation	177
7.22.3.1	COMMAND_LINE_ERROR	177
7.23	src/library/PlinkBinaryDataset.cpp File Reference	177
7.24	src/library/PlinkBinaryDataset.h File Reference	178

7.25	src/library/PlinkDataset.cpp File Reference	179
7.26	src/library/PlinkDataset.h File Reference	179
7.26.1	Enumeration Type Documentation	181
7.26.1.1	MapFileType	181
7.27	src/library/PlinkRawDataset.cpp File Reference	181
7.28	src/library/PlinkRawDataset.h File Reference	181
7.29	src/library/RandomJungle.cpp File Reference	182
7.30	src/library/RandomJungle.h File Reference	183
7.31	src/library/ReliefF.cpp File Reference	184
7.31.1	Typedef Documentation	186
7.31.1.1	AttributeIndex	186
7.31.1.2	AttributeIndexIt	186
7.31.1.3	ScoresMap	186
7.31.1.4	ScoresMapIt	186
7.31.1.5	T	187
7.31.2	Function Documentation	187
7.31.2.1	attributeSort	187
7.31.2.2	librelieff_is_present	187
7.31.2.3	scoreSort	187
7.32	src/library/ReliefF.h File Reference	187
7.32.1	Enumeration Type Documentation	189
7.32.1.1	AnalysisType	189
7.32.2	Function Documentation	189
7.32.2.1	librelieff_is_present	189
7.33	src/library/RReliefF.cpp File Reference	189
7.34	src/library/RReliefF.h File Reference	190
7.35	src/library/Statistics.cpp File Reference	191
7.35.1	Define Documentation	192
7.35.1.1	DEBUG_E	192
7.35.1.2	DEBUG_Z	192
7.35.2	Function Documentation	193
7.35.2.1	ConditionalEntropy	193
7.35.2.2	ConstructAttributeCart	193
7.35.2.3	Entropy	193

7.35.2.4	KendallTau	193
7.35.2.5	KendallTau	193
7.35.2.6	KendallTau	193
7.35.2.7	ZTransform	193
7.36	src/library/Statistics.h File Reference	194
7.36.1	Typedef Documentation	195
7.36.1.1	Histogram	195
7.36.1.2	HistogramIt	195
7.36.1.3	VectorDouble	196
7.36.1.4	VectorDoubleIt	196
7.36.2	Function Documentation	196
7.36.2.1	ConditionalEntropy	196
7.36.2.2	ConstructAttributeCart	196
7.36.2.3	Entropy	197
7.36.2.4	KendallTau	197
7.36.2.5	KendallTau	197
7.36.2.6	KendallTau	197
7.36.2.7	VarStd	198
7.36.2.8	ZTransform	198
7.37	src/library/StringUtils.h File Reference	198
7.37.1	Detailed Description	201



# Chapter 1

## Namespace Index

### 1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<a href="#">insilico</a> . . . . .	9
------------------------------------	---





## Chapter 2

# Class Index

### 2.1 Class Hierarchy

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

ChiSquared . . . . .	21
Dataset . . . . .	31
ArffDataset . . . . .	15
CleanSnpDataset . . . . .	27
PlinkBinaryDataset . . . . .	98
PlinkDataset . . . . .	107
PlinkRawDataset . . . . .	113
DatasetInstance . . . . .	68
deref_less . . . . .	78
deref_less_bcw . . . . .	79
insilico::do_to_lower< charT > . . . . .	79
insilico::do_to_upper< charT > . . . . .	80
EvaporativeCooling . . . . .	82
GSLRandomBase . . . . .	90
GSLRandomFlat . . . . .	93
insilico::is_classified< Type, charT > . . . . .	96
RandomJungle . . . . .	118
ReliefF . . . . .	122
RReliefF . . . . .	133



## Chapter 3

# Class Index

### 3.1 Class List

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

<a href="#">ArffDataset</a> (ARFF file format reader) . . . . .	15
<a href="#">ChiSquared</a> (Chi-squared attribute ranking algorithm) . . . . .	21
<a href="#">CleanSnpDataSet</a> (Experiemnetal data set reader for large/GWAS data) . . . . .	27
<a href="#">Dataset</a> (Base class for collections of instances containing attributea and class) . . . . .	31
<a href="#">DatasetInstance</a> (Class to hold dataset instances (rows of attributes) ) . . . . .	68
<a href="#">deref_less</a> . . . . .	78
<a href="#">deref_less_bcw</a> . . . . .	79
<a href="#">insilico::do_to_lower&lt; charT &gt;</a> . . . . .	79
<a href="#">insilico::do_to_upper&lt; charT &gt;</a> . . . . .	80
<a href="#">EvaporativeCooling</a> (Evaporative Cooling attribute ranking algorithm) . . . . .	82
<a href="#">GSLRandomBase</a> (A base class for GNU Scientific Library (GSL) random number functions) . . . . .	90
<a href="#">GSLRandomFlat</a> (Random numbers in a flat, or uniform distribution) . . . . .	93
<a href="#">insilico::is_classified&lt; Type, charT &gt;</a> . . . . .	96
<a href="#">PlinkBinaryDataset</a> (Plink binary PED/BED file format reader) . . . . .	98
<a href="#">PlinkDataset</a> (Plink MAP/PED file format reader) . . . . .	107
<a href="#">PlinkRawDataset</a> (Plink recodeA/RAW file format reader) . . . . .	113
<a href="#">RandomJungle</a> ( <a href="#">RandomJungle</a> attribute ranking algorithm) . . . . .	118
<a href="#">ReliefF</a> ( <a href="#">ReliefF</a> attribute ranking algorithm) . . . . .	122
<a href="#">RReliefF</a> (Regression <a href="#">ReliefF</a> attribute ranking algorithm) . . . . .	133



## Chapter 4

# File Index

### 4.1 File List

Here is a list of all files with brief descriptions:

src/library/ <a href="#">ArffDataset.cpp</a> . . . . .	139
src/library/ <a href="#">ArffDataset.h</a> . . . . .	140
src/library/ <a href="#">best_n.h</a> (Find the best n keeping original order for ties - stable sort )	141
src/library/ <a href="#">ChiSquared.cpp</a> . . . . .	143
src/library/ <a href="#">ChiSquared.h</a> . . . . .	144
src/library/ <a href="#">CleanSnpDataset.cpp</a> . . . . .	145
src/library/ <a href="#">CleanSnpDataset.h</a> . . . . .	145
src/library/ <a href="#">Dataset.cpp</a> . . . . .	146
src/library/ <a href="#">Dataset.h</a> . . . . .	148
src/library/ <a href="#">DatasetInstance.cpp</a> . . . . .	152
src/library/ <a href="#">DatasetInstance.h</a> . . . . .	153
src/library/ <a href="#">Debugging.h</a> (Debugging utilities ) . . . . .	155
src/library/ <a href="#">DistanceMetrics.cpp</a> . . . . .	157
src/library/ <a href="#">DistanceMetrics.h</a> (Distance metrics for <a href="#">ReliefF</a> ) . . . . .	161
src/library/ <a href="#">EvaporativeCooling.cpp</a> . . . . .	164
src/library/ <a href="#">EvaporativeCooling.h</a> . . . . .	166
src/library/ <a href="#">FilesystemUtils.cpp</a> . . . . .	168
src/library/ <a href="#">FilesystemUtils.h</a> (Filesystem utilities ) . . . . .	169
src/library/ <a href="#">GSLRandomBase.h</a> . . . . .	170
src/library/ <a href="#">GSLRandomFlat.h</a> . . . . .	171
src/library/ <a href="#">Insilico.cpp</a> . . . . .	172
src/library/ <a href="#">Insilico.h</a> (Common functions for Insilico Lab projects ) . . . . .	174
src/library/ <a href="#">PlinkBinaryDataset.cpp</a> . . . . .	177
src/library/ <a href="#">PlinkBinaryDataset.h</a> . . . . .	178
src/library/ <a href="#">PlinkDataset.cpp</a> . . . . .	179
src/library/ <a href="#">PlinkDataset.h</a> . . . . .	179
src/library/ <a href="#">PlinkRawDataset.cpp</a> . . . . .	181
src/library/ <a href="#">PlinkRawDataset.h</a> . . . . .	181
src/library/ <a href="#">RandomJungle.cpp</a> . . . . .	182

src/library/ <a href="#">RandomJungle.h</a> . . . . .	183
src/library/ <a href="#">ReliefF.cpp</a> . . . . .	184
src/library/ <a href="#">ReliefF.h</a> . . . . .	187
src/library/ <a href="#">RReliefF.cpp</a> . . . . .	189
src/library/ <a href="#">RReliefF.h</a> . . . . .	190
src/library/ <a href="#">Statistics.cpp</a> . . . . .	191
src/library/ <a href="#">Statistics.h</a> . . . . .	194
src/library/ <a href="#">StringUtils.h</a> (Various string-related utilities ) . . . . .	198

## Chapter 5

# Namespace Documentation

### 5.1 insilico Namespace Reference

#### Classes

- class [is\\_classified](#)
- class [do\\_to\\_upper](#)
- class [do\\_to\\_lower](#)

#### Functions

- template<typename InputIt , typename OutputIt , typename Comp >  
void [best\\_n](#) (InputIt begin, InputIt end, OutputIt out, size\_t n, Comp comp)  
*Get the best n values with ties keeping same original order.*
- template<typename stringT >  
stringT [trim\\_left](#) (const stringT &s, const std::locale &loc=std::locale())
- template<typename stringT >  
stringT [trim\\_right](#) (const stringT &s, const std::locale &loc=std::locale())
- template<typename stringT >  
stringT [trim](#) (const stringT &s, const std::locale &loc=std::locale())
- template<typename Container , typename stringT >  
void [split](#) (Container &cont, const stringT &s, const std::locale &loc=std::locale())
- template<typename Container , typename stringT >  
void [split](#) (Container &cont, const stringT &s, const stringT &delim)
- template<typename Container , typename stringT , typename Pred >  
void [split\\_if](#) (Container &cont, const stringT &s, const Pred &pred)
- template<typename It , typename stringT >  
stringT [join](#) (const It &begin, const It &end, const stringT &delim)
- template<typename stringT >  
stringT [to\\_upper](#) (const stringT &str, const std::locale &loc=std::locale())
- template<typename stringT >  
stringT [to\\_lower](#) (const stringT &str, const std::locale &loc=std::locale())

- `std::string trim_left` (const char \*s, const std::locale &loc=std::locale())
- `std::wstring trim_left` (const wchar\_t \*s, const std::locale &loc=std::locale())
- `std::string trim_right` (const char \*s, const std::locale &loc=std::locale())
- `std::wstring trim_right` (const wchar\_t \*s, const std::locale &loc=std::locale())
- `std::string trim` (const char \*s, const std::locale &loc=std::locale())
- `std::wstring trim` (const wchar\_t \*s, const std::locale &loc=std::locale())
- `template<typename Container >`  
`void split` (Container &cont, const char \*s, const std::locale &loc=std::locale())
- `template<typename Container >`  
`void split` (Container &cont, const wchar\_t \*s, const std::locale &loc=std::locale())
- `template<typename Container >`  
`void split` (Container &cont, const std::string &s, const char \*delim)
- `template<typename Container >`  
`void split` (Container &cont, const char \*s, const std::string &delim)
- `template<typename Container >`  
`void split` (Container &cont, const char \*s, const char \*delim)
- `template<typename Container >`  
`void split` (Container &cont, const std::wstring &s, const wchar\_t \*delim)
- `template<typename Container >`  
`void split` (Container &cont, const wchar\_t \*s, const std::wstring &delim)
- `template<typename Container >`  
`void split` (Container &cont, const wchar\_t \*s, const wchar\_t \*delim)
- `template<typename Container , typename Pred >`  
`void split_if` (Container &cont, const char \*s, const Pred &pred)
- `template<typename Container , typename Pred >`  
`void split_if` (Container &cont, const wchar\_t \*s, const Pred &pred)
- `template<typename It >`  
`std::string join` (const It &begin, const It &end, const char \*delim)
- `template<typename It >`  
`std::wstring join` (const It &begin, const It &end, const wchar\_t \*delim)
- `std::string to_upper` (const char \*s, const std::locale &loc=std::locale())
- `std::wstring to_upper` (const wchar\_t \*s, const std::locale &loc=std::locale())
- `std::string to_lower` (const char \*s, const std::locale &loc=std::locale())
- `std::wstring to_lower` (const wchar\_t \*s, const std::locale &loc=std::locale())
- `template<typename T >`  
`std::string get_bits` (T value)
- `template<typename T >`  
`std::string zeroPadNumber` (T num, int padSize)

### 5.1.1 Function Documentation

5.1.1.1 `template<typename InputIt , typename OutputIt , typename Comp > void insilico::best.n`  
 ( InputIt *begin*, InputIt *end*, OutputIt *out*, size\_t *n*, Comp *comp* )

Get the best n values with ties keeping same original order.

#### Parameters



in	<i>begin</i>	iterator of the beginning of a input container
in	<i>end</i>	iterator of the end of a input container
out	<i>out</i>	iterator of the beginning of a output container
in	<i>size</i>	best n value
in	<i>comp</i>	compare functor

**Returns**

path/filename without extension

Definition at line 30 of file best\_n.h.

#### 5.1.1.2 `template<typename T> std::string insilico::get_bits ( T value )`

Definition at line 324 of file StringUtils.h.

#### 5.1.1.3 `template<typename It , typename stringT> stringT insilico::join ( const It & begin, const It & end, const stringT & delim )`

Definition at line 198 of file StringUtils.h.

#### 5.1.1.4 `template<typename It> std::string insilico::join ( const It & begin, const It & end, const char * delim ) [inline]`

Definition at line 300 of file StringUtils.h.

#### 5.1.1.5 `template<typename It> std::wstring insilico::join ( const It & begin, const It & end, const wchar_t * delim ) [inline]`

Definition at line 304 of file StringUtils.h.

#### 5.1.1.6 `template<typename Container , typename stringT> void insilico::split ( Container & cont, const stringT & s, const std::locale & loc = std::locale() ) [inline]`

Definition at line 148 of file StringUtils.h.

#### 5.1.1.7 `template<typename Container> void insilico::split ( Container & cont, const char * s, const std::locale & loc = std::locale() ) [inline]`

Definition at line 258 of file StringUtils.h.

**5.1.1.8** `template<typename Container > void insilico::split ( Container & cont, const wchar_t * s, const std::locale & loc = std:: : locale() ) [inline]`

Definition at line 263 of file StringUtils.h.

**5.1.1.9** `template<typename Container > void insilico::split ( Container & cont, const std::string & s, const char * delim ) [inline]`

Definition at line 268 of file StringUtils.h.

**5.1.1.10** `template<typename Container > void insilico::split ( Container & cont, const char * s, const std::string & delim ) [inline]`

Definition at line 272 of file StringUtils.h.

**5.1.1.11** `template<typename Container > void insilico::split ( Container & cont, const char * s, const char * delim ) [inline]`

Definition at line 276 of file StringUtils.h.

**5.1.1.12** `template<typename Container > void insilico::split ( Container & cont, const std::wstring & s, const wchar_t * delim ) [inline]`

Definition at line 280 of file StringUtils.h.

**5.1.1.13** `template<typename Container > void insilico::split ( Container & cont, const wchar_t * s, const std::wstring & delim ) [inline]`

Definition at line 284 of file StringUtils.h.

**5.1.1.14** `template<typename Container , typename stringT > void insilico::split ( Container & cont, const stringT & s, const stringT & delim )`

Definition at line 156 of file StringUtils.h.

**5.1.1.15** `template<typename Container > void insilico::split ( Container & cont, const wchar_t * s, const wchar_t * delim ) [inline]`

Definition at line 288 of file StringUtils.h.

**5.1.1.16** `template<typename Container , typename Pred > void insilico::split_if ( Container & cont, const char * s, const Pred & pred ) [inline]`

Definition at line 292 of file StringUtils.h.

**5.1.1.17** `template<typename Container , typename Pred > void insilico::split_if ( Container & cont, const wchar_t * s, const Pred & pred ) [inline]`

Definition at line 296 of file StringUtils.h.

**5.1.1.18** `template<typename Container , typename stringT , typename Pred > void insilico::split_if ( Container & cont, const stringT & s, const Pred & pred )`

Definition at line 178 of file StringUtils.h.

**5.1.1.19** `std::wstring insilico::to_lower ( const wchar_t * s, const std::locale & loc = std::locale() ) [inline]`

Definition at line 319 of file StringUtils.h.

**5.1.1.20** `template<typename stringT > stringT insilico::to_lower ( const stringT & str, const std::locale & loc = std::locale() )`

Definition at line 224 of file StringUtils.h.

**5.1.1.21** `std::string insilico::to_lower ( const char * s, const std::locale & loc = std::locale() ) [inline]`

Definition at line 315 of file StringUtils.h.

**5.1.1.22** `template<typename stringT > stringT insilico::to_upper ( const stringT & str, const std::locale & loc = std::locale() )`

Definition at line 214 of file StringUtils.h.

**5.1.1.23** `std::wstring insilico::to_upper ( const wchar_t * s, const std::locale & loc = std::locale() ) [inline]`

Definition at line 311 of file StringUtils.h.

**5.1.1.24** `std::string insilico::to_upper ( const char * s, const std::locale & loc = std::locale() ) [inline]`

Definition at line 307 of file StringUtils.h.

**5.1.1.25** `std::wstring insilico::trim ( const wchar_t * s, const std::locale & loc = std::locale() ) [inline]`

Definition at line 253 of file StringUtils.h.

**5.1.1.26** `std::string insilico::trim ( const char * s, const std::locale & loc =  
std::locale() ) [inline]`

Definition at line 249 of file StringUtils.h.

**5.1.1.27** `template<typename stringT > stringT insilico::trim ( const stringT & s, const  
std::locale & loc = std::locale() )`

Definition at line 123 of file StringUtils.h.

**5.1.1.28** `template<typename stringT > stringT insilico::trim_left ( const stringT & s, const  
std::locale & loc = std::locale() )`

Definition at line 101 of file StringUtils.h.

**5.1.1.29** `std::wstring insilico::trim_left ( const wchar_t * s, const std::locale & loc =  
std::locale() ) [inline]`

Definition at line 237 of file StringUtils.h.

**5.1.1.30** `std::string insilico::trim_left ( const char * s, const std::locale & loc =  
std::locale() ) [inline]`

Definition at line 233 of file StringUtils.h.

**5.1.1.31** `std::string insilico::trim_right ( const char * s, const std::locale & loc =  
std::locale() ) [inline]`

Definition at line 241 of file StringUtils.h.

**5.1.1.32** `std::wstring insilico::trim_right ( const wchar_t * s, const std::locale & loc =  
std::locale() ) [inline]`

Definition at line 245 of file StringUtils.h.

**5.1.1.33** `template<typename stringT > stringT insilico::trim_right ( const stringT & s, const  
std::locale & loc = std::locale() )`

Definition at line 112 of file StringUtils.h.

**5.1.1.34** `template<typename T > std::string insilico::zeroPadNumber ( T num, int padSize )`

Definition at line 333 of file StringUtils.h.

## Chapter 6

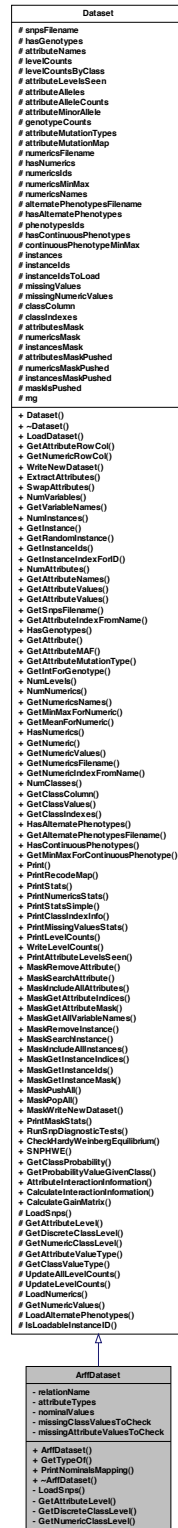
# Class Documentation

### 6.1 ArffDataset Class Reference

ARFF file format reader.

```
#include <ArffDataset.h>
```

Inheritance diagram for ArffDataset:



Collaboration diagram for ArffDataset:



## Public Member Functions

- [ArffDataset](#) ()
- [ArffAttributeType GetTypeOf](#) (unsigned int columnIndex)
- void [PrintNominalsMapping](#) ()
- [~ArffDataset](#) ()

## Private Member Functions

- bool [LoadSnps](#) (std::string filename)  
*Load SNPs from file using the data set filename.*
- bool [GetAttributeLevel](#) (std::string inLevel, std::vector< std::string > [missingValues](#), [AttributeLevel](#) &outLevel)  
*Get the attribute level based on string representation.*
- bool [GetDiscreteClassLevel](#) (std::string inLevel, std::vector< std::string > [missingValues](#), [ClassLevel](#) &outLevel)  
*Get the discrete class level based on string representation.*
- bool [GetNumericClassLevel](#) (std::string inLevel, std::vector< std::string > [missingValues](#), [NumericLevel](#) &outLevel)  
*Get the numeric class level based on string representation.*

## Private Attributes

- std::string [relationName](#)  
*ARFF relation name.*
- std::vector< [ArffAttributeType](#) > [attributeTypes](#)  
*vector of attribute types*
- std::map< std::string, std::vector< std::string > > [nominalValues](#)  
*map of attribute names to valid nominal values*
- std::vector< std::string > [missingClassValuesToCheck](#)  
*missing class values*
- std::vector< std::string > [missingAttributeValuesToCheck](#)  
*missing attribute values*

### 6.1.1 Detailed Description

ARFF file format reader.

<http://www.cs.waikato.ac.nz/ml/weka/arff.html>

#### See also

[Dataset](#)

#### Author

Bill White



**Version**

1.0

Contact: [bill.c.white@gmail.com](mailto:bill.c.white@gmail.com) Created on: 2/24/11

Definition at line 38 of file ArffDataset.h.

**6.1.2 Constructor & Destructor Documentation****6.1.2.1 ArffDataset::ArffDataset ( )**

Definition at line 25 of file ArffDataset.cpp.

**6.1.2.2 ArffDataset::~~ArffDataset ( )****6.1.3 Member Function Documentation****6.1.3.1 bool ArffDataset::GetAttributeLevel ( std::string *inLevel*, std::vector< std::string > *missingValues*, AttributeLevel & *outLevel* ) [private, virtual]**

Get the attribute level based on string representation.

**Parameters**

in	<i>inLevel</i>	attribute level read from file
in	<i>missingValues</i>	list of strings representing missing attribute values
out	<i>outLevel</i>	attribute level to use in the data set class

**Returns**

success

Reimplemented from [Dataset](#).**6.1.3.2 bool ArffDataset::GetDiscreteClassLevel ( std::string *inLevel*, std::vector< std::string > *missingValues*, ClassLevel & *outLevel* ) [private, virtual]**

Get the discrete class level based on string representation.

**Parameters**

in	<i>inLevel</i>	class level read from file
in	<i>missingValues</i>	list of strings representing missing class values
out	<i>outLevel</i>	discrete class level to use in the data set class

**Returns**

success

Reimplemented from [Dataset](#).

**6.1.3.3** `bool ArffDataset::GetNumericClassLevel ( std::string inLevel, std::vector< std::string > missingValues, NumericLevel & outLevel )` [private, virtual]

Get the numeric class level based on string representation.

**Parameters**

in	<i>inLevel</i>	class level read from file
in	<i>missingValues</i>	list of strings representing missing class values
in, out	<i>outLevel</i>	numeric class level to use in the data set class

**Returns**

success

Reimplemented from [Dataset](#).

**6.1.3.4** `ArffAttributeType ArffDataset::GetTypeOf ( unsigned int columnIndex )`

Definition at line 370 of file ArffDataset.cpp.

**6.1.3.5** `bool ArffDataset::LoadSnps ( std::string filename )` [private, virtual]

Load SNPs from file using the data set filename.

**Parameters**

in	<i>filename</i>	SNPs filename
in	<i>deRecodeA</i>	perform a recodeA operation after reading raw data?

**Returns**

success

Reimplemented from [Dataset](#).

**6.1.3.6** `void ArffDataset::PrintNominalsMapping ( )`

Definition at line 377 of file ArffDataset.cpp.

### 6.1.4 Member Data Documentation

**6.1.4.1** `std::vector<ArffAttributeType> ArffDataset::attributeTypes` [private]

vector of attribute types

Definition at line 70 of file ArffDataset.h.

**6.1.4.2** `std::vector<std::string> ArffDataset::missingAttributeValuesToCheck`  
[private]

missing attribute values

Definition at line 77 of file ArffDataset.h.

**6.1.4.3** `std::vector<std::string> ArffDataset::missingClassValuesToCheck`  
[private]

missing class values

Definition at line 75 of file ArffDataset.h.

**6.1.4.4** `std::map<std::string, std::vector<std::string> > ArffDataset::nominalValues`  
[private]

map of attribute names to valid nominal values

Definition at line 72 of file ArffDataset.h.

**6.1.4.5** `std::string ArffDataset::relationName` [private]

ARFF relation name.

Definition at line 68 of file ArffDataset.h.

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

- [src/library/ArffDataset.h](#)
- [src/library/ArffDataset.cpp](#)

## 6.2 ChiSquared Class Reference

Chi-squared attribute ranking algorithm.

```
#include <ChiSquared.h>
```

Collaboration diagram for ChiSquared:



## Public Member Functions

- [ChiSquared](#) ([Dataset](#) \*ds)  
*Construct an chi-squared algorithm object.*
- [~ChiSquared](#) ()
- const std::vector< std::pair< double, double > > & [ComputeScores](#) ()  
*For each attribute, calculate chi-squared and associated p-value.*
- std::pair< double, double > [ComputeScore](#) (unsigned int index)  
*For the attribute at the specified index, calculate the chi-squared and associated p-value.*
- void [PrintTables](#) ()  
*Print calculation tables.*
- void [PrintScores](#) (std::ofstream &outStream, unsigned int topN=0)  
*Print the scores to a stream.*
- void [WriteScores](#) (std::string outFilename, unsigned int topN=0)  
*Print the scores to a stream.*
- std::vector< std::vector< double > > [GetFrequencyCounts](#) ()  
*Get the observed frequencies table as a vector of vector of doubles.*

## Private Member Functions

- void [PrepareForAttribute](#) (unsigned int attributeIndex)  
*Private method to setup the chi-squared contingency tables for a particular attribute.*
- void [ClearTables](#) ()  
*Clear calculation tables.*

## Private Attributes

- [Dataset](#) \* [dataset](#)  
*pointer to a [Dataset](#) object*
- unsigned int [numLevels](#)  
*number of levels in the attributes*
- unsigned int [numClasses](#)  
*number of classes in the instances*
- std::vector< std::vector< double > > [observedFreqTable](#)  
*observed frequencies*
- std::vector< std::vector< double > > [expectedContingencyTable](#)
- std::vector< std::vector< double > > [chiSquaredValues](#)  
*chi squared computed values*
- std::vector< std::pair< double, double > > [scores](#)  
*chi-squared value, p-value for each attribute*

### 6.2.1 Detailed Description

Chi-squared attribute ranking algorithm.

[ChiSquared](#) algorithm interface. For performing chi-squared tests of association between an attribute and its class across all instances in a data set.

#### Author

Bill White

#### Version

1.0

Contact: [bill.c.white@gmail.com](mailto:bill.c.white@gmail.com) Created on: 6/15/05

Definition at line 25 of file ChiSquared.h.

### 6.2.2 Constructor & Destructor Documentation

#### 6.2.2.1 ChiSquared::ChiSquared ( Dataset \* ds )

Construct an chi-squared algorithm object.

#### Parameters

<i>in</i>	<i>ds</i>	pointer to a <a href="#">Dataset</a> object
-----------	-----------	---

Definition at line 21 of file ChiSquared.cpp.

#### 6.2.2.2 ChiSquared::~~ChiSquared ( )

Definition at line 31 of file ChiSquared.cpp.

### 6.2.3 Member Function Documentation

#### 6.2.3.1 void ChiSquared::ClearTables ( ) [private]

Clear calculation tables.

Definition at line 230 of file ChiSquared.cpp.

#### 6.2.3.2 pair< double, double > ChiSquared::ComputeScore ( unsigned int *index* )

For the attribute at the specified index, calculate the chi-squared and associated p-value.

Return as a pair.

**Parameters**

in	<i>index</i>	index into the attributes of the data set
----	--------------	---

**Returns**

pairs of chi-squared score and associated p-value for the attribute

Definition at line 46 of file ChiSquared.cpp.

**6.2.3.3** `const vector< pair< double, double > > & ChiSquared::ComputeScores ( )`

For each attribute, calculate chi-squared and associated p-value.

Return in a vector of pairs indexed by attribute index.

**Returns**

vector of pairs of chi-squared scores and associated p-values

Definition at line 34 of file ChiSquared.cpp.

**6.2.3.4** `std::vector<std::vector<double> > ChiSquared::GetFrequencyCounts ( )`  
[inline]

Get the observed frequencies table as a vector of vector of doubles.

Definition at line 62 of file ChiSquared.h.

**6.2.3.5** `void ChiSquared::PrepareForAttribute ( unsigned int attributeIndex )` [private]

Private method to setup the chi-squared contingency tables for a particular attribute.

**Parameters**

in	<i>attributeIndex</i>	attribute index
----	-----------------------	-----------------

Definition at line 209 of file ChiSquared.cpp.

**6.2.3.6** `void ChiSquared::PrintScores ( std::ofstream & outStream, unsigned int topN = 0 )`

Print the scores to a stream.

**Parameters**

in	<i>outStream</i>	reference to an output stream
in	<i>topN</i>	top number of attributes to print

Definition at line 176 of file ChiSquared.cpp.

### 6.2.3.7 void ChiSquared::PrintTables ( )

Print calculation tables.

Definition at line 145 of file ChiSquared.cpp.

### 6.2.3.8 void ChiSquared::WriteScores ( std::string *outFilename*, unsigned int *topN* = 0 )

Print the scores to a stream.

#### Parameters

in	<i>outFilename</i>	filename to write scores to
in	<i>topN</i>	top number of attributes to print

Definition at line 193 of file ChiSquared.cpp.

## 6.2.4 Member Data Documentation

### 6.2.4.1 std::vector<std::vector<double> > ChiSquared::chiSquaredValues [private]

chi squared computed values

Definition at line 86 of file ChiSquared.h.

### 6.2.4.2 Dataset\* ChiSquared::dataset [private]

pointer to a [Dataset](#) object

Definition at line 76 of file ChiSquared.h.

### 6.2.4.3 std::vector<std::vector<double> > ChiSquared::expectedContingencyTable [private]

Definition at line 84 of file ChiSquared.h.

### 6.2.4.4 unsigned int ChiSquared::numClasses [private]

number of classes in the instances

Definition at line 80 of file ChiSquared.h.

### 6.2.4.5 unsigned int ChiSquared::numLevels [private]

number of levels in the attributes

Definition at line 78 of file ChiSquared.h.



6.2.4.6 `std::vector<std::vector<double>>> ChiSquared::observedFreqTable`  
[private]

observed frequencies

Definition at line 82 of file ChiSquared.h.

6.2.4.7 `std::vector<std::pair<double, double>> ChiSquared::scores` [private]

chi-squared value, p-value for each attribute

Definition at line 88 of file ChiSquared.h.

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

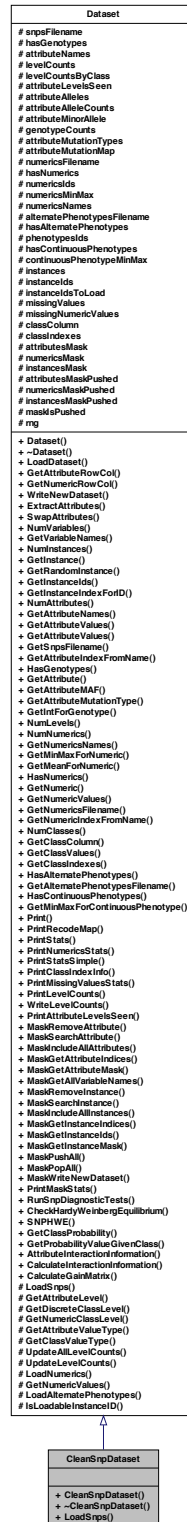
- [src/library/ChiSquared.h](#)
- [src/library/ChiSquared.cpp](#)

## 6.3 CleanSnpDataset Class Reference

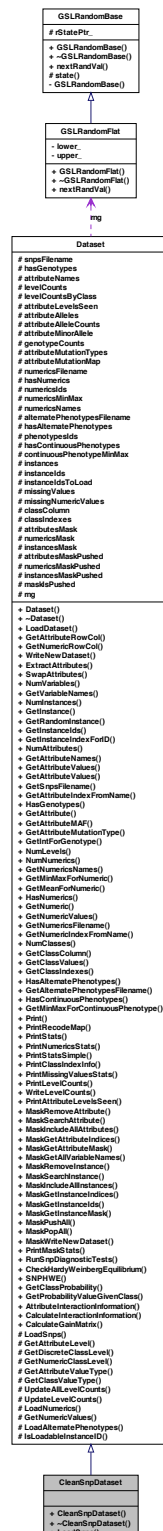
Experiemnetal data set reader for large/GWAS data.

```
#include <CleanSnpDataset.h>
```

Inheritance diagram for CleanSnpDataset:



Collaboration diagram for CleanSnpDataset:



## Public Member Functions

- [CleanSnpDataset](#) ()
- [~CleanSnpDataset](#) ()
- bool [LoadSnps](#) (std::string filename)

*Load SNPs from file using the data set filename.*

### 6.3.1 Detailed Description

Experiemnetal data set reader for large/GWAS data.

#### See also

[Dataset](#)

#### Author

Bill White

#### Version

1.0

Contact: [bill.c.white@gmail.com](mailto:bill.c.white@gmail.com) Created on: 9/22/11

Definition at line 20 of file CleanSnpDataset.h.

### 6.3.2 Constructor & Destructor Documentation

#### 6.3.2.1 CleanSnpDataset::CleanSnpDataset ( )

Definition at line 28 of file CleanSnpDataset.cpp.

#### 6.3.2.2 CleanSnpDataset::~~CleanSnpDataset ( ) `[inline]`

Definition at line 24 of file CleanSnpDataset.h.

### 6.3.3 Member Function Documentation

#### 6.3.3.1 bool CleanSnpDataset::LoadSnps ( std::string *filename* ) `[virtual]`

Load SNPs from file using the data set filename.

#### Parameters

in	<i>filename</i>	SNPs filename
in	<i>deRecodeA</i>	perform a recodeA operation after reading raw data?

**Returns**

SUCCESS

Reimplemented from [Dataset](#).

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

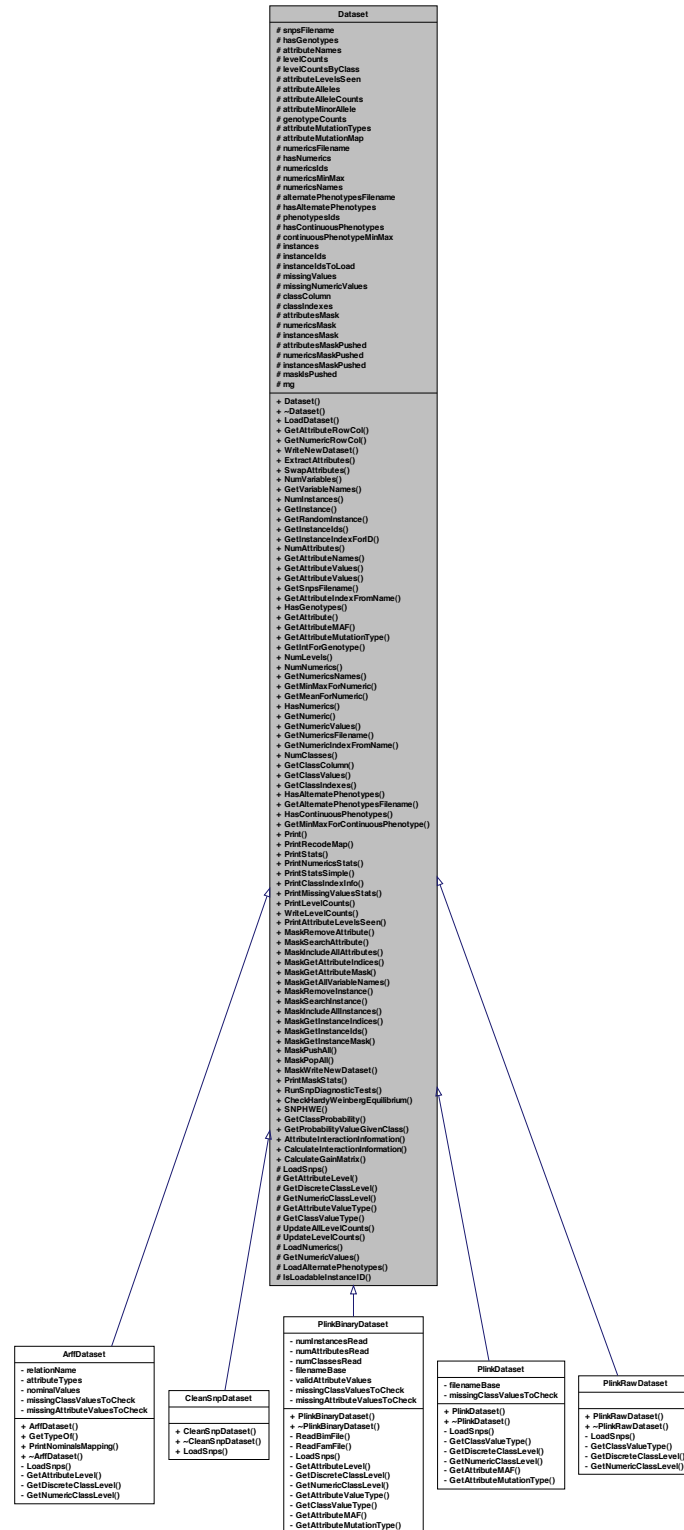
- src/library/[CleanSnpDataDataset.h](#)
- src/library/[CleanSnpDataDataset.cpp](#)

## 6.4 Dataset Class Reference

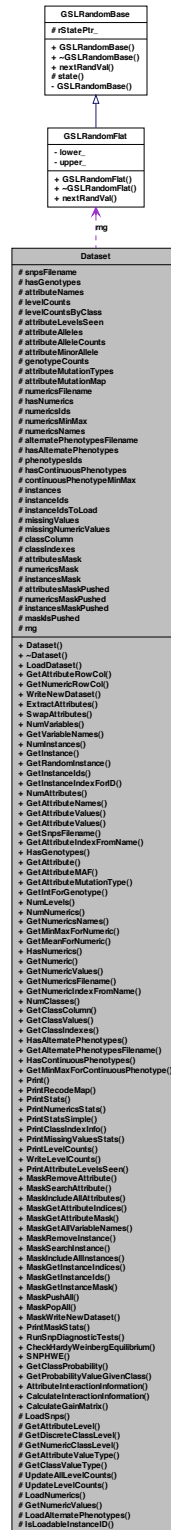
Base class for collections of instances containing attributea and class.

```
#include <Dataset.h>
```

Inheritance diagram for Dataset:



Collaboration diagram for Dataset:



## Public Member Functions

- [Dataset](#) ()  
*Construct a default data set.*
- [~Dataset](#) ()  
*Destruct all dynamically allocated memory.*
- bool [LoadDataset](#) (std::string snpsFilename, std::string numericsFilename, std::string altPhenoFilename, std::vector< std::string > ids)  
*Load the dataset from file set in the constructor.*
- bool [GetAttributeRowCol](#) (unsigned int row, unsigned int col, [AttributeLevel](#) &attrVal)  
*Get the attribute value at row, column.*
- bool [GetNumericRowCol](#) (unsigned int row, unsigned int col, [NumericLevel](#) &numVal)  
*Get the numeric value at row, column.*
- bool [WriteNewDataset](#) (std::string newDatasetFilename, [OutputDatasetType](#) outputDatasetType)  
*Write the dataset to a new filename, respecting masked attributes and numerics and class/phenotype data type.*
- bool [ExtractAttributes](#) (std::string scoresFilename, unsigned int topN, std::string newDatasetFilename)  
*Extracts top N attributes based on a file of attribute scores and writes a new dataset.*
- bool [SwapAttributes](#) (unsigned int a1, unsigned int a2)  
*Swap two attributes/columns in the dataset.*
- unsigned int [NumVariables](#) ()  
*Return the number of discrete plus continuous variables in the data set.*
- std::vector< std::string > [GetVariableNames](#) ()  
*Returns the names of discrete and continuous variables in the data set.*
- virtual unsigned int [NumInstances](#) ()  
*Returns the number of instances in the data set.*
- [DatasetInstance](#) \* [GetInstance](#) (unsigned int index)  
*Returns a pointer to a dataset instance selected by index.*
- [DatasetInstance](#) \* [GetRandomInstance](#) ()  
*Returns a pointer to a randomly chosen data set instance.*
- std::vector< std::string > [GetInstanceIds](#) ()  
*Get all instance IDs.*
- bool [GetInstanceIndexForID](#) (std::string ID, unsigned int &instanceIndex)  
*Get the instance index from the instance ID.*
- virtual unsigned int [NumAttributes](#) ()  
*Return the number of unmasked discrete attributes in the data set.*
- std::vector< std::string > [GetAttributeNames](#) ()  
*Return the discrete (SNP) attribute names.*
- bool [GetAttributeValues](#) (unsigned int attributeIndex, std::vector< [AttributeLevel](#) > &attributeValues)



- Loads the referenced vector with an attribute's values (column).*

  - bool [GetAttributeValues](#) (std::string attributeName, std::vector< [AttributeLevel](#) > &attributeValues)
- Loads the referenced vector with an attribute's values (column) from the dataset.*

  - std::string [GetSnpsFilename](#) ()

*Get the filename SNPs were read from.*
- unsigned int [GetAttributeIndexFromName](#) (std::string attributeName)

*Looks up original attribute index from attribute name.*
- bool [HasGenotypes](#) ()

*Does the data set have genotype variables?*
- [AttributeLevel](#) [GetAttribute](#) (unsigned instanceIndex, std::string name)

*Get attribute value for attribute name at instance index.*
- virtual std::pair< char, double > [GetAttributeMAF](#) (unsigned int attributeIndex)

*Get attribute minor allele and frequency.*
- virtual [AttributeMutationType](#) [GetAttributeMutationType](#) (unsigned int attributeIndex)

*Get attribute mutation type.*
- bool [GetIntForGenotype](#) (std::string genotype, [AttributeLevel](#) &newAttr)

*Get integer value for string genotype.*
- unsigned int [NumLevels](#) (unsigned int index)

*Returns the number of levels in a given attribute index.*
- unsigned int [NumNumerics](#) ()

*Return the number of unmasked discrete attributes in the data set.*
- std::vector< std::string > [GetNumericsNames](#) ()

*Return the numeric attribute names.*
- std::pair< double, double > [GetMinMaxForNumeric](#) (unsigned int numericIdx)

*Get the minimum and maximum values for a numeric at index.*
- double [GetMeanForNumeric](#) (unsigned int numericIdx)

*Get the mean/average of numeric at index.*
- bool [HasNumerics](#) ()

*Does the data set have numeric variables?*
- [NumericLevel](#) [GetNumeric](#) (unsigned int instanceIndex, std::string name)

*Get numeric value for numeric name at instance index.*
- bool [GetNumericValues](#) (std::string numericName, std::vector< [NumericLevel](#) > &numericValues)

*Loads the referenced vector with a numeric's values (column) from the dataset.*
- std::string [GetNumericsFilename](#) ()

*Get the filename numerics were read from.*
- unsigned int [GetNumericIndexFromName](#) (std::string numericName)

*Looks up original numeric index from numeric name.*
- unsigned int [NumClasses](#) ()

*Get the number of classes in the data set.*
- unsigned int [GetClassColumn](#) ()

- Get the class column as read from the file.*

  - bool [GetClassValues](#) (std::vector< [ClassLevel](#) > &classValues)

*Loads the referenced vector with the dataset's class labels.*

  - const std::map< [ClassLevel](#), std::vector< unsigned int > > & [GetClassIndexes](#) ()

*Get a map from class levels to a vector of instance indices.*

  - bool [HasAlternatePhenotypes](#) ()

*Does the data set have alternate phenotypes loaded?*

  - std::string [GetAlternatePhenotypesFilename](#) ()

*Get the alternate phenotype filename.*

  - bool [HasContinuousPhenotypes](#) ()
  - std::pair< double, double > [GetMinMaxForContinuousPhenotype](#) ()

*Get the mininum and maximum values for the continuous phenotype.*

  - void [Print](#) ()

*Print the entire data set in compact format.*

  - void [PrintRecodeMap](#) (std::vector< std::map< unsigned int, unsigned int > > recodeMap)

*Print the passed recode map to stdout.*

  - void [PrintStats](#) ()

*Print basic statistics about the data set - discrete/SNPs only.*

  - void [PrintNumericsStats](#) ()

*Print statistics about the data set including numerics.*

  - void [PrintStatsSimple](#) ()

*Print very simple statistics about the data set with no formatting.*

  - void [PrintClassIndexInfo](#) ()

*Print class index information.*

  - void [PrintMissingValuesStats](#) ()

*Print missing value statistics.*

  - void [PrintLevelCounts](#) ()

*Print attribute level counts.*

  - void [WriteLevelCounts](#) (std::string levelsFilename)

*Write attribute level counts to a text file.*

  - void [PrintAttributeLevelsSeen](#) ()

*Print unique attribute levels seen.*

  - bool [MaskRemoveAttribute](#) (std::string attributeName, [AttributeType](#) attrType)

*Removes the attribute name from consideration in any data set operations.*

  - bool [MaskSearchAttribute](#) (std::string attributeName, [AttributeType](#) attrType)

*Determines if the names attribute is in the current masked dataaset.*

  - bool [MaskIncludeAllAttributes](#) ([AttributeType](#) attrType)

*Mark all attributes for inclusion in data set operations.*

  - std::vector< unsigned int > [MaskGetAttributeIndices](#) ([AttributeType](#) attrType)

*Return a vector of all the attribute indices under consideration.*

  - const std::map< std::string, unsigned int > & [MaskGetAttributeMask](#) ([AttributeType](#) attrType)

- Return a map of attribute name to attribute index of attributes to include.*

  - `std::vector< std::string > MaskGetAllVariableNames ()`

*Return a vector of all the variable names under consideration.*

  - `bool MaskRemoveInstance (std::string instanceId)`

*Removes the instance from consideration in any data set operations.*

  - `bool MaskSearchInstance (std::string instanceId)`

*Determines if the names Instance is in the current masked dataset.*

  - `bool MaskIncludeAllInstances ()`

*Mark all instances for inclusion in algorithms.*

  - `std::vector< unsigned int > MaskGetInstanceIndices ()`

*Return a vector of all the instance indices under consideration.*

  - `std::vector< std::string > MaskGetInstanceIds ()`

*Return a vector of all the instance ids under consideration.*

  - `const std::map< std::string, unsigned int > & MaskGetInstanceMask ()`

*Return a map of instance name to instance index of instances to include.*

  - `bool MaskPushAll ()`

*Save the current masks for later restore.*

  - `bool MaskPopAll ()`

*Restore the masks previously pushed.*

  - `bool MaskWriteNewDataset (std::string newDatasetFilename)`

*Saved the unmasked attributes as a tab-delimited text file.*

  - `void PrintMaskStats ()`

*Print mask statistics.*

  - `void RunSnptestDiagnosticTests (std::string logFilename, double globalGenotype-Threshold=0.01, unsigned int cellThreshold=5)`

*Perform and report SNP diagnostic test information.*

  - `bool CheckHardyWeinbergEquilibrium (std::vector< unsigned int > genotype-Counts)`

*Calculate whether passed genotype counts are in HWE.*

  - `double SNPHWE (int obs_hets, int obs_hom1, int obs_hom2)`

*This code implements an exact SNP test of Hardy-Weinberg Equilibrium.*

  - `double GetClassProbability (ClassLevel thisClass)`

*Get the probability of a class value in the data set.*

  - `double GetProbabilityValueGivenClass (unsigned int attributeIndex, AttributeLevel A, ClassLevel classValue)`

*Get the probability of an attribute value at an attribute index.*

  - `void AttributeInteractionInformation ()`

*Calculate and display interaction information for all attribute combinations.*

  - `void CalculateInteractionInformation (std::map< std::pair< unsigned int, unsigned int >, std::map< std::string, double > > &results)`

*Calculate all the information needed to construct the interaction diagram.*

  - `bool CalculateGainMatrix (double **gainMatrix)`

*Calculate the GAIN matrix to run snprank on this data set.*

## Protected Member Functions

- virtual bool [LoadSnps](#) (std::string filename)  
*Load SNPs from file using the data set filename.*
- virtual bool [GetAttributeLevel](#) (std::string inLevel, std::vector< std::string > [missingValues](#), [AttributeLevel](#) &outLevel)  
*Get the attribute level based on string representation.*
- virtual bool [GetDiscreteClassLevel](#) (std::string inLevel, std::vector< std::string > [missingValues](#), [ClassLevel](#) &outLevel)  
*Get the discrete class level based on string representation.*
- virtual bool [GetNumericClassLevel](#) (std::string inLevel, std::vector< std::string > [missingValues](#), [NumericLevel](#) &outLevel)  
*Get the numeric class level based on string representation.*
- virtual [ValueType](#) [GetAttributeValueType](#) (std::string value, std::vector< std::string > [missingValues](#))  
*Get the passed attribute value's type.*
- virtual [ValueType](#) [GetClassValueType](#) (std::string value, std::vector< std::string > [missingValues](#))  
*Get the passed class value's type.*
- void [UpdateAllLevelCounts](#) ()  
*Update all attribute level counts from data set instances.*
- void [UpdateLevelCounts](#) ([DatasetInstance](#) \*dsi)  
*Update all attribute level counts from one data set instance.*
- bool [LoadNumerics](#) (std::string filename)  
*Load numerics (continuous attributes) from a file set in the constructor.*
- bool [GetNumericValues](#) (unsigned int numericIndex, std::vector< [NumericLevel](#) > &numericValues)  
*Loads the referenced vector with an numeric's values (column).*
- bool [LoadAlternatePhenotypes](#) (std::string filename)  
*Load alternate phenotype/class values from a plink covariate .cov file.*
- bool [IsLoadableInstanceID](#) (std::string ID)  
*Is the passed instance ID loadable (not filtered).*

## Protected Attributes

- std::string [snpsFilename](#)  
*file from which the discrete attributes (SNPSs) were read*
- bool [hasGenotypes](#)  
*does the data set contain any genotypes?*
- std::vector< std::string > [attributeNames](#)  
*discrete attribute names read from file*
- std::vector< std::map< [AttributeLevel](#), unsigned int > > [levelCounts](#)  
*attribute values/levels counts*

- `std::vector< std::map< std::pair< AttributeLevel, ClassLevel >, unsigned int > > levelCountsByClass`  
*attribute values/levels counts by discrete class*
- `std::vector< std::set< std::string > > attributeLevelsSeen`  
*unique attribute values/levels read from file*
- `std::vector< std::pair< char, char > > attributeAlleles`  
*allele1, allele2*
- `std::vector< std::map< char, unsigned int > > attributeAlleleCounts`  
*allele->count*
- `std::vector< std::pair< char, double > > attributeMinorAllele`  
*minor allele, minor allele frequency*
- `std::vector< std::map< std::string, unsigned int > > genotypeCounts`  
*genotype->count*
- `std::vector< AttributeMutationType > attributeMutationTypes`  
*Keep mutation type for all attributes.*
- `std::map< std::pair< char, char >, AttributeMutationType > attributeMutation-Map`  
*Lookup table for mutation type.*
- `std::string numericsFilename`  
*file from which the continuous attributes were read*
- `bool hasNumerics`  
*does the data set contain any continuous attributes?*
- `std::vector< std::string > numericsIds`  
*IDs associated with the numerics read from file.*
- `std::vector< std::pair< NumericLevel, NumericLevel > > numericsMinMax`  
*the minimum and maximum value for each continuous attribute*
- `std::vector< std::string > numericsNames`  
*continuous attribute names read from file*
- `std::string alternatePhenotypesFilename`  
*file from which the alternate phenotypes (class labels) were read*
- `bool hasAlternatePhenotypes`  
*does the data set contain alternate phenotypes?*
- `std::vector< std::string > phenotypesIds`  
*IDs associated with the phenotypes/classes read from file.*
- `bool hasContinuousPhenotypes`  
*does the data set contain continuous phenotypes?*
- `std::pair< NumericLevel, NumericLevel > continuousPhenotypeMinMax`  
*the minimum and maximum value for each continuous phenotype*
- `std::vector< DatasetInstance * > instances`  
*vector of pointers to all instances in the data set*
- `std::vector< std::string > instanceIds`  
*IDs associated with the instances read from file.*
- `std::vector< std::string > instanceIdsToLoad`

*IDs of instances to load from numeric and/or phenotype files.*

- `std::map< std::string, std::vector< unsigned int > >` [missingValues](#)  
*missing discrete values and their instance indices*
- `std::map< std::string, std::vector< unsigned int > >` [missingNumericValues](#)  
*missing continuous values and their instance indices*
- `unsigned int` [classColumn](#)  
*class column from the original data set*
- `std::map< ClassLevel, std::vector< unsigned int > >` [classIndexes](#)  
*class values mapped to instance indices*
- `std::map< std::string, unsigned int >` [attributesMask](#)
- `std::map< std::string, unsigned int >` [numericsMask](#)
- `std::map< std::string, unsigned int >` [instancesMask](#)
- `std::map< std::string, unsigned int >` [attributesMaskPushed](#)  
*masks can be temporarily pushed and popped*
- `std::map< std::string, unsigned int >` [numericsMaskPushed](#)
- `std::map< std::string, unsigned int >` [instancesMaskPushed](#)
- `bool` [maskIsPushed](#)
- `GSLRandomFlat * rng`  
*random number generator classes use GNU Scientific Library (GSL)*

### 6.4.1 Detailed Description

Base class for collections of instances containing attributea and class.

Added interaction information week of 4/18-26/06 Totally redone for McKinney Lab. February 2011.

#### Author

Bill White

#### Version

1.0

Contact: [bill.c.white@gmail.com](mailto:bill.c.white@gmail.com) Created on: 6/14/05

Definition at line 101 of file Dataset.h.

### 6.4.2 Constructor & Destructor Documentation

#### 6.4.2.1 `Dataset::Dataset ( )`

Construct a default data set.

Set private data to defaults.

Load attribute mutation map for transitions/transversions.

Definition at line 45 of file Dataset.cpp.

**6.4.2.2 Dataset::~~Dataset ( )**

Destruct all dynamically allocated memory.

Definition at line 77 of file Dataset.cpp.

**6.4.3 Member Function Documentation****6.4.3.1 void Dataset::AttributeInteractionInformation ( )**

Calculate and display interaction information for all attribute combinations.

get the column sum

display results detail; selected column as percentages

Definition at line 1498 of file Dataset.cpp.

**6.4.3.2 bool Dataset::CalculateGainMatrix ( double \*\* *gainMatrix* )**

Calculate the GAIN matrix to run snprank on this data set.

Uses OpenMP to calculate matrix entries in parallel threads.

**Parameters**

out	<i>gainMatrix</i>	pointer to an allocated n x n matrix, n = number of attributes
-----	-------------------	--

**Returns**

!success

for all possible (unique) interactions, ie nCk

compute gainMatrix[i][j]

Definition at line 1618 of file Dataset.cpp.

**6.4.3.3 void Dataset::CalculateInteractionInformation ( std::map< std::pair< unsigned int, unsigned int >, std::map< std::string, double > > & *results* )**

Calculate all the information needed to construct the interaction diagram.

**Parameters**

out	<i>results</i>	map of attribute combinations to results
-----	----------------	--

Definition at line 1551 of file Dataset.cpp.

#### 6.4.3.4 `bool Dataset::CheckHardyWeinbergEquilibrium ( std::vector< unsigned int > genotypeCounts )`

Calculate whether passed genotype counts are in HWE.

##### Parameters

<i>genotype-Counts</i>	vector of genotype counts: AA, Aa, aa
------------------------	---------------------------------------

##### Returns

counts are in HWE?

observed counts

HWE probabilities

expected values

perform Pearson's chi-squared test

one degree of freedom (# genotypes - # alleles), 5% significance level

Definition at line 1320 of file Dataset.cpp.

#### 6.4.3.5 `bool Dataset::ExtractAttributes ( std::string scoresFilename, unsigned int topN, std::string newDatasetFilename )`

Extracts top N attributes based on a file of attribute scores and writes a new dataset.

Revised 10/3/11 for numerics and continuous class/phenotypes.

##### Parameters

in	<i>scoresFilename</i>	filename of attribute scores and names
in	<i>topN</i>	top N attributes
in	<i>newDatasetFilename</i>	filename of new dataset

##### Returns

success

Definition at line 354 of file Dataset.cpp.

#### 6.4.3.6 `string Dataset::GetAlternatePhenotypesFilename ( )`

Get the alternate phenotype filename.

Definition at line 721 of file Dataset.cpp.



**6.4.3.7 AttributeLevel Dataset::GetAttribute ( unsigned *instanceIndex*, std::string *name* )**

Get attribute value for attribute name at instance index.

**Parameters**

in	<i>instanceIndex</i>	instance index
in	<i>name</i>	attribute name

**Returns**

attributevalue

**6.4.3.8 unsigned int Dataset::GetAttributeIndexFromName ( std::string *attributeName* )**

Looks up original attribute index from attribute name.

**Parameters**

in	<i>attributeName</i>	attribute name
----	----------------------	----------------

**Returns**

attribute index or INVALID\_INDEX

Definition at line 614 of file Dataset.cpp.

**6.4.3.9 virtual bool Dataset::GetAttributeLevel ( std::string *inLevel*, std::vector< std::string > *missingValues*, AttributeLevel & *outLevel* ) [protected, virtual]**

Get the attribute level based on string representation.

**Parameters**

in	<i>inLevel</i>	attribute level read from file
in	<i>missingValues</i>	list of strings representing missing attribute values
out	<i>outLevel</i>	attribute level to use in the data set class

**Returns**

success

Reimplemented in [ArffDataset](#), and [PlinkBinaryDataset](#).

**6.4.3.10** `pair< char, double > Dataset::GetAttributeMAF ( unsigned int attributeIndex )`  
`[virtual]`

Get attribute minor allele and frequency.

#### Parameters

<code>in</code>	<code><i>attribute</i></code>	index
-----------------	-------------------------------	-------

#### Returns

pair (minor allele, minor allele frequency)

An Intriduction to Genetic Analysis by Griffiths, Miller, Suzuki, Lewontin and Gelbart, 2000, page 715.

Reimplemented in [PlinkBinaryDataset](#), and [PlinkDataset](#).

Definition at line 575 of file Dataset.cpp.

**6.4.3.11** `AttributeMutationType Dataset::GetAttributeMutationType ( unsigned int attributeIndex )`  
`[virtual]`

Get attribute mutation type.

#### Parameters

<code>in</code>	<code><i>attribute</i></code>	index
-----------------	-------------------------------	-------

#### Returns

mutation type (transition, transversion, unknown)

Reimplemented in [PlinkBinaryDataset](#), and [PlinkDataset](#).

Definition at line 600 of file Dataset.cpp.

**6.4.3.12** `vector< string > Dataset::GetAttributeNames ( )`

Return the discrete (SNP) attribute names.

#### Returns

vector of attribute names

Definition at line 506 of file Dataset.cpp.

**6.4.3.13** `bool Dataset::GetAttributeRowCol ( unsigned int row, unsigned int col, AttributeLevel & attrVal )`

Get the attribute value at row, column.

Same as instance index, attribute index.

#### Parameters

in	<i>row</i>	instance row
in	<i>col</i>	attribute column
out	<i>attrVal</i>	attribute value

#### Returns

success

Definition at line 166 of file Dataset.cpp.

**6.4.3.14** `bool Dataset::GetAttributeValues ( std::string attributeName, std::vector< AttributeLevel > & attributeValues )`

Loads the referenced vector with an attribute's values (column) from the dataset.

#### Parameters

in	<i>attribute-Name</i>	attribute name
out	<i>attributeValues</i>	reference to a a vector allocated by the caller

#### Returns

SUCCESS

**6.4.3.15** `bool Dataset::GetAttributeValues ( unsigned int attributeIndex, std::vector< AttributeLevel > & attributeValues )`

Loads the referenced vector with an attribute's values (column).

from the dataset

#### Parameters

in	<i>attributeIndex</i>	attribute index
out	<i>attributeValues</i>	reference to a a vector allocated by the caller

#### Returns

SUCCESS

6.4.3.16 `virtual ValueType Dataset::GetAttributeValueType ( std::string value, std::vector< std::string > missingValues )` [protected, virtual]

Get the passed attribute value's type.

#### Parameters

in	<i>value</i>	value to check
in	<i>missingValues</i>	vector of possible missing values

#### Returns

value's type

Reimplemented in [PlinkBinaryDataset](#).

6.4.3.17 `unsigned int Dataset::GetClassColumn ( )`

Get the class column as read from the file.

Definition at line 699 of file Dataset.cpp.

6.4.3.18 `const std::map< ClassLevel, std::vector< unsigned int > > & Dataset::GetClassIndexes ( )`

Get a map from class levels to a vector of instance indices.

#### Returns

map of class => instance indices

Definition at line 713 of file Dataset.cpp.

6.4.3.19 `double Dataset::GetClassProbability ( ClassLevel thisClass )`

Get the probability of a class value in the data set.

#### Parameters

<i>thisClass</i>	class value
------------------	-------------

#### Returns

probability

Definition at line 1475 of file Dataset.cpp.

**6.4.3.20** `bool Dataset::GetClassValues ( std::vector< ClassLevel > & classValues )`

Loads the referenced vector with the dataset's class labels.

**Parameters**

out	<i>classValues</i>	reference to a a vector allocated by the caller
-----	--------------------	---

**Returns**

success

Definition at line 703 of file Dataset.cpp.

**6.4.3.21** `virtual ValueType Dataset::GetClassValueType ( std::string value, std::vector< std::string > missingValues )` [protected, virtual]

Get the passed class value's type.

**Parameters**

in	<i>value</i>	value to check
in	<i>missingValues</i>	vector of possible missing values

**Returns**

value's type

Reimplemented in [PlinkBinaryDataset](#), [PlinkDataset](#), and [PlinkRawDataset](#).

**6.4.3.22** `virtual bool Dataset::GetDiscreteClassLevel ( std::string inLevel, std::vector< std::string > missingValues, ClassLevel & outLevel )` [protected, virtual]

Get the discrete class level based on string representation.

**Parameters**

in	<i>inLevel</i>	class level read from file
in	<i>missingValues</i>	list of strings representing missing class values
out	<i>outLevel</i>	discrete class level to use in the data set class

**Returns**

success

Reimplemented in [ArffDataset](#), [PlinkBinaryDataset](#), [PlinkDataset](#), and [PlinkRawDataset](#).

#### 6.4.3.23 `DatasetInstance * Dataset::GetInstance ( unsigned int index )`

Returns a pointer to a dataset instance selected by index.

##### Parameters

<i>in</i>	<i>index</i>	index of instance
-----------	--------------	-------------------

##### Returns

pointer to an instance

Definition at line 468 of file Dataset.cpp.

#### 6.4.3.24 `vector< string > Dataset::GetInstanceIds ( )`

Get all instance IDs.

##### Returns

vector of instance IDs

Definition at line 481 of file Dataset.cpp.

#### 6.4.3.25 `bool Dataset::GetInstanceIndexForID ( std::string ID, unsigned int & instanceIndex )`

Get the instance index from the instance ID.

##### Parameters

<i>in</i>	<i>ID</i>	string ID
<i>out</i>	<i>instanceIndex</i>	instance index

##### Returns

success

Definition at line 490 of file Dataset.cpp.

#### 6.4.3.26 `bool Dataset::GetIntForGenotype ( std::string genotype, AttributeLevel & newAttr )`

Get integer value for string genotype.

##### Parameters

<i>in</i>	<i>genotype</i>	genotype string
<i>out</i>	<i>newAttr</i>	new attribute value

**Returns**

success

**6.4.3.27** `double Dataset::GetMeanForNumeric ( unsigned int numericIdx )`

Get the mean/average of numeric at index.

**Parameters**

<code>in</code>	<code><i>numericIdx</i></code>	numeric index
-----------------	--------------------------------	---------------

**Returns**

average value of numeric attribute at index

Definition at line 641 of file Dataset.cpp.

**6.4.3.28** `pair< double, double > Dataset::GetMinMaxForContinuousPhenotype ( )`

Get the minimum and maximum values for the continuous phenotype.

**Returns**

minimum/maximum pair

Definition at line 729 of file Dataset.cpp.

**6.4.3.29** `pair< NumericLevel, NumericLevel > Dataset::GetMinMaxForNumeric ( unsigned int numericIdx )`

Get the minimum and maximum values for a numeric at index.

**Parameters**

<code>in</code>	<code><i>numericIdx</i></code>	numeric index
-----------------	--------------------------------	---------------

**Returns**

minimum/maximum pair

Definition at line 637 of file Dataset.cpp.

**6.4.3.30** `NumericLevel Dataset::GetNumeric ( unsigned int instanceIndex, std::string name )`

Get numeric value for numeric name at instance index.

**Parameters**

in	<i>instanceIndex</i>	instance index
in	<i>name</i>	numeric name

**Returns**

numeric value at index

```
6.4.3.31 virtual bool Dataset::GetNumericClassLevel ( std::string inLevel, std::vector<
std::string > missingValues, NumericLevel & outLevel ) [protected,
virtual]
```

Get the numeric class level based on string representation.

**Parameters**

in	<i>inLevel</i>	class level read from file
in	<i>missingValues</i>	list of strings representing missing class values
in, out	<i>outLevel</i>	numeric class level to use in the data set class

**Returns**

success

Reimplemented in [ArffDataset](#), [PlinkBinaryDataset](#), [PlinkDataset](#), and [PlinkRawDataset](#).

```
6.4.3.32 unsigned int Dataset::GetNumericIndexFromName ( std::string numericName )
```

Looks up original numeric index from numeric name.

**Parameters**

in	<i>numericName</i>	numeric name
----	--------------------	--------------

**Returns**

attribute index or INVALID\_INDEX

Definition at line 686 of file Dataset.cpp.

```
6.4.3.33 bool Dataset::GetNumericRowCol ( unsigned int row, unsigned int col,
NumericLevel & numVal )
```

Get the numeric value at row, column.

Same as instance index, numeric index.



**Parameters**

in	<i>row</i>	instance row
in	<i>col</i>	numeric column
out	<i>numVal</i>	numeric value

**Returns**

success

Definition at line 179 of file Dataset.cpp.

**6.4.3.34** `std::string Dataset::GetNumericsFilename ( )`

Get the filename numerics were read from.

Definition at line 682 of file Dataset.cpp.

**6.4.3.35** `vector< string > Dataset::GetNumericsNames ( )`

Return the numeric attribute names.

**Returns**

vector of attribute names

Definition at line 627 of file Dataset.cpp.

**6.4.3.36** `bool Dataset::GetNumericValues ( std::string numericName, std::vector< NumericLevel > & numericValues )`

Loads the referenced vector with a numeric's values (column) from the dataset.

**Parameters**

in	<i>numeric-Name</i>	numeric name
out	<i>numericValues</i>	reference to a a vector allocated by the caller

**Returns**

success

**6.4.3.37** `bool Dataset::GetNumericValues ( unsigned int numericIndex, std::vector< NumericLevel > & numericValues ) [protected]`

Loads the referenced vector with an numeric's values (column).

from the dataset

**Parameters**

in	<i>numericIndex</i>	numeric index
out	<i>numericValues</i>	reference to a a vector allocated by the caller

**Returns**

success

**6.4.3.38** `double Dataset::GetProbabilityValueGivenClass ( unsigned int attributeIndex,  
AttributeLevel A, ClassLevel classValue )`

Get the probability of an attribute value at an attribute index.

**Parameters**

in	<i>attributeIndex</i>	attribute index
in	<i>A</i>	attribute value
in	<i>classValue</i>	class value

**Returns**

probability of the value in attribute given class

Definition at line 1482 of file Dataset.cpp.

**6.4.3.39** `DatasetInstance * Dataset::GetRandomInstance ( )`

Returns a pointer to a randomly chosen data set instance.

The random number generator is set to give values in range of instance indexes.

**Returns**

pointer to a data set instance

Definition at line 476 of file Dataset.cpp.

**6.4.3.40** `std::string Dataset::GetSnpsFilename ( )`

Get the filename SNPs were read from.

Definition at line 551 of file Dataset.cpp.

**6.4.3.41** `vector< string > Dataset::GetVariableNames ( )`

Returns the names of discrete and continuous variables in the data set.

**Returns**

vector of names as strings

Definition at line 452 of file Dataset.cpp.

**6.4.3.42 bool Dataset::HasAlternatePhenotypes ( )**

Does the data set have alternate phenotypes loaded?

Definition at line 717 of file Dataset.cpp.

**6.4.3.43 bool Dataset::HasContinuousPhenotypes ( )**

Definition at line 725 of file Dataset.cpp.

**6.4.3.44 bool Dataset::HasGenotypes ( )**

Does the data set have genotype variables?

Definition at line 555 of file Dataset.cpp.

**6.4.3.45 bool Dataset::HasNumerics ( )**

Does the data set have numeric variables?

Definition at line 651 of file Dataset.cpp.

**6.4.3.46 bool Dataset::IsLoadableInstanceID ( std::string *ID* ) [protected]**

Is the passed instance ID loadable (not filtered).

**Parameters**

<i>in</i>	<i>ID</i>	instance ID
-----------	-----------	-------------

**Returns**

[out] success

Definition at line 2423 of file Dataset.cpp.

**6.4.3.47 bool Dataset::LoadAlternatePhenotypes ( std::string *filename* ) [protected]**

Load alternate phenotype/class values from a plink covariate .cov file.

Format described here: <http://pngu.mgh.harvard.edu/~purcell/plink/data.shtml#covar>  
MAJOR CHANGES: for continuous phenotypes/class - 9/29/11

**Parameters**

in	<i>filename</i>	alternate phenotype data filename in PLINK covar format
----	-----------------	---

**Returns**

success

Definition at line 2262 of file Dataset.cpp.

**6.4.3.48** `bool Dataset::LoadDataset ( std::string snpFilename, std::string numericsFilename, std::string altPhenoFilename, std::vector< std::string > ids )`

Load the dataset from file set in the constructor.

This is a virtual function overridden by subclasses.

**Parameters**

in	<i>snpFilename</i>	discrete values (SNPs) filename
in	<i>doRecodeA</i>	perform recodeA encoding after reading
in	<i>numerics-Filename</i>	continuous values (numerics) filename or empty string
in	<i>altPhenoFilename</i>	alternate class (phenotype) filename or empty string
in	<i>ids</i>	vector of possibly empty IDs to match in auxiliary files

**Returns**

success

Definition at line 89 of file Dataset.cpp.

**6.4.3.49** `bool Dataset::LoadNumerics ( std::string filename )` `[protected]`

Load numerics (continuous attributes) from a file set in the constructor.

**Parameters**

in	<i>filename</i>	numerics data filename in PLINK covar format
----	-----------------	--

**Returns**

success

Definition at line 2099 of file Dataset.cpp.

**6.4.3.50** `bool Dataset::LoadSnps ( std::string filename )` `[protected, virtual]`

Load SNPs from file using the data set filename.

**Parameters**

in	<i>filename</i>	SNPs filename
in	<i>deRecodeA</i>	perform a recodeA operation after reading raw data?

**Returns**

success

Reimplemented in [ArffDataset](#), [CleanSnpDataset](#), [PlinkBinaryDataset](#), [PlinkDataset](#), and [PlinkRawDataset](#).

Definition at line 1729 of file Dataset.cpp.

6.4.3.51 `vector< string > Dataset::MaskGetAllVariableNames ( )`

Return a vector of all the variable names under consideration.

**Returns**

vector of discrete and numeric variable

Definition at line 1043 of file Dataset.cpp.

6.4.3.52 `vector< unsigned int > Dataset::MaskGetAttributeIndices ( AttributeType attrType )`

Return a vector of all the attribute indices under consideration.

**Parameters**

<i>attrType</i>	attribute type
-----------------	----------------

**Returns**

vector of indices into currently considered discrete attributes

Definition at line 1018 of file Dataset.cpp.

6.4.3.53 `const map< string, unsigned int > & Dataset::MaskGetAttributeMask ( AttributeType attrType )`

Return a map of attribute name to attribute index of attributes to include.

**Parameters**

in	<i>attrType</i>	attribute type
----	-----------------	----------------

**Returns**

attributes mask: name->index

Definition at line 1035 of file Dataset.cpp.

#### 6.4.3.54 `vector< string > Dataset::MaskGetInstancelds ( )`

Return a vector of all the instance ids under consideration.

##### Returns

vector of ids of currently included instances

Definition at line 1100 of file Dataset.cpp.

#### 6.4.3.55 `vector< unsigned int > Dataset::MaskGetInstanceIndices ( )`

Return a vector of all the instance indices under consideration.

vector of indices into current instances

Definition at line 1091 of file Dataset.cpp.

#### 6.4.3.56 `const map< string, unsigned int > & Dataset::MaskGetInstanceMask ( )`

Return a map of instance name to instance index of instances to include.

##### Returns

instances mask: instance ID=>vector of instance indices

Definition at line 1109 of file Dataset.cpp.

#### 6.4.3.57 `bool Dataset::MaskIncludeAllAttributes ( AttributeType attrType )`

Mark all attributes for inclusion in data set operations.

##### Parameters

<i>in</i>	<i>attrType</i>	attribute type
-----------	-----------------	----------------

##### Returns

success

Definition at line 996 of file Dataset.cpp.

#### 6.4.3.58 `bool Dataset::MaskIncludeAllInstances ( )`

Mark all instances for inclusion in algorithms.

**Returns**

success

Definition at line 1079 of file Dataset.cpp.

**6.4.3.59 bool Dataset::MaskPopAll ( )**

Restore the masks previously pushed.

**Returns**

success

Definition at line 1127 of file Dataset.cpp.

**6.4.3.60 bool Dataset::MaskPushAll ( )**

Save the current masks for later restore.

**Returns**

success

Definition at line 1113 of file Dataset.cpp.

**6.4.3.61 bool Dataset::MaskRemoveAttribute ( std::string *attributeName*, AttributeType *attrType* )**

Removes the attribute name from consideration in any data set operations.

**Parameters**

in	<i>attribute-Name</i>	attribute name
in	<i>attrType</i>	attribute type

**Returns**

success

Definition at line 949 of file Dataset.cpp.

**6.4.3.62 bool Dataset::MaskRemoveInstance ( std::string *instanceId* )**

Removes the instance from consideration in any data set operations.

**Parameters**

in	<i>instanceId</i>	instance ID
----	-------------------	-------------

**Returns**

success

Definition at line 1056 of file Dataset.cpp.

**6.4.3.63 bool Dataset::MaskSearchAttribute ( std::string *attributeName*, AttributeType *attrType* )**

Determines if the names attribute is in the current masked dataaset.

**Parameters**

in	<i>attribute-Name</i>	attribute name
in	<i>attributeType</i>	attribute type

**Returns**

true if discrete attribute name is being considered in operations.

Definition at line 976 of file Dataset.cpp.

**6.4.3.64 bool Dataset::MaskSearchInstance ( std::string *instanceId* )**

Determines if the names Instance is in the current masked dataaset.

**Parameters**

in	<i>instanceID</i>	instance ID
----	-------------------	-------------

**Returns**

true if instance ID is in the dataset, considering instance mask

Definition at line 1069 of file Dataset.cpp.

**6.4.3.65 bool Dataset::MaskWriteNewDataset ( std::string *newDatasetFilename* )**

Saved the unmasked attributes as a tab-delimited text file.

**Parameters**

in	<i>newDataset-Filename</i>	new data set filename
----	----------------------------	-----------------------



**Returns**

success

Definition at line 1140 of file Dataset.cpp.

**6.4.3.66** `unsigned int Dataset::NumAttributes ( ) [virtual]`

Return the number of unmasked discrete attributes in the data set.

Definition at line 502 of file Dataset.cpp.

**6.4.3.67** `unsigned int Dataset::NumClasses ( )`

Get the number of classes in the data set.

Definition at line 695 of file Dataset.cpp.

**6.4.3.68** `unsigned int Dataset::NumInstances ( ) [virtual]`

Returns the number of instances in the data set.

Definition at line 464 of file Dataset.cpp.

**6.4.3.69** `unsigned int Dataset::NumLevels ( unsigned int index )`

Returns the number of levels in a given attribute index.

**Parameters**

<i>in</i>	<i>index</i>	attribute index
-----------	--------------	-----------------

**Returns**

number of levels

Definition at line 604 of file Dataset.cpp.

**6.4.3.70** `unsigned int Dataset::NumNumerics ( )`

Return the number of unmasked discrete attributes in the data set.

Definition at line 623 of file Dataset.cpp.

**6.4.3.71** `unsigned int Dataset::NumVariables ( )`

Return the number of discrete plus continuous variables in the data set.

The number does not include masked variables removed.

**Returns**

number of discrete plus continuous variables

Definition at line 448 of file Dataset.cpp.

**6.4.3.72 void Dataset::Print ( )**

Print the entire data set in compact format.

Definition at line 733 of file Dataset.cpp.

**6.4.3.73 void Dataset::PrintAttributeLevelsSeen ( )**

Print unique attribute levels seen.

Definition at line 935 of file Dataset.cpp.

**6.4.3.74 void Dataset::PrintClassIndexInfo ( )**

Print class index information.

Definition at line 841 of file Dataset.cpp.

**6.4.3.75 void Dataset::PrintLevelCounts ( )**

Print attribute level counts.

Definition at line 878 of file Dataset.cpp.

**6.4.3.76 void Dataset::PrintMaskStats ( )**

Print mask statistics.

Definition at line 1186 of file Dataset.cpp.

**6.4.3.77 void Dataset::PrintMissingValuesStats ( )**

Print missing value statistics.

Definition at line 850 of file Dataset.cpp.

**6.4.3.78 void Dataset::PrintNumericsStats ( )**

Print statistics about the data set including numerics.

Definition at line 775 of file Dataset.cpp.

**6.4.3.79** `void Dataset::PrintRecodeMap ( std::vector< std::map< unsigned int, unsigned int >  
> recodeMap )`

Print the passed recode map to stdout.

#### See also

DoRecodeA()

#### Parameters

in	<i>recodeMap</i>	recoding map
----	------------------	--------------

Definition at line 921 of file Dataset.cpp.

**6.4.3.80** `void Dataset::PrintStats ( )`

Print basic statistics about the data set - discrete/SNPs only.

Definition at line 741 of file Dataset.cpp.

**6.4.3.81** `void Dataset::PrintStatsSimple ( )`

Print very simple statistics about the data set with no formatting.

Definition at line 812 of file Dataset.cpp.

**6.4.3.82** `void Dataset::RunSnpDiagnosticTests ( std::string logFilename, double  
globalGenotypeThreshold = 0.01, unsigned int cellThreshold = 5 )`

Perform and report SNP diagnostic test information.

#### Parameters

in	<i>logFilename</i>	log filename
in	<i>globalGeno- typeThresh- old</i>	genotype count threshold
in	<i>cellThresh- old</i>	$\chi^2$ cell count threshold

Definition at line 1195 of file Dataset.cpp.

**6.4.3.83** `double Dataset::SNPWE ( int obs_hets, int obs_hom1, int obs_hom2 )`

This code implements an exact SNP test of Hardy-Weinberg Equilibrium.

As described in Wigginton, JE, Cutler, DJ, and Abecasis, GR (2005) A Note on Exact Tests of Hardy-Weinberg Equilibrium. American Journal of Human Genetics: 76.

Written by Jan Wigginton.

#### Parameters

in	<i>obs_hets</i>	observed heterozygotes
in	<i>obs_hom1</i>	observed homozygotes type 1
in	<i>obs_hom2</i>	homozygotes type 2

#### Returns

HWE value

Definition at line 1382 of file Dataset.cpp.

#### 6.4.3.84 bool Dataset::SwapAttributes ( unsigned int *a1*, unsigned int *a2* )

Swap two attributes/columns in the dataset.

#### Parameters

in	<i>a1</i>	attribue index 1
in	<i>a2</i>	attribue index 2

#### Returns

success

Definition at line 439 of file Dataset.cpp.

#### 6.4.3.85 void Dataset::UpdateAllLevelCounts ( ) [protected]

Update all attribute level counts from data set instances.

Updates levelCounts, levelCountsByClass.

Definition at line 2060 of file Dataset.cpp.

#### 6.4.3.86 void Dataset::UpdateLevelCounts ( DatasetInstance \* *dsi* ) [protected]

Update all attribute level counts from one data set instance.

Updates levelCountsByClass.

#### Parameters

in	<i>dsi</i>	pointer to a data set instance
----	------------	--------------------------------

Definition at line 2084 of file Dataset.cpp.

**6.4.3.87** void Dataset::WriteLevelCounts ( std::string *levelsFilename* )

Write attribute level counts to a text file.

**Parameters**

in	<i>levelsFilename</i>	filename to write levels to
----	-----------------------	-----------------------------

Definition at line 896 of file Dataset.cpp.

**6.4.3.88** bool Dataset::WriteNewDataset ( std::string *newDatasetFilename*,  
OutputDatasetType *outputDatasetType* )

Write the dataset to a new filename, respecting masked attributes and numerics and class/phenotype data type.

**Parameters**

in	<i>newDatasetFilename</i>	new dataset filename
----	---------------------------	----------------------

**Returns**

success

write the attribute names header

write the data, respecting the masked attributes, numerics and masked instances -  
10/28/11 write the attribute names header

write continuous attribute values

Definition at line 192 of file Dataset.cpp.

**6.4.4 Member Data Documentation****6.4.4.1** std::string Dataset::alternatePhenotypesFilename [protected]

file from which the alternate phenotypes (class labels) were read

Definition at line 655 of file Dataset.h.

**6.4.4.2** std::vector<std::map<char, unsigned int> > Dataset::attributeAlleleCounts  
[protected]

allele->count

Definition at line 633 of file Dataset.h.

**6.4.4.3** `std::vector<std::pair<char, char> > Dataset::attributeAlleles`  
[protected]

allele1, allele2

Definition at line 631 of file Dataset.h.

**6.4.4.4** `std::vector<std::set<std::string> > Dataset::attributeLevelsSeen`  
[protected]

unique attribute values/levels read from file

Definition at line 629 of file Dataset.h.

**6.4.4.5** `std::vector<std::pair<char, double> > Dataset::attributeMinorAllele`  
[protected]

minor allele, minor allele frequency

Definition at line 635 of file Dataset.h.

**6.4.4.6** `std::map<std::pair<char, char>, AttributeMutationType>`  
`Dataset::attributeMutationMap` [protected]

Lookup table for mutation type.

Definition at line 641 of file Dataset.h.

**6.4.4.7** `std::vector<AttributeMutationType> Dataset::attributeMutationTypes`  
[protected]

Keep mutation type for all attributes.

Definition at line 639 of file Dataset.h.

**6.4.4.8** `std::vector<std::string> Dataset::attributeNames` [protected]

discrete attribute names read from file

Definition at line 623 of file Dataset.h.

**6.4.4.9** `std::map<std::string, unsigned int> Dataset::attributesMask` [protected]

Definition at line 686 of file Dataset.h.

**6.4.4.10** `std::map<std::string, unsigned int> Dataset::attributesMaskPushed`  
[protected]

masks can be temporarily pushed and popped

Definition at line 690 of file Dataset.h.

**6.4.4.11** `unsigned int Dataset::classColumn` [protected]

class column from the original data set

Definition at line 677 of file Dataset.h.

**6.4.4.12** `std::map<ClassLevel, std::vector<unsigned int> > Dataset::classIndexes`  
[protected]

class values mapped to instance indices

Definition at line 679 of file Dataset.h.

**6.4.4.13** `std::pair<NumericLevel, NumericLevel>`  
`Dataset::continuousPhenotypeMinMax` [protected]

the minimum and maximum value for each continuous phenotype

Definition at line 663 of file Dataset.h.

**6.4.4.14** `std::vector<std::map<std::string, unsigned int> > Dataset::genotypeCounts`  
[protected]

genotype->count

Definition at line 637 of file Dataset.h.

**6.4.4.15** `bool Dataset::hasAlternatePhenotypes` [protected]

does the data set contain alternate phenotypes?

Definition at line 657 of file Dataset.h.

**6.4.4.16** `bool Dataset::hasContinuousPhenotypes` [protected]

does the data set contain continuous phenotypes?

Definition at line 661 of file Dataset.h.

**6.4.4.17 bool Dataset::hasGenotypes** [protected]

does the data set contain any genotypes?

Definition at line 621 of file Dataset.h.

**6.4.4.18 bool Dataset::hasNumerics** [protected]

does the data set contain any continuous attributes?

Definition at line 646 of file Dataset.h.

**6.4.4.19 std::vector<std::string> Dataset::instanceIds** [protected]

IDs associated with the instances read from file.

Definition at line 668 of file Dataset.h.

**6.4.4.20 std::vector<std::string> Dataset::instanceIdsToLoad** [protected]

IDs of instances to load from numeric and/or phenotype files.

Definition at line 670 of file Dataset.h.

**6.4.4.21 std::vector<DatasetInstance\*> Dataset::instances** [protected]

vector of pointers to all instances in the data set

Definition at line 666 of file Dataset.h.

**6.4.4.22 std::map<std::string, unsigned int> Dataset::instancesMask**  
[protected]

Definition at line 688 of file Dataset.h.

**6.4.4.23 std::map<std::string, unsigned int> Dataset::instancesMaskPushed**  
[protected]

Definition at line 692 of file Dataset.h.

**6.4.4.24 std::vector<std::map<AttributeLevel, unsigned int> > Dataset::levelCounts**  
[protected]

attribute values/levels counts

Definition at line 625 of file Dataset.h.



**6.4.4.25** `std::vector<std::map<std::pair<AttributeLevel, ClassLevel>, unsigned int> > Dataset::levelCountsByClass` [protected]

attribute values/levels counts by discrete class

Definition at line 627 of file Dataset.h.

**6.4.4.26** `bool Dataset::maskIsPushed` [protected]

Definition at line 693 of file Dataset.h.

**6.4.4.27** `std::map<std::string, std::vector<unsigned int> > Dataset::missingNumericValues` [protected]

missing continuous values and their instance indices

Definition at line 674 of file Dataset.h.

**6.4.4.28** `std::map<std::string, std::vector<unsigned int> > Dataset::missingValues` [protected]

missing discrete values and their instance indices

Definition at line 672 of file Dataset.h.

**6.4.4.29** `std::string Dataset::numericsFilename` [protected]

file from which the continuous attributes were read

Definition at line 644 of file Dataset.h.

**6.4.4.30** `std::vector<std::string> Dataset::numericIds` [protected]

IDs associated with the numerics read from file.

Definition at line 648 of file Dataset.h.

**6.4.4.31** `std::map<std::string, unsigned int> Dataset::numericsMask` [protected]

Definition at line 687 of file Dataset.h.

**6.4.4.32** `std::map<std::string, unsigned int> Dataset::numericsMaskPushed` [protected]

Definition at line 691 of file Dataset.h.

**6.4.4.33** `std::vector< std::pair<NumericLevel, NumericLevel> >`  
`Dataset::numericsMinMax` [protected]

the minimum and maximum value for each continuous attribute

Definition at line 650 of file Dataset.h.

**6.4.4.34** `std::vector<std::string>` `Dataset::numericsNames` [protected]

continuous attribute names read from file

Definition at line 652 of file Dataset.h.

**6.4.4.35** `std::vector<std::string>` `Dataset::phenotypesIds` [protected]

IDs associated with the phenotypes/classes read from file.

Definition at line 659 of file Dataset.h.

**6.4.4.36** `GSLRandomFlat*` `Dataset::rng` [protected]

random number generator classes use GNU Scienitfc Library (GSL)

Definition at line 696 of file Dataset.h.

**6.4.4.37** `std::string` `Dataset::snpsFilename` [protected]

file from which the discrete attributes (SNPSs) were read

Definition at line 619 of file Dataset.h.

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

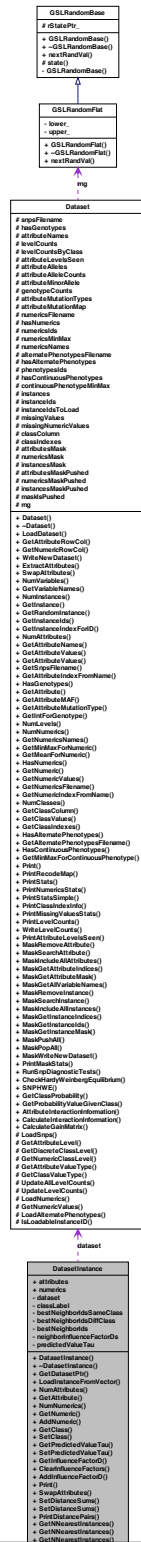
- [src/library/Dataset.h](#)
- [src/library/Dataset.cpp](#)

## 6.5 DatasetInstance Class Reference

Class to hold dataset instances (rows of attributes).

```
#include <DatasetInstance.h>
```

Generated on Fri Jan 6 2012 19:38:13 for Evaporative Cooling by Doxygen



## Public Member Functions

- [DatasetInstance](#) ([Dataset](#) \*ds)  
*Construct an data set instance object.*
- [~DatasetInstance](#) ()
- [Dataset](#) \* [GetDatasetPtr](#) ()  
*return the [Dataset](#) pointer associated with this instance*
- bool [LoadInstanceFromVector](#) (std::vector< [AttributeLevel](#) > newAttributes)  
*Load this instance with the attributes and class value from the newAttributes vector.*
- unsigned int [NumAttributes](#) ()  
*return the number of discrete attributes*
- [AttributeLevel](#) [GetAttribute](#) (unsigned int index)  
*Get and return an attribute value at index.*
- unsigned int [NumNumerics](#) ()  
*return the number of continuous attributes*
- [NumericLevel](#) [GetNumeric](#) (unsigned int index)  
*Get and return numeric value at index.*
- bool [AddNumeric](#) ([NumericLevel](#) newNum)  
*Add a numeric value to the instance's numerics vector.*
- [ClassLevel](#) [GetClass](#) ()  
*Get the discrete class value.*
- void [SetClass](#) ([ClassLevel](#) classValue)  
*Set the discrete class value.*
- double [GetPredictedValueTau](#) ()  
*Get the continuous class value.*
- void [SetPredictedValueTau](#) (double newValue)  
*Set the continuous class value.*
- double [GetInfluenceFactorD](#) (unsigned int neighborIndex)  
*Get the nearest neighbor value at neighborIndex.*
- void [ClearInfluenceFactors](#) ()  
*Clear all nearest neighbor values.*
- bool [AddInfluenceFactorD](#) (double factor)  
*Add the next nearest neighbor influence factor.*
- void [Print](#) ()  
*Print the attributes, numerics and class name of this instance to stdout.*
- bool [SwapAttributes](#) (unsigned int a1, unsigned int a2)  
*Swap attribute/column values in this instance.*
- void [SetDistanceSums](#) (unsigned int kNearestNeighbors, [DistancePairs](#) &same-ClassSums, std::map< [ClassLevel](#), [DistancePairs](#) > &diffClassSums)  
*Set the best kNearestNeighbors from the same and different classes SIDE\_EFFECT: Sorts and loads class the vairables: sameSums snd diffSums from the neighbors.*
- void [SetDistanceSums](#) (unsigned int kNearestNeighbors, [DistancePairs](#) instances-Sums)  
*Set the best kNearestNeighbors from all other instances/neighbors.*

- void [PrintDistancePairs](#) (const [DistancePairs](#) &distPairs)  
*Prints passed distance pairs.*
- bool [GetNNearestInstances](#) (unsigned int n, std::vector< unsigned int > &sameClassInstances, std::vector< unsigned int > &diffClassInstances)  
*Returns N closest instances using the sameSums and diffSums class variables.*
- bool [GetNNearestInstances](#) (unsigned int n, std::vector< unsigned int > &sameClassInstances, std::map< [ClassLevel](#), std::vector< unsigned int > > &diffClassInstances)  
*Returns N closest instances using the sameSums and diffSums class variables.*
- bool [GetNNearestInstances](#) (unsigned int n, std::vector< unsigned int > &closestInstances)  
*Returns N closest instances to this instance.*

### Public Attributes

- std::vector< [AttributeLevel](#) > [attributes](#)  
*discrete attributes*
- std::vector< [NumericLevel](#) > [numerics](#)  
*continuous attributes*

### Private Attributes

- [Dataset](#) \* [dataset](#)  
*pointer to a [Dataset](#) object*
- [ClassLevel](#) [classLabel](#)  
*the class value for this instance*
- std::vector< std::string > [bestNeighborIdsSameClass](#)  
*vector of instance IDs for the best neighbors in this instance's class*
- std::map< [ClassLevel](#), std::vector< std::string > > [bestNeighborIdsDiffClass](#)  
*vector of instance IDs for the best neighbors of different class(es)*
- std::vector< std::string > [bestNeighborIds](#)  
*best neighbor IDs for continuous class*
- std::vector< double > [neighborInfluenceFactorDs](#)  
*nearest neighbor weighting factors*
- double [predictedValueTau](#)  
*continuous value for this class*

#### 6.5.1 Detailed Description

Class to hold dataset instances (rows of attributes).

Reworked entirely for McKinney Lab work - 2/28/11

**Author**

Bill White

**Version**

1.0

Contact: [bill.c.white@gmail.com](mailto:bill.c.white@gmail.com) Created on: 6/14/05

Definition at line 40 of file DatasetInstance.h.

**6.5.2 Constructor & Destructor Documentation****6.5.2.1 DatasetInstance::DatasetInstance ( Dataset \* *ds* )**

Construct an data set instance object.

**Parameters**

<i>in</i>	<i>ds</i>	pointer to a <a href="#">Dataset</a> object
-----------	-----------	---

Definition at line 34 of file DatasetInstance.cpp.

**6.5.2.2 DatasetInstance::~DatasetInstance ( )**

Definition at line 40 of file DatasetInstance.cpp.

**6.5.3 Member Function Documentation****6.5.3.1 bool DatasetInstance::AddInfluenceFactorD ( double *factor* )**

Add the next nearest neighbor influence factor.

Definition at line 129 of file DatasetInstance.cpp.

**6.5.3.2 bool DatasetInstance::AddNumeric ( NumericLevel *newNum* )**

Add a numeric value to the instance's numerics vector.

**Parameters**

<i>in</i>	<i>newNum</i>	new numeric value
-----------	---------------	-------------------

**Returns**

success

Definition at line 99 of file DatasetInstance.cpp.

**6.5.3.3 void DatasetInstance::ClearInfluenceFactors ( )**

Clear all nearest neighbor values.

Definition at line 125 of file DatasetInstance.cpp.

**6.5.3.4 AttributeLevel DatasetInstance::GetAttribute ( unsigned int *index* )**

Get and return an attribute value at index.

**Parameters**

in	<i>index</i>	attribute index
----	--------------	-----------------

**Returns**

attribute value at index

Definition at line 559 of file Dataset.cpp.

**6.5.3.5 ClassLevel DatasetInstance::GetClass ( )**

Get the discrete class value.

Definition at line 105 of file DatasetInstance.cpp.

**6.5.3.6 Dataset \* DatasetInstance::GetDatasetPtr ( )**

return the [Dataset](#) pointer associated with this instance

Definition at line 43 of file DatasetInstance.cpp.

**6.5.3.7 double DatasetInstance::GetInfluenceFactorD ( unsigned int *neighborIndex* )**

Get the nearest neighbor value at neighborIndex.

Definition at line 121 of file DatasetInstance.cpp.

**6.5.3.8 bool DatasetInstance::GetNNearestInstances ( unsigned int *n*, std::vector< unsigned int > & *sameClassInstances*, std::vector< unsigned int > & *diffClassInstances* )**

Returns N closest instances using the sameSums and diffSums class variables.

**Parameters**

in	<i>n</i>	n nearest neighbors
in	<i>sameClassInstances</i>	vector of same class instances indices
in	<i>diffClassInstances</i>	vector of different class instance indices

**Returns**

success

6.5.3.9 `bool DatasetInstance::GetNNearestInstances ( unsigned int n, std::vector< unsigned int > & sameClassInstances, std::map< ClassLevel, std::vector< unsigned int > > & diffClassInstances )`

Returns N closest instances using the sameSums and diffSums class variables.

**Parameters**

in	<i>n</i>	n nearest neighbors
in	<i>sameClassInstances</i>	vector of same class instances indices
in	<i>diffClassInstances</i>	vector of different classes instance indices

**Returns**

success

6.5.3.10 `bool DatasetInstance::GetNNearestInstances ( unsigned int n, std::vector< unsigned int > & closestInstances )`

Returns N closest instances to this instance.

**Parameters**

in	<i>n</i>	n nearest neighbors
in	<i>closestInstances</i>	reference to a vector of instance indices

**Returns**

success

6.5.3.11 `double DatasetInstance::GetNumeric ( unsigned int index )`

Get and return numeric value at index.

**Parameters**

in	<i>index</i>	numeric index
----	--------------	---------------

**Returns**

numeric value at index



Definition at line 655 of file Dataset.cpp.

#### 6.5.3.12 double DatasetInstance::GetPredictedValueTau ( )

Get the continuous class value.

Definition at line 113 of file DatasetInstance.cpp.

#### 6.5.3.13 bool DatasetInstance::LoadInstanceFromVector ( std::vector< AttributeLevel > newAttributes )

Load this instance with the attributes and class value from the newAttributes vector.

##### Parameters

in	<i>newAttributes</i>	vector of new attribute values
----	----------------------	--------------------------------

##### Returns

success

Definition at line 48 of file DatasetInstance.cpp.

#### 6.5.3.14 unsigned int DatasetInstance::NumAttributes ( )

return the number of discrete attributes

Definition at line 60 of file DatasetInstance.cpp.

#### 6.5.3.15 unsigned int DatasetInstance::NumNumerics ( )

return the number of continuous attributes

Definition at line 80 of file DatasetInstance.cpp.

#### 6.5.3.16 void DatasetInstance::Print ( )

Print the attributes, numerics and class name of this instance to stdout.

Definition at line 134 of file DatasetInstance.cpp.

#### 6.5.3.17 void DatasetInstance::PrintDistancePairs ( const DistancePairs & distPairs )

Prints passed distance pairs.

##### Parameters

in	<i>distPairs</i>	distance pairs
----	------------------	----------------

Definition at line 240 of file DatasetInstance.cpp.

#### 6.5.3.18 void DatasetInstance::SetClass ( **ClassLevel** *classValue* )

Set the discrete class value.

Definition at line 109 of file DatasetInstance.cpp.

#### 6.5.3.19 void DatasetInstance::SetDistanceSums ( unsigned int *kNearestNeighbors*, **DistancePairs** & *sameClassSums*, std::map< **ClassLevel**, **DistancePairs** > & *diffClassSums* )

Set the best kNearestNeighbors from the same and different classes SIDE\_EFFECT: Sorts and loads class the vairables: sameSums snd diffSums from the neighbors.

##### Parameters

in	<i>kNearest-Neighbors</i>	k nearest nerighbors,
in	<i>sameClass-Sums</i>	vectors of pairs <instance, sum> of same class
in	<i>diffClass-Sums</i>	vectors of pairs <instance, sum> of other classes

##### Returns

nothing

#### 6.5.3.20 void DatasetInstance::SetDistanceSums ( unsigned int *kNearestNeighbors*, **DistancePairs** *instancesSums* )

Set the best kNearestNeighbors from all other instances/neighbors.

SIDE\_EFFECT: Sorts and loads neighborSums from the instanceSums

##### Parameters

in	<i>kNearest-Neighbors</i>	k nearest neighbors
in	<i>instance-Sums</i>	vectors of k pairs <instance, sum> for neighbors

##### Returns

nothing

Definition at line 215 of file DatasetInstance.cpp.

**6.5.3.21 void DatasetInstance::SetPredictedValueTau ( double *newValue* )**

Set the continuous class value.

Definition at line 117 of file DatasetInstance.cpp.

**6.5.3.22 bool DatasetInstance::SwapAttributes ( unsigned int *a1*, unsigned int *a2* )**

Swap attribute/column values in this instance.

**Parameters**

in	<i>a1</i>	attribue index 1
in	<i>a2</i>	attribue index 2

**Returns**

bool success

Definition at line 154 of file DatasetInstance.cpp.

**6.5.4 Member Data Documentation****6.5.4.1 std::vector<AttributeLevel> DatasetInstance::attributes**

discrete attributes

Definition at line 158 of file DatasetInstance.h.

**6.5.4.2 std::vector<std::string> DatasetInstance::bestNeighborIds [private]**

best neighbor IDs for continuous class

Definition at line 171 of file DatasetInstance.h.

**6.5.4.3 std::map<ClassLevel, std::vector<std::string> > DatasetInstance::bestNeighborIdsDiffClass [private]**

vector of instance IDs for the best neighbors of different class(es)

Definition at line 169 of file DatasetInstance.h.

**6.5.4.4 std::vector<std::string> DatasetInstance::bestNeighborIdsSameClass [private]**

vector of instance IDs for the best neighbors in this instance's class

Definition at line 167 of file DatasetInstance.h.

#### 6.5.4.5 ClassLevel DatasetInstance::classLabel [private]

the class value for this instance

Definition at line 165 of file DatasetInstance.h.

#### 6.5.4.6 Dataset\* DatasetInstance::dataset [private]

pointer to a [Dataset](#) object

Definition at line 163 of file DatasetInstance.h.

#### 6.5.4.7 std::vector<double> DatasetInstance::neighborInfluenceFactorDs [private]

nearest neighbor weighting factors

Definition at line 173 of file DatasetInstance.h.

#### 6.5.4.8 std::vector<NumericLevel> DatasetInstance::numerics

continuous attributes

Definition at line 160 of file DatasetInstance.h.

#### 6.5.4.9 double DatasetInstance::predictedValueTau [private]

countinuous value for this class

Definition at line 175 of file DatasetInstance.h.

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

- src/library/[DatasetInstance.h](#)
- src/library/[Dataset.cpp](#)
- src/library/[DatasetInstance.cpp](#)

## 6.6 deref\_less Class Reference

### Public Member Functions

- bool [operator\(\)](#) (const [T](#) a, const [T](#) b) const

### 6.6.1 Detailed Description

Definition at line 57 of file ReliefF.cpp.

## 6.6.2 Member Function Documentation

6.6.2.1 `bool deref_less::operator() ( const T a, const T b ) const` `[inline]`

Definition at line 61 of file ReliefF.cpp.

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

- [src/library/ReliefF.cpp](#)

## 6.7 deref\_less\_bcw Class Reference

### Public Member Functions

- `bool operator() (const T a, const T b) const`

### 6.7.1 Detailed Description

Definition at line 25 of file DatasetInstance.cpp.

## 6.7.2 Member Function Documentation

6.7.2.1 `bool deref_less_bcw::operator() ( const T a, const T b ) const` `[inline]`

Definition at line 29 of file DatasetInstance.cpp.

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

- [src/library/DatasetInstance.cpp](#)

## 6.8 insilico::do\_to\_lower< charT > Class Template Reference

```
#include <StringUtils.h>
```

### Public Member Functions

- `do_to_lower` (std::ctype< charT > &ct)
- `do_to_lower` (const std::locale &loc=std::locale())
- `charT operator() (charT c) const`

### Private Attributes

- `std::ctype< charT > const & m_ctype`

### 6.8.1 Detailed Description

```
template<class charT = char>class insilico::do_to_lower< charT >
```

Definition at line 79 of file StringUtils.h.

### 6.8.2 Constructor & Destructor Documentation

6.8.2.1 `template<class charT = char> insilico::do_to_lower< charT >::do_to_lower ( std::ctype< charT > & ct ) [inline]`

Definition at line 83 of file StringUtils.h.

6.8.2.2 `template<class charT = char> insilico::do_to_lower< charT >::do_to_lower ( const std::locale & loc = std::locale() ) [inline]`

Definition at line 86 of file StringUtils.h.

### 6.8.3 Member Function Documentation

6.8.3.1 `template<class charT = char> charT insilico::do_to_lower< charT >::operator() ( charT c ) const [inline]`

Definition at line 89 of file StringUtils.h.

### 6.8.4 Member Data Documentation

6.8.4.1 `template<class charT = char> std::ctype<charT> const& insilico::do_to_lower< charT >::m_ctype [private]`

Definition at line 93 of file StringUtils.h.

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

- [src/library/StringUtils.h](#)

## 6.9 insilico::do\_to\_upper< charT > Class Template Reference

```
#include <StringUtils.h>
```

### Public Member Functions

- [do\\_to\\_upper](#) (std::ctype< charT > &ct)

- [do\\_to\\_upper](#) (const std::locale &loc=std::locale())
- charT [operator\(\)](#) (charT c) const

### Private Attributes

- std::ctype< charT > const & [m\\_ctype](#)

### 6.9.1 Detailed Description

template<class charT = char> class insilico::do\_to\_upper< charT >

Definition at line 59 of file StringUtils.h.

### 6.9.2 Constructor & Destructor Documentation

6.9.2.1 template<class charT = char> insilico::do\_to\_upper< charT >::do\_to\_upper (std::ctype< charT > & ct) [inline]

Definition at line 63 of file StringUtils.h.

6.9.2.2 template<class charT = char> insilico::do\_to\_upper< charT >::do\_to\_upper (const std::locale & loc = std::locale()) [inline]

Definition at line 66 of file StringUtils.h.

### 6.9.3 Member Function Documentation

6.9.3.1 template<class charT = char> charT insilico::do\_to\_upper< charT >::operator() (charT c) const [inline]

Definition at line 69 of file StringUtils.h.

### 6.9.4 Member Data Documentation

6.9.4.1 template<class charT = char> std::ctype<charT> const& insilico::do\_to\_upper< charT >::m\_ctype [private]

Definition at line 73 of file StringUtils.h.

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

- src/library/[StringUtils.h](#)

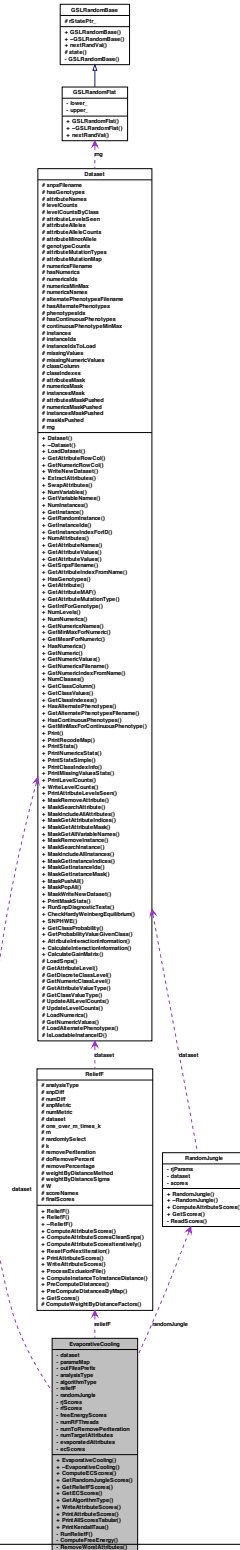
## 6.10 EvaporativeCooling Class Reference

Evaporative Cooling attribute ranking algorithm.

```
#include <EvaporativeCooling.h>
```



Generated on Fri Jan 6 2012 19:38:13 for Evaporative Cooling by Doxygen



## Public Member Functions

- [EvaporativeCooling](#) ([Dataset](#) \*ds, po::variables\_map &vm, [AnalysisType](#) anaType=SNP\_ONLY\_ANALYSIS)  
*Construct an EC algorithm object.*
- virtual [~EvaporativeCooling](#) ()
- bool [ComputeECScores](#) ()  
*Compute the EC scores based on the current set of attributes.*
- [EcScores](#) & [GetRandomJungleScores](#) ()  
*Get the last computed [RandomJungle](#) scores.*
- [EcScores](#) & [GetReliefFScores](#) ()  
*Get the last computed [ReliefF](#) scores.*
- [EcScores](#) & [GetECScores](#) ()  
*Get the last computed EC scores.*
- [EcAlgorithmType](#) [GetAlgorithmType](#) ()  
*Return the algorithm type: EC\_ALL, EC\_RJ or EC\_RF.*
- void [WriteAttributeScores](#) (std::string baseFilename)  
*Write the scores and attribute names to file.*
- void [PrintAttributeScores](#) (std::ofstream &outStream)  
*Write the scores and attribute names to stream.*
- bool [PrintAllScoresTabular](#) ()  
*Print the current attributes scores to stdout in tab-delimited format.*
- bool [PrintKendallTaus](#) ()  
*Print the kendall taus between the [ReliefF](#) and [RandomJungle](#) scores.*

## Private Member Functions

- bool [RunReliefF](#) ()  
*Run the [ReliefF](#) algorithm.*
- bool [ComputeFreeEnergy](#) (double temperature)  
*Compute the attributes' free energy using the couple temperature.*
- bool [RemoveWorstAttributes](#) (unsigned int numToRemove=1)  
*Remove the worst attribute based on free energy scores.*

## Private Attributes

- [Dataset](#) \* dataset  
*pointer to a [Dataset](#) object*
- po::variables\_map paramsMap  
*command line parameters map*
- std::string outFilesPrefix  
*prefix for all output files*
- [AnalysisType](#) analysisType

- type of analysis to perform*
- [EcAlgorithmType](#) *algorithmType*
- algorithm steps to perform*
- [ReliefF](#) \* [reliefF](#)
- pointer to a [ReliefF](#) or [RReliefF](#) algorithm object*
- [RandomJungle](#) \* [randomJungle](#)
- pointer to a [RandomJungle](#) algorithm onject*
- [EcScores](#) *rjScores*
- current random jungle scores*
- [EcScores](#) *rfScores*
- current relief scores*
- [EcScores](#) *freeEnergyScores*
- current free energy scores*
- unsigned int [numRFThreads](#)
- unsigned int [numToRemovePerIteration](#)
- number of attributes to remove per iteration*
- unsigned int [numTargetAttributes](#)
- number of target attributes*
- [EcScores](#) *evaporatedAttributes*
- attributes that have been evaporated so far*
- [EcScores](#) *ecScores*
- current set of ec scores*

### 6.10.1 Detailed Description

Evaporative Cooling attribute ranking algorithm.

Implements the Evaporative Cooling algorithm in: McKinney, et. al. "Capturing the Spectrum of Interaction Effects in Genetic Association Studies by Simulated Evaporative Cooling Network Analysis." PLoS Genetics, Vol 5, Issue 3, 2009.

#### See also

[ReliefF](#)  
[RReliefF](#)  
[RandomJungle](#)

#### Author

Bill White

#### Version

1.0

Contact: [bill.c.white@gmail.com](mailto:bill.c.white@gmail.com) Created on: 7/14/11

Definition at line 52 of file EvaporativeCooling.h.

## 6.10.2 Constructor & Destructor Documentation

### 6.10.2.1 EvaporativeCooling::EvaporativeCooling ( Dataset \* *ds*, po::variables\_map & *vm*, AnalysisType *anaType* = SNP\_ONLY\_ANALYSIS )

Construct an EC algorithm object.

#### Parameters

in	<i>ds</i>	pointer to a <a href="#">Dataset</a> object
in	<i>vm</i>	reference to a Boost map of command line options
in	<i>anaType</i>	analysis type

Definition at line 52 of file EvaporativeCooling.cpp.

### 6.10.2.2 EvaporativeCooling::~~EvaporativeCooling ( ) [virtual]

Definition at line 150 of file EvaporativeCooling.cpp.

## 6.10.3 Member Function Documentation

### 6.10.3.1 bool EvaporativeCooling::ComputeECScores ( )

Compute the EC scores based on the current set of attributes.

Definition at line 159 of file EvaporativeCooling.cpp.

### 6.10.3.2 bool EvaporativeCooling::ComputeFreeEnergy ( double *temperature* ) [private]

Compute the attributes' free energy using the couple temperature.

#### Parameters

in	<i>tempreature</i>	coupling temperature T
----	--------------------	------------------------

#### Returns

distance

Definition at line 478 of file EvaporativeCooling.cpp.

### 6.10.3.3 EcAlgorithmType EvaporativeCooling::GetAlgorithmType ( )

Return the algorithm type: EC\_ALL, EC\_RJ or EC\_RF.

Definition at line 307 of file EvaporativeCooling.cpp.

**6.10.3.4 EcScores & EvaporativeCooling::GetECScores ( )**

Get the last computed EC scores.

Definition at line 303 of file EvaporativeCooling.cpp.

**6.10.3.5 EcScores & EvaporativeCooling::GetRandomJungleScores ( )**

Get the last computed [RandomJungle](#) scores.

Definition at line 295 of file EvaporativeCooling.cpp.

**6.10.3.6 EcScores & EvaporativeCooling::GetReliefFScores ( )**

Get the last computed [ReliefF](#) scores.

Definition at line 299 of file EvaporativeCooling.cpp.

**6.10.3.7 bool EvaporativeCooling::PrintAllScoresTabular ( )**

Print the current attributes scores to stdout in tab-delimited format.

Definition at line 353 of file EvaporativeCooling.cpp.

**6.10.3.8 void EvaporativeCooling::PrintAttributeScores ( std::ofstream & *outStream* )**

Write the scores and attribute names to stream.

**Parameters**

<i>in</i>	<i>outStream</i>	stream to write score-attribute name pairs
-----------	------------------	--

**6.10.3.9 bool EvaporativeCooling::PrintKendallTaus ( )**

Print the kendall taus between the [ReliefF](#) and [RandomJungle](#) scores.

Definition at line 385 of file EvaporativeCooling.cpp.

**6.10.3.10 bool EvaporativeCooling::RemoveWorstAttributes ( unsigned int *numToRemove* = 1 )  
[private]**

Remove the worst attribute based on free energy scores.

**Parameters**

<i>in</i>	<i>numToRemove</i>	number of attributes to remove/evaporate
-----------	--------------------	--

**Returns**

distance

Definition at line 520 of file EvaporativeCooling.cpp.

**6.10.3.11 bool EvaporativeCooling::RunReliefF ( ) [private]**

Run the [ReliefF](#) algorithm.

Definition at line 428 of file EvaporativeCooling.cpp.

**6.10.3.12 void EvaporativeCooling::WriteAttributeScores ( std::string *baseFilename* )**

Write the scores and attribute names to file.

**Parameters**

in	<i>baseFilename</i>	filename to write score-attribute name pairs
----	---------------------	--

**6.10.4 Member Data Documentation****6.10.4.1 EcAlgorithmType EvaporativeCooling::algorithmType [private]**

algorithm steps to perform

Definition at line 115 of file EvaporativeCooling.h.

**6.10.4.2 AnalysisType EvaporativeCooling::analysisType [private]**

type of analysis to perform

**See also**

[ReliefF](#)

Definition at line 113 of file EvaporativeCooling.h.

**6.10.4.3 Dataset\* EvaporativeCooling::dataset [private]**

pointer to a [Dataset](#) object

Definition at line 105 of file EvaporativeCooling.h.

**6.10.4.4 EcScores EvaporativeCooling::ecScores [private]**

current set of ec scores

Definition at line 138 of file EvaporativeCooling.h.

#### 6.10.4.5 EcScores EvaporativeCooling::evaporatedAttributes [private]

attributes that have been evaporated so far

Definition at line 136 of file EvaporativeCooling.h.

#### 6.10.4.6 EcScores EvaporativeCooling::freeEnergyScores [private]

current free energy scores

Definition at line 127 of file EvaporativeCooling.h.

#### 6.10.4.7 unsigned int EvaporativeCooling::numRFThreads [private]

Definition at line 130 of file EvaporativeCooling.h.

#### 6.10.4.8 unsigned int EvaporativeCooling::numTargetAttributes [private]

number of target attributes

Definition at line 134 of file EvaporativeCooling.h.

#### 6.10.4.9 unsigned int EvaporativeCooling::numToRemovePerIteration [private]

number of attributes to remove per iteration

Definition at line 132 of file EvaporativeCooling.h.

#### 6.10.4.10 std::string EvaporativeCooling::outFilesPrefix [private]

prefix for all output files

Definition at line 109 of file EvaporativeCooling.h.

#### 6.10.4.11 po::variables\_map EvaporativeCooling::paramsMap [private]

command line parameters map

Definition at line 107 of file EvaporativeCooling.h.

#### 6.10.4.12 RandomJungle\* EvaporativeCooling::randomJungle [private]

pointer to a [RandomJungle](#) algorithm object

Definition at line 120 of file EvaporativeCooling.h.

#### 6.10.4.13 ReliefF\* EvaporativeCooling::reliefF [private]

pointer to a [ReliefF](#) or [RReliefF](#) algorithm object

Definition at line 118 of file EvaporativeCooling.h.

#### 6.10.4.14 EcScores EvaporativeCooling::rfScores [private]

current relieff scores

Definition at line 125 of file EvaporativeCooling.h.

#### 6.10.4.15 EcScores EvaporativeCooling::rjScores [private]

current random jungle scores

Definition at line 123 of file EvaporativeCooling.h.

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

- [src/library/EvaporativeCooling.h](#)
- [src/library/EvaporativeCooling.cpp](#)

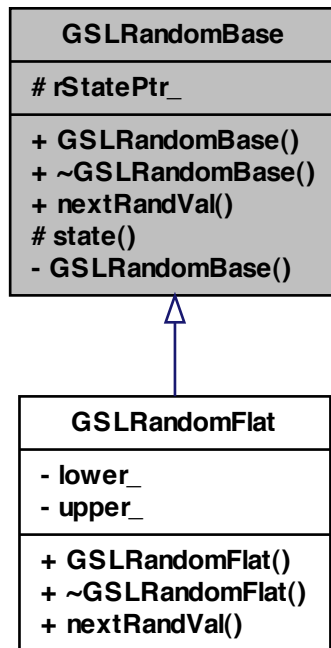
## 6.11 GSLRandomBase Class Reference

A base class for GNU Scientific Library (GSL) random number functions.

```
#include <GSLRandomBase.h>
```



Inheritance diagram for GSLRandomBase:



### Public Member Functions

- [GSLRandomBase](#) (int seedVal)
- virtual [~GSLRandomBase](#) ()
- virtual double [nextRandVal](#) ()=0

### Protected Member Functions

- `gsl_rng * state ()`

### Protected Attributes

- `gsl_rng * rStatePtr\_`

## Private Member Functions

- [GSLRandomBase](#) (const [GSLRandomBase](#) &rhs)

### 6.11.1 Detailed Description

A base class for GNU Scientific Library (GSL) random number functions.

The setup, initialization and clean-up is the same for all GSL random number functions. This class abstracts away these details, placing the setup and initialization in the class constructor and the clean-up in the class destructor. The class constructor is passed a seed value for the random number generator.

A class that provides access to one or more GSL random number functions should be derived from this class. This class must provide an implementation for the [nextRandVal\(\)](#) pure virtual function. The nextRandVal will call the specific random number function (for example `gsl_ran_ugaussian()` for Gaussian distribution or `gsl_ran_flat()` for a flat random number distribution).

This class uses the default random number generator. At least on Windows XP using the Visual C++ 6.0 compiler the type definitions for the random functions (for example `gsl_rng_mt19937` or `gsl_rng_knuthran`) would not link properly. Perhaps they are not properly exported from the pre-built library.

I decided to use the GSL because it is supported on all major platforms (UNIX, Linux and Windows) and provides high quality pseudo-random number generation support. The standard POSIX `rand()` function is notorious for its poor quality. While the `random()` function on UNIX provides better pseudo-random number quality, but is still not as good as functions like MT19937.

Definition at line 39 of file `GSLRandomBase.h`.

### 6.11.2 Constructor & Destructor Documentation

**6.11.2.1** `GSLRandomBase::GSLRandomBase ( const GSLRandomBase & rhs )`  
`[private]`

**6.11.2.2** `GSLRandomBase::GSLRandomBase ( int seedVal )` `[inline]`

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

**6.11.2.3** `virtual GSLRandomBase::~GSLRandomBase ( )` `[inline, virtual]`

Definition at line 67 of file `GSLRandomBase.h`.

### 6.11.3 Member Function Documentation

6.11.3.1 `virtual double GSLRandomBase::nextRandVal ( )` [pure virtual]

Implemented in [GSLRandomFlat](#).

6.11.3.2 `gsl_rng* GSLRandomBase::state ( )` [inline, protected]

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

#### 6.11.4 Member Data Documentation

6.11.4.1 `gsl_rng* GSLRandomBase::rStatePtr_` [protected]

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

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

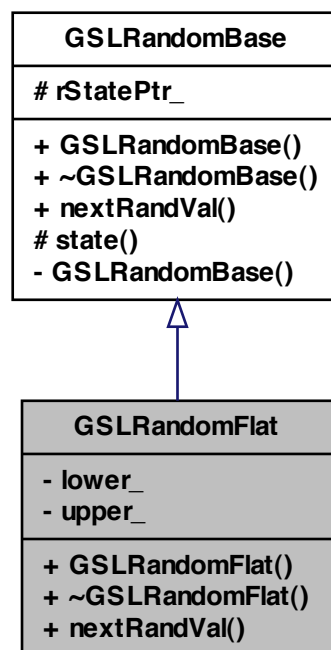
- `src/library/GSLRandomBase.h`

## 6.12 GSLRandomFlat Class Reference

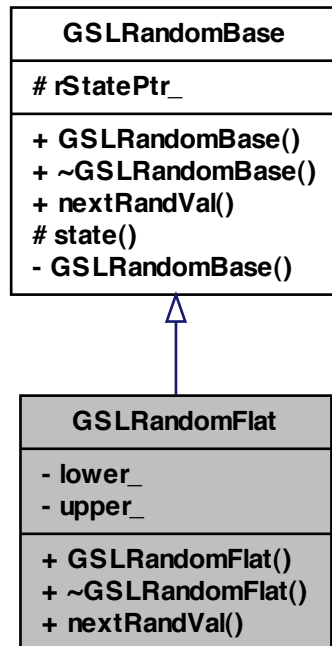
Random numbers in a flat, or uniform distribution.

```
#include <GSLRandomFlat.h>
```

Inheritance diagram for GSLRandomFlat:



Collaboration diagram for GSLRandomFlat:



### Public Member Functions

- [GSLRandomFlat](#) (int seedVal, double lower, double upper)
- [~GSLRandomFlat](#) ()
- double [nextRandVal](#) ()

### Private Attributes

- double [lower\\_](#)
- double [upper\\_](#)

#### 6.12.1 Detailed Description

Random numbers in a flat, or uniform distribution.

The class constructor is given a seed and a lower and upper bound value for the uniform distribution. The random numbers that result will be a uniform distribution in the range

```
lower <= randVal < upper
```

Definition at line 21 of file GSLRandomFlat.h.

### 6.12.2 Constructor & Destructor Documentation

6.12.2.1 `GSLRandomFlat::GSLRandomFlat ( int seedVal, double lower, double upper )`  
[inline]

Definition at line 27 of file GSLRandomFlat.h.

6.12.2.2 `GSLRandomFlat::~GSLRandomFlat ( )` [inline]

Definition at line 36 of file GSLRandomFlat.h.

### 6.12.3 Member Function Documentation

6.12.3.1 `double GSLRandomFlat::nextRandVal ( )` [inline, virtual]

Implements [GSLRandomBase](#).

Definition at line 40 of file GSLRandomFlat.h.

### 6.12.4 Member Data Documentation

6.12.4.1 `double GSLRandomFlat::lower_` [private]

Definition at line 23 of file GSLRandomFlat.h.

6.12.4.2 `double GSLRandomFlat::upper_` [private]

Definition at line 23 of file GSLRandomFlat.h.

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

- [src/library/GSLRandomFlat.h](#)

## 6.13 insilico::is\_classified< Type, charT > Class Template Reference

```
#include <StringUtils.h>
```

## Public Member Functions

- [is\\_classified](#) (std::ctype< charT > &ct)
- [is\\_classified](#) (const std::locale &loc=std::locale())
- bool [operator\(\)](#) (charT c) const

## Private Attributes

- std::ctype< charT > const & [m\\_ctype](#)

### 6.13.1 Detailed Description

template<std::ctype\_base::mask Type, class charT = char>class insilico::is\_classified< Type, charT >

Definition at line 40 of file StringUtils.h.

### 6.13.2 Constructor & Destructor Documentation

6.13.2.1 template<std::ctype\_base::mask Type, class charT = char>  
insilico::is\_classified< Type, charT >::is\_classified ( std::ctype< charT > & ct  
) [inline]

Definition at line 44 of file StringUtils.h.

6.13.2.2 template<std::ctype\_base::mask Type, class charT = char>  
insilico::is\_classified< Type, charT >::is\_classified ( const std::locale & loc =  
std::locale() ) [inline]

Definition at line 47 of file StringUtils.h.

### 6.13.3 Member Function Documentation

6.13.3.1 template<std::ctype\_base::mask Type, class charT = char> bool  
insilico::is\_classified< Type, charT >::operator() ( charT c ) const [inline]

Definition at line 50 of file StringUtils.h.

### 6.13.4 Member Data Documentation

6.13.4.1 template<std::ctype\_base::mask Type, class charT = char> std::ctype<charT>  
const& insilico::is\_classified< Type, charT >::m\_ctype [private]

Definition at line 54 of file StringUtils.h.

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

- src/library/[StringUtils.h](#)

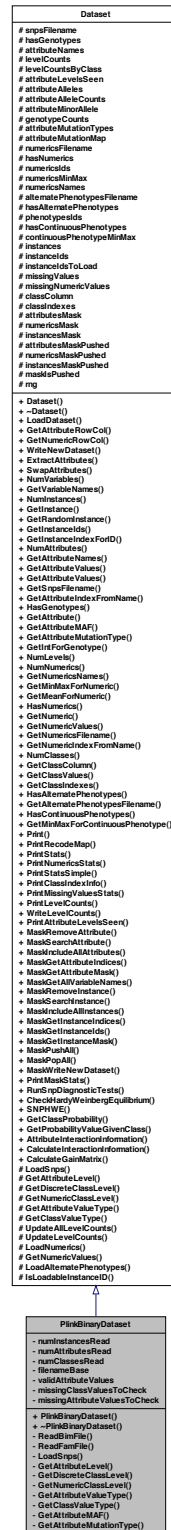
## 6.14 PlinkBinaryDataset Class Reference

Plink binary PED/BED file format reader.

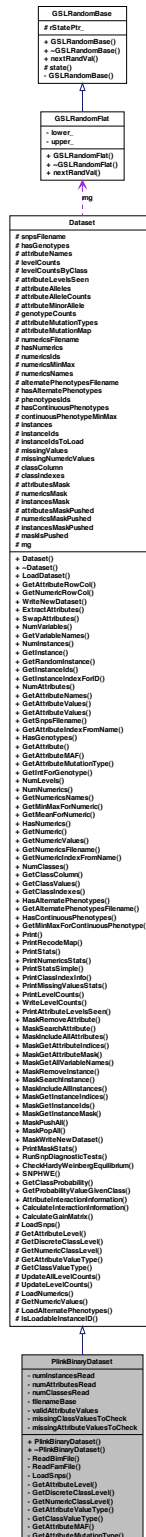
```
#include <PlinkBinaryDataset.h>
```



Inheritance diagram for PlinkBinaryDataset:



Collaboration diagram for PlinkBinaryDataset:



## Public Member Functions

- [PlinkBinaryDataset](#) ()
- [~PlinkBinaryDataset](#) ()

## Private Member Functions

- bool [ReadBimFile](#) (std::string bimFilename)  
*Load attribute information.*
- bool [ReadFamFile](#) (std::string famFilename)  
*Load individual information.*
- bool [LoadSnps](#) (std::string filename)  
*Load SNPs from file using the data set filename.*
- bool [GetAttributeLevel](#) (std::string inLevel, std::vector< std::string > [missingValues](#), [AttributeLevel](#) &outLevel)  
*Get the attribute level based on string representation.*
- bool [GetDiscreteClassLevel](#) (std::string inLevel, std::vector< std::string > [missingValues](#), [ClassLevel](#) &outLevel)  
*Get the discrete class level based on string representation.*
- bool [GetNumericClassLevel](#) (std::string inLevel, std::vector< std::string > [missingValues](#), [NumericLevel](#) &outLevel)  
*Get the numeric class level based on string representation.*
- [ValueType](#) [GetAttributeValueType](#) (std::string value, std::vector< std::string > [missingValues](#))  
*Get the passed attribute value's type.*
- [ValueType](#) [GetClassValueType](#) (std::string value, std::vector< std::string > [missingValues](#))  
*Get the passed class value's type.*
- std::pair< char, double > [GetAttributeMAF](#) (unsigned int attributeIndex)  
*Get attribute minor allele and frequency.*
- [AttributeMutationType](#) [GetAttributeMutationType](#) (unsigned int attributeIndex)  
*Get attribute mutation type.*

## Private Attributes

- unsigned int [numInstancesRead](#)
- unsigned int [numAttributesRead](#)
- unsigned int [numClassesRead](#)
- std::string [filenameBase](#)
- std::vector< std::string > [validAttributeValues](#)  
*for checking attribute values*
- std::vector< std::string > [missingClassValuesToCheck](#)  
*missing class values*
- std::vector< std::string > [missingAttributeValuesToCheck](#)  
*missing attribute values*

### 6.14.1 Detailed Description

Plink binary PED/BED file format reader.

#### See also

[Dataset](#)

#### Author

Bill White

#### Version

1.0

Contact: [bill.c.white@gmail.com](mailto:bill.c.white@gmail.com) Created on: 3/10/11

Definition at line 21 of file PlinkBinaryDataset.h.

### 6.14.2 Constructor & Destructor Documentation

#### 6.14.2.1 PlinkBinaryDataset::PlinkBinaryDataset ( )

Definition at line 36 of file PlinkBinaryDataset.cpp.

#### 6.14.2.2 PlinkBinaryDataset::~~PlinkBinaryDataset ( )

### 6.14.3 Member Function Documentation

#### 6.14.3.1 bool PlinkBinaryDataset::GetAttributeLevel ( std::string *inLevel*, std::vector< std::string > *missingValues*, AttributeLevel & *outLevel* ) [private, virtual]

Get the attribute level based on string representation.

#### Parameters

in	<i>inLevel</i>	attribute level read from file
in	<i>missingValues</i>	list of strings representing missing attribute values
out	<i>outLevel</i>	attribute level to use in the data set class

#### Returns

success

Reimplemented from [Dataset](#).

Definition at line 390 of file ArffDataset.cpp.

**6.14.3.2** `pair< char, double > PlinkBinaryDataset::GetAttributeMAF ( unsigned int  
attributeIndex ) [private, virtual]`

Get attribute minor allele and frequency.

#### Parameters

in	attribute	index
----	-----------	-------

#### Returns

pair (minor allele, minor allele frequency)

An Intriduction to Genetic Analysis by Griffiths, Miller, Suzuki, Lewontin and Gelbart, 2000, page 715.

Reimplemented from [Dataset](#).

Definition at line 629 of file PlinkBinaryDataset.cpp.

**6.14.3.3** `AttributeMutationType PlinkBinaryDataset::GetAttributeMutationType ( unsigned  
int attributeIndex ) [private, virtual]`

Get attribute mutation type.

#### Parameters

in	attribute	index
----	-----------	-------

#### Returns

mutation type (transition, transversion, unknown)

Reimplemented from [Dataset](#).

Definition at line 638 of file PlinkBinaryDataset.cpp.

**6.14.3.4** `ValueType PlinkBinaryDataset::GetAttributeValueType ( std::string value,  
std::vector< std::string > missingValues ) [private, virtual]`

Get the passed attribute value's type.

#### Parameters

in	value	value to check
in	missingValues	vector of possible missing values

#### Returns

value's type

Reimplemented from [Dataset](#).

Definition at line 2028 of file Dataset.cpp.

**6.14.3.5** `ValueType PlinkBinaryDataset::GetClassValueType ( std::string value, std::vector< std::string > missingValues ) [private, virtual]`

Get the passed class value's type.

#### Parameters

in	<i>value</i>	value to check
in	<i>missingValues</i>	vector of possible missing values

#### Returns

value's type

Reimplemented from [Dataset](#).

**6.14.3.6** `bool PlinkBinaryDataset::GetDiscreteClassLevel ( std::string inLevel, std::vector< std::string > missingValues, ClassLevel & outLevel ) [private, virtual]`

Get the discrete class level based on string representation.

#### Parameters

in	<i>inLevel</i>	class level read from file
in	<i>missingValues</i>	list of strings representing missing class values
out	<i>outLevel</i>	discrete class level to use in the data set class

#### Returns

success

Reimplemented from [Dataset](#).

**6.14.3.7** `bool PlinkBinaryDataset::GetNumericClassLevel ( std::string inLevel, std::vector< std::string > missingValues, NumericLevel & outLevel ) [private, virtual]`

Get the numeric class level based on string representation.

#### Parameters

in	<i>inLevel</i>	class level read from file
----	----------------	----------------------------

in	<i>missingValues</i>	list of strings representing missing class values
in, out	<i>outLevel</i>	numeric class level to use in the data set class

**Returns**

success

Reimplemented from [Dataset](#).

**6.14.3.8** `bool PlinkBinaryDataset::LoadSnps ( std::string filename ) [private, virtual]`

Load SNPs from file using the data set filename.

**Parameters**

in	<i>filename</i>	SNPs filename
in	<i>deRecodeA</i>	perform a recodeA operation after reading raw data?

**Returns**

success

Reimplemented from [Dataset](#).

**6.14.3.9** `bool PlinkBinaryDataset::ReadBimFile ( std::string bimFilename ) [private]`

Load attribute information.

**Parameters**

in	<i>PLINK</i>	bim filename
----	--------------	--------------

**Returns**

success

set the mutation type

Definition at line 333 of file PlinkBinaryDataset.cpp.

**6.14.3.10** `bool PlinkBinaryDataset::ReadFamFile ( std::string famFilename ) [private]`

Load individual information.

**Parameters**

in	<i>PLIN</i>	fam filename
----	-------------	--------------

**Returns**

success

determine class data type

assign class level

Definition at line 402 of file PlinkBinaryDataset.cpp.

**6.14.4 Member Data Documentation**

**6.14.4.1** `std::string PlinkBinaryDataset::filenameBase` [private]

Definition at line 60 of file PlinkBinaryDataset.h.

**6.14.4.2** `std::vector<std::string> PlinkBinaryDataset::missingAttributeValuesToCheck`  
[private]

missing attribute values

Definition at line 67 of file PlinkBinaryDataset.h.

**6.14.4.3** `std::vector<std::string> PlinkBinaryDataset::missingClassValuesToCheck`  
[private]

missing class values

Definition at line 65 of file PlinkBinaryDataset.h.

**6.14.4.4** `unsigned int PlinkBinaryDataset::numAttributesRead` [private]

Definition at line 57 of file PlinkBinaryDataset.h.

**6.14.4.5** `unsigned int PlinkBinaryDataset::numClassesRead` [private]

Definition at line 58 of file PlinkBinaryDataset.h.

**6.14.4.6** `unsigned int PlinkBinaryDataset::numInstancesRead` [private]

Definition at line 56 of file PlinkBinaryDataset.h.

**6.14.4.7** `std::vector<std::string> PlinkBinaryDataset::validAttributeValues`  
[private]

for checking attribute values



Definition at line 63 of file PlinkBinaryDataset.h.

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

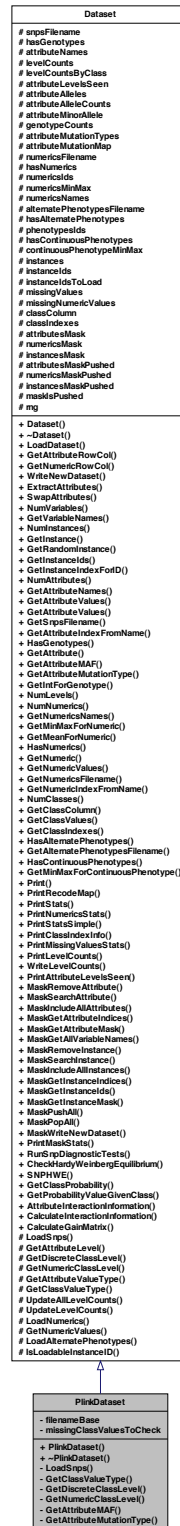
- src/library/[PlinkBinaryDataset.h](#)
- src/library/[ArffDataset.cpp](#)
- src/library/[Dataset.cpp](#)
- src/library/[PlinkBinaryDataset.cpp](#)

## 6.15 PlinkDataset Class Reference

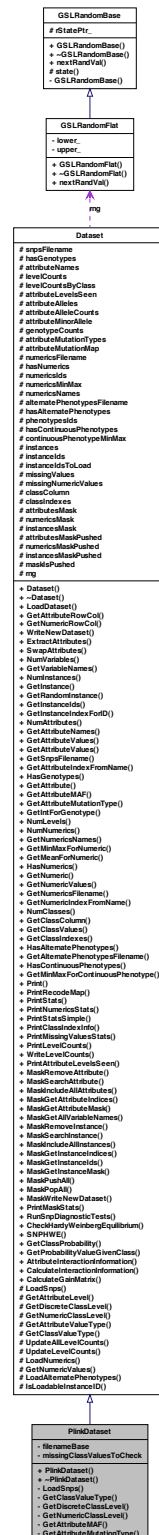
Plink MAP/PED file format reader.

```
#include <PlinkDataset.h>
```

Inheritance diagram for PlinkDataset:



Collaboration diagram for PlinkDataset:



## Public Member Functions

- [PlinkDataset](#) ()  
*Construct a PLINK data set reader. Calls [Dataset](#) base class constructor.*
- [~PlinkDataset](#) ()

## Private Member Functions

- bool [LoadSnps](#) (std::string filename)  
*Load SNPs from file using the data set filename.*
- [ValueType](#) [GetClassValueType](#) (std::string value, std::vector< std::string > [missingValues](#))  
*Get the passed class value's type.*
- bool [GetDiscreteClassLevel](#) (std::string inLevel, std::vector< std::string > [missingValues](#), [ClassLevel](#) &outLevel)  
*Get the discrete class level based on string representation.*
- bool [GetNumericClassLevel](#) (std::string inLevel, std::vector< std::string > [missingValues](#), [NumericLevel](#) &outLevel)  
*Get the numeric class level based on string representation.*
- std::pair< char, double > [GetAttributeMAF](#) (unsigned int attributeIndex)  
*Get attribute minor allele and frequency.*
- [AttributeMutationType](#) [GetAttributeMutationType](#) (unsigned int attributeIndex)  
*Get attribute mutation type.*

## Private Attributes

- std::string [filenameBase](#)  
*base filename for auxiliary files*
- std::vector< std::string > [missingClassValuesToCheck](#)  
*missing class values*

### 6.15.1 Detailed Description

Plink MAP/PED file format reader.

#### See also

[Dataset](#)

#### Author

Bill White

#### Version

1.0

Contact: [bill.c.white@gmail.com](mailto:bill.c.white@gmail.com) Created on: 2/1/11

Definition at line 35 of file PlinkDataset.h.

## 6.15.2 Constructor & Destructor Documentation

### 6.15.2.1 PlinkDataset::PlinkDataset ( )

Construct a PLINK data set reader. Calls [Dataset](#) base class constructor.

Definition at line 26 of file PlinkDataset.cpp.

### 6.15.2.2 PlinkDataset::~PlinkDataset ( )

## 6.15.3 Member Function Documentation

### 6.15.3.1 pair< char, double > PlinkDataset::GetAttributeMAF ( unsigned int *attributeIndex* ) [private, virtual]

Get attribute minor allele and frequency.

#### Parameters

in	<i>attribute</i>	index
----	------------------	-------

#### Returns

pair (minor allele, minor allele frequency)

An Intriduction to Genetic Analysis by Griffiths, Miller, Suzuki, Lewontin and Gelbart, 2000, page 715.

Reimplemented from [Dataset](#).

Definition at line 457 of file PlinkDataset.cpp.

### 6.15.3.2 AttributeMutationType PlinkDataset::GetAttributeMutationType ( unsigned int *attributeIndex* ) [private, virtual]

Get attribute mutation type.

#### Parameters

in	<i>attribute</i>	index
----	------------------	-------

#### Returns

mutation type (transition, transversion, unknown)

Reimplemented from [Dataset](#).

Definition at line 466 of file PlinkDataset.cpp.

**6.15.3.3** `ValueType PlinkDataset::GetClassValueType ( std::string value, std::vector< std::string > missingValues ) [private, virtual]`

Get the passed class value's type.

#### Parameters

in	<i>value</i>	value to check
in	<i>missingValues</i>	vector of possible missing values

#### Returns

value's type

Reimplemented from [Dataset](#).

**6.15.3.4** `bool PlinkDataset::GetDiscreteClassLevel ( std::string inLevel, std::vector< std::string > missingValues, ClassLevel & outLevel ) [private, virtual]`

Get the discrete class level based on string representation.

#### Parameters

in	<i>inLevel</i>	class level read from file
in	<i>missingValues</i>	list of strings representing missing class values
out	<i>outLevel</i>	discrete class level to use in the data set class

#### Returns

success

Reimplemented from [Dataset](#).

**6.15.3.5** `bool PlinkDataset::GetNumericClassLevel ( std::string inLevel, std::vector< std::string > missingValues, NumericLevel & outLevel ) [private, virtual]`

Get the numeric class level based on string representation.

#### Parameters

in	<i>inLevel</i>	class level read from file
in	<i>missingValues</i>	list of strings representing missing class values
in, out	<i>outLevel</i>	numeric class level to use in the data set class

Returns

success

Reimplemented from [Dataset](#).

6.15.3.6 `bool PlinkDataset::LoadSnps ( std::string filename ) [private, virtual]`

Load SNPs from file using the data set filename.

Parameters

in	<i>filename</i>	SNPs filename
in	<i>deRecodeA</i>	perform a recodeA operation after reading raw data?

Returns

success

Reimplemented from [Dataset](#).

6.15.4 Member Data Documentation

6.15.4.1 `std::string PlinkDataset::filenameBase [private]`

base filename for auxiliary files

Definition at line 55 of file PlinkDataset.h.

6.15.4.2 `std::vector<std::string> PlinkDataset::missingClassValuesToCheck [private]`

missing class values

Definition at line 57 of file PlinkDataset.h.

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

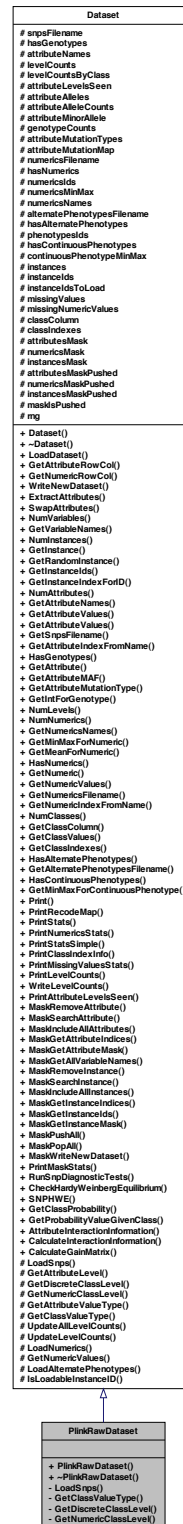
- [src/library/PlinkDataset.h](#)
- [src/library/PlinkDataset.cpp](#)

6.16 PlinkRawDataset Class Reference

Plink recodeA/RAW file format reader.

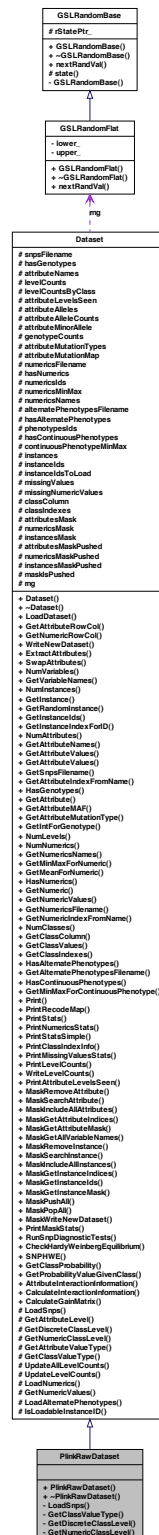
```
#include <PlinkRawDataset.h>
```

Inheritance diagram for PlinkRawDataset:





Collaboration diagram for PlinkRawDataset:



## Public Member Functions

- [PlinkRawDataset](#) ()
- [~PlinkRawDataset](#) ()

## Private Member Functions

- bool [LoadSnps](#) (std::string filename)  
*Load SNPs from file using the data set filename.*
- [ValueType](#) [GetClassValueType](#) (std::string value, std::vector< std::string > [missingValues](#))  
*Get the passed class value's type.*
- bool [GetDiscreteClassLevel](#) (std::string inLevel, std::vector< std::string > [missingValues](#), [ClassLevel](#) &outLevel)  
*Get the discrete class level based on string representation.*
- bool [GetNumericClassLevel](#) (std::string inLevel, std::vector< std::string > [missingValues](#), [NumericLevel](#) &outLevel)  
*Get the numeric class level based on string representation.*

### 6.16.1 Detailed Description

Plink recodeA/RAW file format reader.

#### See also

[Dataset](#)

#### Author

Bill White

#### Version

1.0

Contact: [bill.c.white@gmail.com](mailto:bill.c.white@gmail.com) Created on: 2/24/11

Definition at line 23 of file PlinkRawDataset.h.

### 6.16.2 Constructor & Destructor Documentation

#### 6.16.2.1 [PlinkRawDataset::PlinkRawDataset](#) ( )

Definition at line 22 of file PlinkRawDataset.cpp.

## 6.16.2.2 PlinkRawDataset::~~PlinkRawDataset ( )

## 6.16.3 Member Function Documentation

6.16.3.1 ValueType PlinkRawDataset::GetClassValueType ( std::string *value*, std::vector< std::string> *missingValues* ) [private, virtual]

Get the passed class value's type.

**Parameters**

in	<i>value</i>	value to check
in	<i>missingValues</i>	vector of possible missing values

**Returns**

value's type

Reimplemented from [Dataset](#).

Definition at line 2044 of file Dataset.cpp.

6.16.3.2 bool PlinkRawDataset::GetDiscreteClassLevel ( std::string *inLevel*, std::vector< std::string> *missingValues*, ClassLevel & *outLevel* ) [private, virtual]

Get the discrete class level based on string representation.

**Parameters**

in	<i>inLevel</i>	class level read from file
in	<i>missingValues</i>	list of strings representing missing class values
out	<i>outLevel</i>	discrete class level to use in the data set class

**Returns**

success

Reimplemented from [Dataset](#).

Definition at line 407 of file ArffDataset.cpp.

6.16.3.3 bool PlinkRawDataset::GetNumericClassLevel ( std::string *inLevel*, std::vector< std::string> *missingValues*, NumericLevel & *outLevel* ) [private, virtual]

Get the numeric class level based on string representation.

**Parameters**

in	<i>inLevel</i>	class level read from file
in	<i>missingValues</i>	list of strings representing missing class values
in, out	<i>outLevel</i>	numeric class level to use in the data set class

**Returns**

success

Reimplemented from [Dataset](#).

Definition at line 424 of file ArffDataset.cpp.

**6.16.3.4** `bool PlinkRawDataset::LoadSnps ( std::string filename ) [private, virtual]`

Load SNPs from file using the data set filename.

**Parameters**

in	<i>filename</i>	SNPs filename
in	<i>deRecodeA</i>	perform a recodeA operation after reading raw data?

**Returns**

success

assign class level

Reimplemented from [Dataset](#).

Definition at line 31 of file ArffDataset.cpp.

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

- src/library/[PlinkRawDataset.h](#)
- src/library/[ArffDataset.cpp](#)
- src/library/[CleanSnpDataset.cpp](#)
- src/library/[Dataset.cpp](#)
- src/library/[PlinkBinaryDataset.cpp](#)
- src/library/[PlinkDataset.cpp](#)
- src/library/[PlinkRawDataset.cpp](#)

## 6.17 RandomJungle Class Reference

[RandomJungle](#) attribute ranking algorithm.

```
#include <RandomJungle.h>
```

Collaboration diagram for RandomJungle:



## Public Member Functions

- [RandomJungle](#) ([Dataset](#) \*ds, po::variables\_map &vm)

*Construct an [RandomJungle](#) algorithm object.*

- virtual [~RandomJungle](#) ()
- bool [ComputeAttributeScores](#) ()
- std::vector< std::pair< double, std::string > > [GetScores](#) ()

*Get the (importance) scores as a vector of pairs: score, attribute name.*

## Private Member Functions

- bool [ReadScores](#) (std::string importanceFilename)

*Read the importance scores as attribute rankings from file.*

## Private Attributes

- RJunglePar [rjParams](#)

*[RandomJungle](#) parameters object.*

- [Dataset](#) \* [dataset](#)

*pointer to a [Dataset](#) object*

- std::vector< std::pair< double, std::string > > [scores](#)

*vector of pairs: scores, attribute names*

### 6.17.1 Detailed Description

[RandomJungle](#) attribute ranking algorithm.

Adapter class to map EC call for Random Jungle importance scores to Random Jungle library functions.

#### Author

Bill White

#### Version

1.0

Contact: [bill.c.white@gmail.com](mailto:bill.c.white@gmail.com) Created on: 10/16/11

Definition at line 28 of file RandomJungle.h.

## 6.17.2 Constructor & Destructor Documentation

### 6.17.2.1 RandomJungle::RandomJungle ( Dataset \* *ds*, po::variables\_map & *vm* )

Construct an [RandomJungle](#) algorithm object.

#### Parameters

in	<i>ds</i>	pointer to a <a href="#">Dataset</a> object
in	<i>vm</i>	reference to a Boost map of command line options

Definition at line 26 of file RandomJungle.cpp.

### 6.17.2.2 RandomJungle::~~RandomJungle ( ) [virtual]

Definition at line 63 of file RandomJungle.cpp.

## 6.17.3 Member Function Documentation

### 6.17.3.1 bool RandomJungle::ComputeAttributeScores ( )

Definition at line 69 of file RandomJungle.cpp.

### 6.17.3.2 vector< pair< double, string > > RandomJungle::GetScores ( )

Get the (importance) scores as a vector of pairs: score, attribute name.

#### Returns

vector of pairs

Definition at line 331 of file RandomJungle.cpp.

### 6.17.3.3 bool RandomJungle::ReadScores ( std::string *importanceFilename* ) [private]

Read the importance scores as attribute rankings from file.

Definition at line 335 of file RandomJungle.cpp.

## 6.17.4 Member Data Documentation

### 6.17.4.1 Dataset\* RandomJungle::dataset [private]

pointer to a [Dataset](#) object

Definition at line 50 of file RandomJungle.h.

#### 6.17.4.2 `RJunglePar RandomJungle::rjParams` [private]

[RandomJungle](#) parameters object.

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

#### 6.17.4.3 `std::vector<std::pair<double, std::string> > RandomJungle::scores` [private]

vector of pairs: scores, attribute names

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

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

- `src/library/RandomJungle.h`
- `src/library/RandomJungle.cpp`

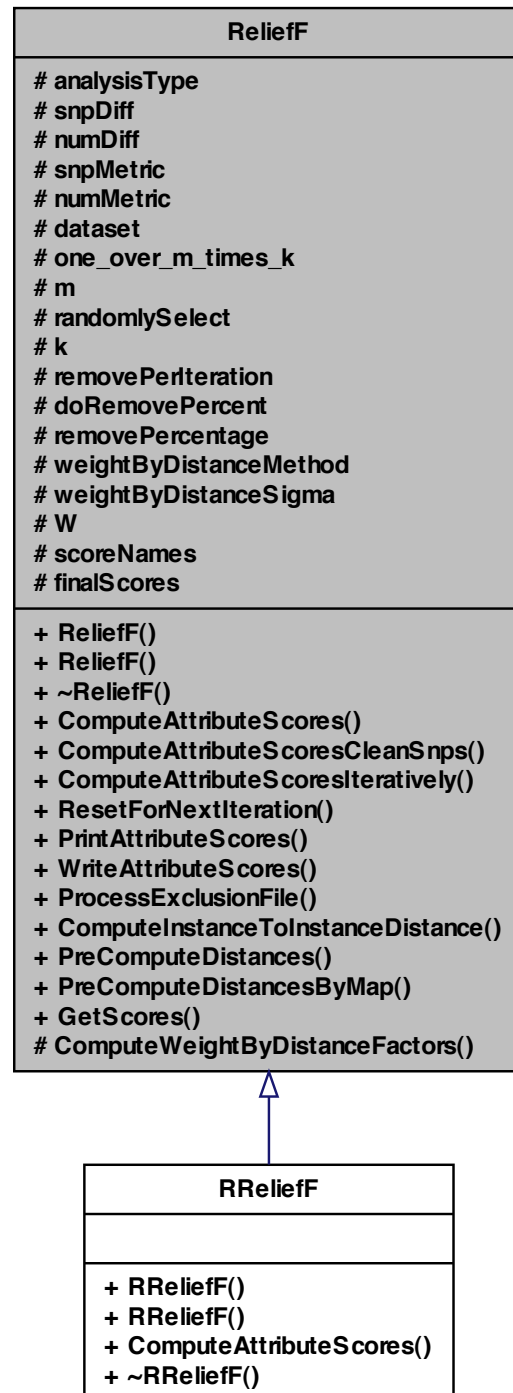
## 6.18 ReliefF Class Reference

[ReliefF](#) attribute ranking algorithm.

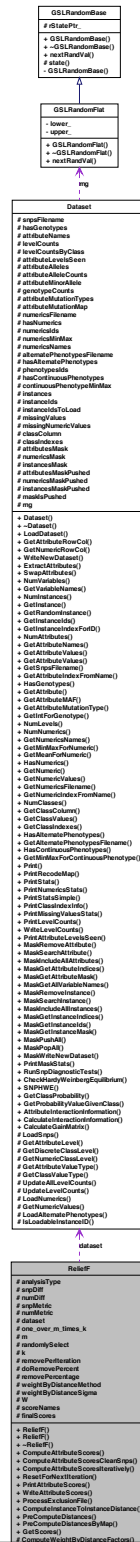
```
#include <ReliefF.h>
```



Inheritance diagram for ReliefF:



Collaboration diagram for ReliefF:



## Public Member Functions

- [ReliefF](#) ([Dataset](#) \*ds, [AnalysisType](#) anaType)  
*Construct an [ReliefF](#) algorithm object.*
- [ReliefF](#) ([Dataset](#) \*ds, po::variables\_map &vm, [AnalysisType](#) anaType)  
*Construct an [ReliefF](#) algorithm object.*
- [~ReliefF](#) ()
- virtual bool [ComputeAttributeScores](#) ()  
*Compute the [ReliefF](#) scores for the current set of attributes.*
- bool [ComputeAttributeScoresCleanSnps](#) ()  
*Compute the [ReliefF](#) scores for the current set of attributes 0/1/2 encoded.*
- bool [ComputeAttributeScoresIteratively](#) ()  
*Compute the [ReliefF](#) scores by iteratively removing worst attributes.*
- bool [ResetForNextIteration](#) ()  
*Resets some data structures for the next iteration of [ReliefF](#).*
- void [PrintAttributeScores](#) (std::ofstream &outStream)  
*Write the scores and attribute names to stream.*
- void [WriteAttributeScores](#) (std::string baseFilename)  
*Write the scores and attribute names to file.*
- bool [ProcessExclusionFile](#) (std::string exclusionFilename)  
*Remove file of attribute names from consideration in [ReliefF](#).*
- double [ComputeInstanceToInstanceDistance](#) ([DatasetInstance](#) \*dsi1, [DatasetInstance](#) \*dsi2)  
*Compute the distance between two [DatasetInstances](#).*
- bool [PreComputeDistances](#) ()  
*Precompute all pairwise instance-to-instance distances.*
- bool [PreComputeDistancesByMap](#) ()  
*Precompute all pairwise distances homoring excluded instances.*
- std::vector< std::pair< double, std::string > > [GetScores](#) ()  
*Get the last computed [ReliefF](#) scores.*

## Protected Member Functions

- bool [ComputeWeightByDistanceFactors](#) ()  
*Compute the weight by distance factors for nearest neighbors.*

## Protected Attributes

- [AnalysisType](#) analysisType  
*type of analysis to perform*
- double(\* [snpDiff](#))(unsigned int attributeIndex, [DatasetInstance](#) \*dsi1, [DatasetInstance](#) \*dsi2)  
*Compute the discrete difference in an attribute between two instances.*

- `double(* numDiff)(unsigned int attributeIndex, DatasetInstance *dsi1, DatasetInstance *dsi2)`  
*Compute the continuous difference in an attribute between two instances.*
- `std::string snpMetric`  
*the name of discrete diff(ference) function*
- `std::string numMetric`  
*the name of continuous diff(ference) function*
- `Dataset * dataset`  
*the dataset on which the algorithm is working*
- `double one_over_m_times_k`  
*normalizing factor for ReliefF  $m * k$  loop*
- `unsigned int m`  
*number of instances to sample*
- `bool randomlySelect`  
*are instances being randomly selected?*
- `unsigned int k`  
*k nearest neighbors*
- `unsigned int removePerIteration`  
*number of attributes to remove each iteration if running iteratively*
- `bool doRemovePercent`  
*are we removing a percentage per iteration?*
- `double removePercentage`  
*percentage of attributes to remove per iteration if running iteratively*
- `std::string weightByDistanceMethod`  
*name of the weight-by-distance method*
- `double weightByDistanceSigma`  
*sigma value used in exponential decay weight-by-distance*
- `std::vector< double > W`  
*attribute scores/weights*
- `std::vector< std::string > scoreNames`  
*attribute names associated with scores*
- `std::map< std::string, double > finalScores`  
*final scores after all iterations*

### 6.18.1 Detailed Description

ReliefF attribute ranking algorithm.

Totally redone for the McKinney insilico lab in 2011. Large refactoring to move all attribute elimination handling to the Dataset and its subclasses. 9/11/11

#### See also

[RReliefF](#)

**Author**

Bill White

**Version**

1.0

Contact: [bill.c.white@gmail.com](mailto:bill.c.white@gmail.com) Created on: 7/16/05

Definition at line 46 of file ReliefF.h.

**6.18.2 Constructor & Destructor Documentation****6.18.2.1 ReliefF::ReliefF ( Dataset \* *ds*, AnalysisType *anaType* )**

Construct an [ReliefF](#) algorithm object.

**Parameters**

in	<i>ds</i>	pointer to a <a href="#">Dataset</a> object
in	<i>anaType</i>	analysis type

Definition at line 66 of file ReliefF.cpp.

**6.18.2.2 ReliefF::ReliefF ( Dataset \* *ds*, po::variables\_map & *vm*, AnalysisType *anaType* )**

Construct an [ReliefF](#) algorithm object.

**Parameters**

in	<i>ds</i>	pointer to a <a href="#">Dataset</a> object
in	<i>vm</i>	reference to a Boost map of command line options
in	<i>anaType</i>	analysis type

Definition at line 132 of file ReliefF.cpp.

**6.18.2.3 ReliefF::~~ReliefF ( )**

Definition at line 272 of file ReliefF.cpp.

**6.18.3 Member Function Documentation****6.18.3.1 bool ReliefF::ComputeAttributeScores ( ) [virtual]**

Compute the [ReliefF](#) scores for the current set of attributes.

Implements [ReliefF](#) algorithm: Marko Robnik-Sikonja, Igor Kononenko: Theoretical and

Empirical Analysis of [ReliefF](#) and [RReliefF](#). Machine Learning Journal, 53:23-69, 2003  
<http://lkm.fri.uni-lj.si/rmarko/papers/robnik03-mlj.pdf>

Reimplemented in [RReliefF](#).

Definition at line 275 of file ReliefF.cpp.

#### 6.18.3.2 bool ReliefF::ComputeAttributeScoresCleanSnps ( )

Compute the [ReliefF](#) scores for the current set of attributes 0/1/2 encoded.

Definition at line 451 of file ReliefF.cpp.

#### 6.18.3.3 bool ReliefF::ComputeAttributeScoresIteratively ( )

Compute the [ReliefF](#) scores by iteratively removing worst attributes.

Definition at line 572 of file ReliefF.cpp.

#### 6.18.3.4 double ReliefF::ComputeInstanceToInstanceDistance ( DatasetInstance \* *dsi1*, DatasetInstance \* *dsi2* )

Compute the distance between two DatasetInstances.

#### Parameters

in	<i>dsi1</i>	pointer to <a href="#">DatasetInstance</a> 1
in	<i>dsi2</i>	pointer to <a href="#">DatasetInstance</a> 2

#### Returns

distance

Definition at line 720 of file ReliefF.cpp.

#### 6.18.3.5 bool ReliefF::ComputeWeightByDistanceFactors ( ) [protected]

Compute the weight by distance factors for nearest neighbors.

Definition at line 1018 of file ReliefF.cpp.

#### 6.18.3.6 vector< pair< double, string > > ReliefF::GetScores ( )

Get the last computed [ReliefF](#) scores.

Definition at line 1004 of file ReliefF.cpp.

#### 6.18.3.7 bool ReliefF::PreComputeDistances ( )

Precompute all pairwise instance-to-instance distances.

Definition at line 748 of file ReliefF.cpp.

#### 6.18.3.8 bool ReliefF::PreComputeDistancesByMap ( )

Precompute all pairwise distances homoring excluded instances.

Definition at line 888 of file ReliefF.cpp.

#### 6.18.3.9 void ReliefF::PrintAttributeScores ( std::ofstream & *outStream* )

Write the scores and attribute names to stream.

##### Parameters

in	<i>outStream</i>	stream to write score-attribute name pairs
----	------------------	--

Definition at line 311 of file EvaporativeCooling.cpp.

#### 6.18.3.10 bool ReliefF::ProcessExclusionFile ( std::string *exclusionFilename* )

Remove file of attribute names from consideration in [ReliefF](#).

##### Parameters

in	<i>exclusion-Filename</i>	filename of attributes to exclude
----	---------------------------	-----------------------------------

##### Returns

success

Definition at line 694 of file ReliefF.cpp.

#### 6.18.3.11 bool ReliefF::ResetForNextIteration ( )

Resets some data structures for the next iteration of [ReliefF](#).

Definition at line 656 of file ReliefF.cpp.

#### 6.18.3.12 void ReliefF::WriteAttributeScores ( std::string *baseFilename* )

Write the scores and attribute names to file.

##### Parameters

in	<i>baseFilename</i>	filename to write score-attribute name pairs
----	---------------------	--

Definition at line 320 of file EvaporativeCooling.cpp.

## 6.18.4 Member Data Documentation

### 6.18.4.1 **AnalysisType ReliefF::analysisType** [protected]

type of analysis to perform

Definition at line 111 of file ReliefF.h.

### 6.18.4.2 **Dataset\* ReliefF::dataset** [protected]

the dataset on which the algorithm is working

Definition at line 137 of file ReliefF.h.

### 6.18.4.3 **bool ReliefF::doRemovePercent** [protected]

are we removing a percentage per iteration?

Definition at line 149 of file ReliefF.h.

### 6.18.4.4 **std::map<std::string, double> ReliefF::finalScores** [protected]

final scores after all iterations

Definition at line 162 of file ReliefF.h.

### 6.18.4.5 **unsigned int ReliefF::k** [protected]

k nearest neighbors

Definition at line 145 of file ReliefF.h.

### 6.18.4.6 **unsigned int ReliefF::m** [protected]

number of instances to sample

Definition at line 141 of file ReliefF.h.

### 6.18.4.7 **double(\* ReliefF::numDiff)(unsigned int attributeIndex, DatasetInstance \*dsi1, DatasetInstance \*dsi2)** [protected]

Compute the continuous difference in an attribute between two instances.

## Parameters

---



in	<i>attributeIndex</i>	index into vector of all attributes
in	<i>dsi1</i>	pointer to <a href="#">DatasetInstance</a> 1
in	<i>dsi2</i>	pointer to <a href="#">DatasetInstance</a> 2

**Returns**

diff(erence)

Definition at line 129 of file ReliefF.h.

**6.18.4.8 std::string ReliefF::numMetric** [protected]

the name of continuous diff(erence) function

Definition at line 135 of file ReliefF.h.

**6.18.4.9 double ReliefF::one\_over\_m\_times\_k** [protected]

normalizing factor for [ReliefF](#) m \* k loop

Definition at line 139 of file ReliefF.h.

**6.18.4.10 bool ReliefF::randomlySelect** [protected]

are instances being randomly selected?

Definition at line 143 of file ReliefF.h.

**6.18.4.11 double ReliefF::removePercentage** [protected]

percentage of attributes to remove per iteration if running iteratively

Definition at line 151 of file ReliefF.h.

**6.18.4.12 unsigned int ReliefF::removePerIteration** [protected]

number of attributes to remove each iteration if running iteratively

Definition at line 147 of file ReliefF.h.

**6.18.4.13 std::vector<std::string> ReliefF::scoreNames** [protected]

attribute names associated with scores

Definition at line 160 of file ReliefF.h.

**6.18.4.14** `double(* ReliefF::snpDiff)(unsigned int attributeIndex, DatasetInstance *dsi1, DatasetInstance *dsi2)` [protected]

Compute the discrete difference in an attribute between two instances.

#### Parameters

in	<i>attributeIndex</i>	index into vector of all attributes
in	<i>dsi1</i>	pointer to <a href="#">DatasetInstance</a> 1
in	<i>dsi2</i>	pointer to <a href="#">DatasetInstance</a> 2

#### Returns

diff(erence)

Definition at line 119 of file ReliefF.h.

**6.18.4.15** `std::string ReliefF::snpMetric` [protected]

the name of discrete diff(erence) function

Definition at line 133 of file ReliefF.h.

**6.18.4.16** `std::vector<double> ReliefF::W` [protected]

attribute scores/weights

Definition at line 158 of file ReliefF.h.

**6.18.4.17** `std::string ReliefF::weightByDistanceMethod` [protected]

name of the weight-by-distance method

Definition at line 153 of file ReliefF.h.

**6.18.4.18** `double ReliefF::weightByDistanceSigma` [protected]

sigma value used in exponential decay weight-by-distance

Definition at line 155 of file ReliefF.h.

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

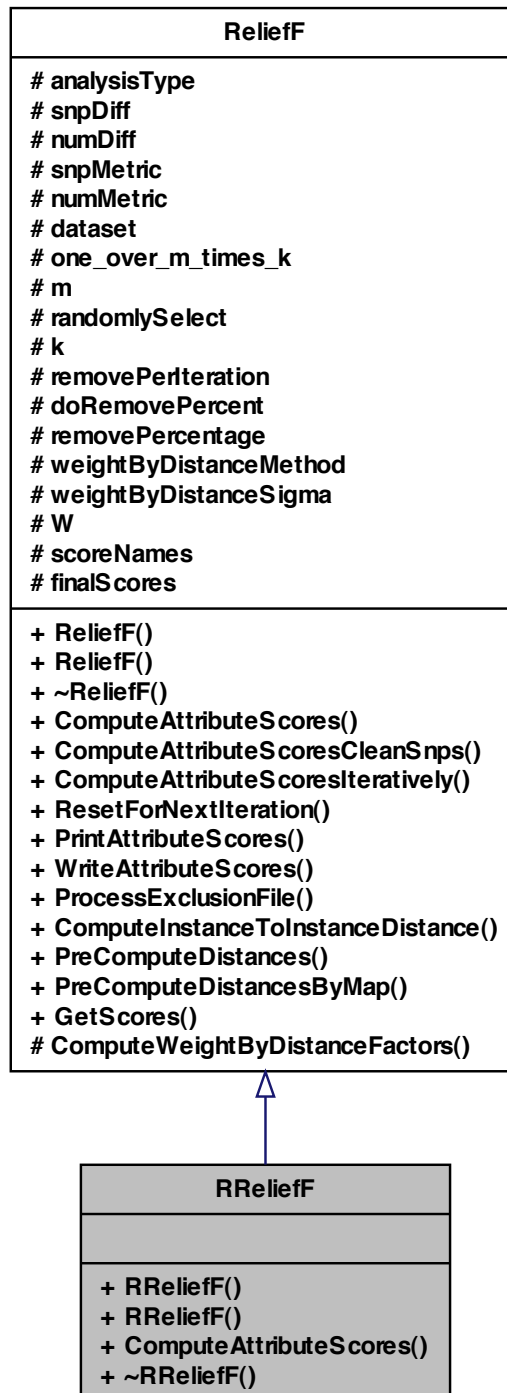
- src/library/[ReliefF.h](#)
- src/library/[EvaporativeCooling.cpp](#)
- src/library/[ReliefF.cpp](#)

## 6.19 RReliefF Class Reference

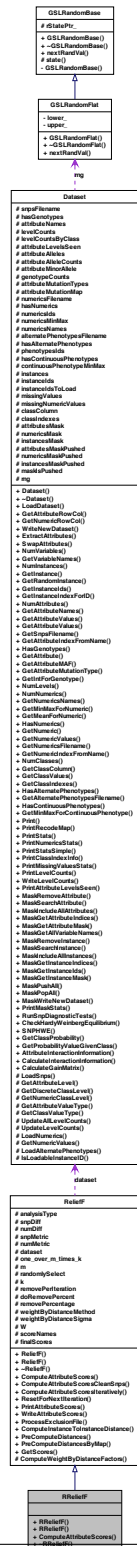
Regression [ReliefF](#) attribute ranking algorithm.

```
#include <RReliefF.h>
```

Inheritance diagram for RReliefF:



Generated on Fri Jan 6 2012 19:38:13 for Evaporative Cooling by Doxygen



## Public Member Functions

- [RReliefF](#) ([Dataset](#) \*ds)  
*Construct an [ReliefF](#) algorithm object.*
- [RReliefF](#) ([Dataset](#) \*ds, po::variables\_map &vm)  
*Construct an [ReliefF](#) algorithm object.*
- bool [ComputeAttributeScores](#) ()  
*Compute the [ReliefF](#) scores for the current set of attributes.*
- virtual [~RReliefF](#) ()

### 6.19.1 Detailed Description

Regression [ReliefF](#) attribute ranking algorithm.

Totally redone for the McKinney insilico lab in 2011. Large refactoring to move all attribute elimination handling to the [Dataset](#) and its subclasses. 9/11/11

#### See also

[ReliefF](#)

#### Author

Bill White

#### Version

1.0

Contact: [bill.c.white@gmail.com](mailto:bill.c.white@gmail.com) Created on: 9/27/11

Definition at line 32 of file RReliefF.h.

### 6.19.2 Constructor & Destructor Documentation

#### 6.19.2.1 RReliefF::RReliefF ( [Dataset](#) \* ds )

Construct an [ReliefF](#) algorithm object.

#### Parameters

in	ds	pointer to a <a href="#">Dataset</a> object
----	----	---

Definition at line 19 of file RReliefF.cpp.

#### 6.19.2.2 RReliefF::RReliefF ( [Dataset](#) \* ds, po::variables\_map & vm )

Construct an [ReliefF](#) algorithm object.

**Parameters**

in	<i>ds</i>	pointer to a <a href="#">Dataset</a> object
in	<i>vm</i>	reference to a Boost map of command line options

Definition at line 29 of file RReliefF.cpp.

**6.19.2.3 RReliefF::~RReliefF ( ) [virtual]**

Definition at line 39 of file RReliefF.cpp.

**6.19.3 Member Function Documentation****6.19.3.1 bool RReliefF::ComputeAttributeScores ( ) [virtual]**

Compute the [ReliefF](#) scores for the current set of attributes.

Implements [ReliefF](#) algorithm: Marko Robnik-Sikonja, Igor Kononenko: Theoretical and Empirical Analysis of [ReliefF](#) and [RReliefF](#). Machine Learning Journal, 53:23-69, 2003  
<http://lkm.fri.uni-lj.si/rmarko/papers/robnik03-mlj.pdf>

Used to hold the probability of a different class val given nearest instances (numeric class)

Used to hold the prob of different value of an attribute given nearest instances (numeric class case)

Used to hold the prob of a different class val and different att val given nearest instances (numeric class case)

Reimplemented from [ReliefF](#).

Definition at line 42 of file RReliefF.cpp.

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

- src/library/[RReliefF.h](#)
- src/library/[RReliefF.cpp](#)





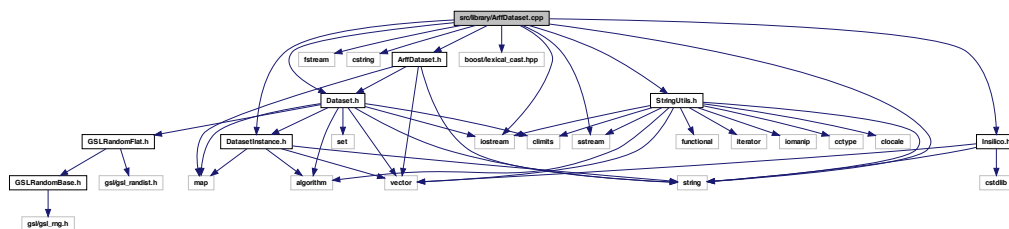
## Chapter 7

## File Documentation

## 7.1 src/library/ArffDataset.cpp File Reference

```
#include <string>
#include <iostream>
#include <fstream>
#include <cstring>
#include <sstream>
#include <boost/lexical_cast.hpp>
#include "Dataset.h"
#include "DatasetInstance.h"
#include "StringUtils.h"
#include "ArffDataset.h"
#include "Insilico.h"
```

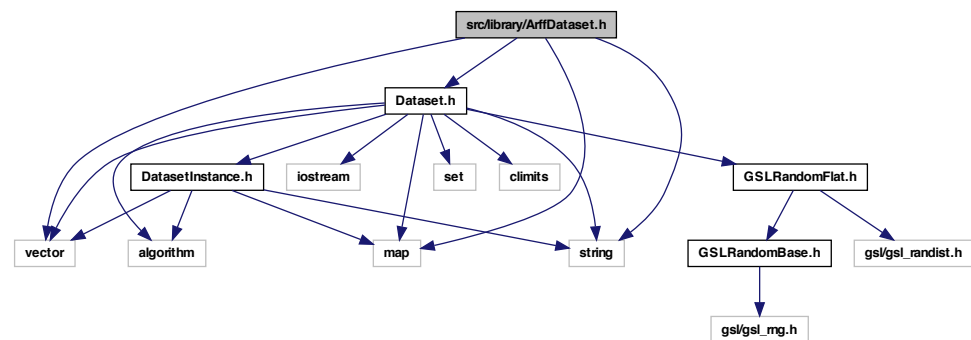
Include dependency graph for ArffDataset.cpp:



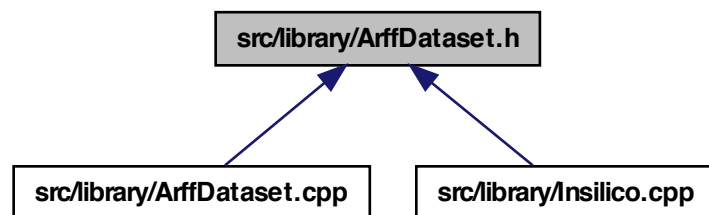
## 7.2 src/library/ArffDataset.h File Reference

```
#include <vector>
#include <map>
#include <string>
#include "Dataset.h"
```

Include dependency graph for ArffDataset.h:



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



### Classes

- class [ArffDataset](#)  
*ARFF file format reader.*

## Enumerations

- enum [ArffAttributeType](#) {  
  
    [ARFF\\_NUMERIC\\_TYPE](#), [ARFF\\_NOMINAL\\_TYPE](#), [ARFF\\_STRING\\_TYPE](#), [ARFF\\_DATE\\_TYPE](#),  
  
    [ARFF\\_ERROR\\_TYPE](#) }

### 7.2.1 Enumeration Type Documentation

#### 7.2.1.1 enum [ArffAttributeType](#)

ARFF attribute types.

##### Enumerator:

***ARFF\_NUMERIC\_TYPE*** continuous levels

***ARFF\_NOMINAL\_TYPE*** discrete levels

***ARFF\_STRING\_TYPE*** string levels

***ARFF\_DATE\_TYPE*** date levels

***ARFF\_ERROR\_TYPE*** unknown type

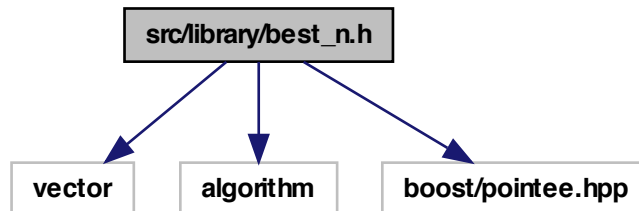
Definition at line 29 of file [ArffDataset.h](#).

## 7.3 src/library/best\_n.h File Reference

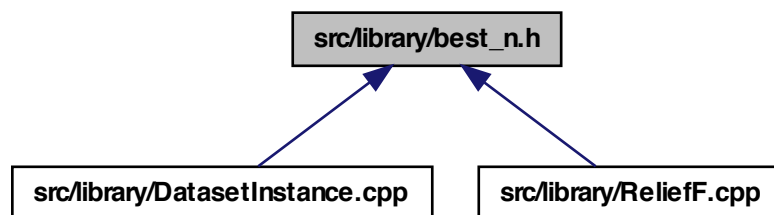
Find the best n keeping original order for ties - stable sort.

```
#include <vector>
#include <algorithm>
#include <boost/pointee.hpp>
```

Include dependency graph for `best_n.h`:



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



## Namespaces

- namespace `insilico`

## Functions

- `template<typename InputIt, typename OutputIt, typename Comp >`  
`void insilico::best_n(InputIt begin, InputIt end, OutputIt out, size_t n, Comp comp)`

*Get the best  $n$  values with ties keeping same original order.*

### 7.3.1 Detailed Description

Find the best n keeping original order for ties - stable sort.

#### Author

Nate Barney

#### Version

1.0

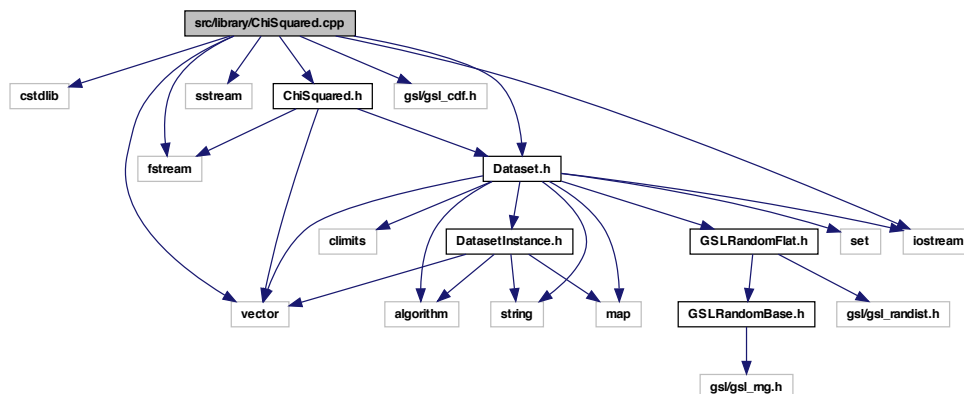
Contact: [bill.c.white@gmail.com](mailto:bill.c.white@gmail.com) Created on: 4/7/04

Definition in file [best\\_n.h](#).

## 7.4 src/library/ChiSquared.cpp File Reference

```
#include <cstdlib>
#include <iostream>
#include <fstream>
#include <sstream>
#include <vector>
#include "gsl/gsl_cdf.h"
#include "ChiSquared.h"
#include "Dataset.h"
```

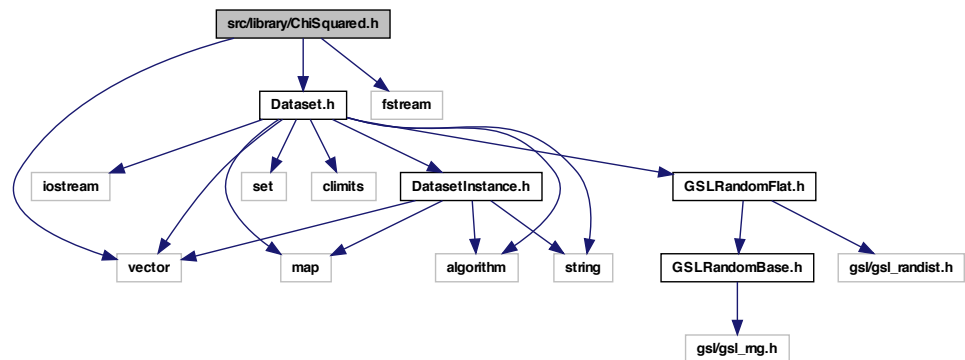
Include dependency graph for ChiSquared.cpp:



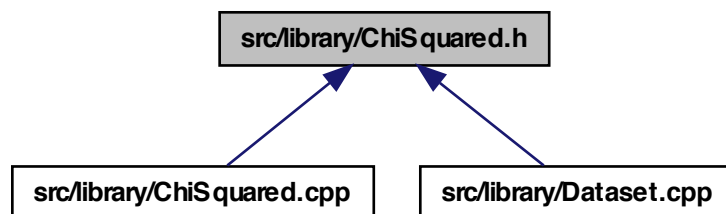
## 7.5 src/library/ChiSquared.h File Reference

```
#include <vector>
#include <fstream>
#include "Dataset.h"
```

Include dependency graph for ChiSquared.h:



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

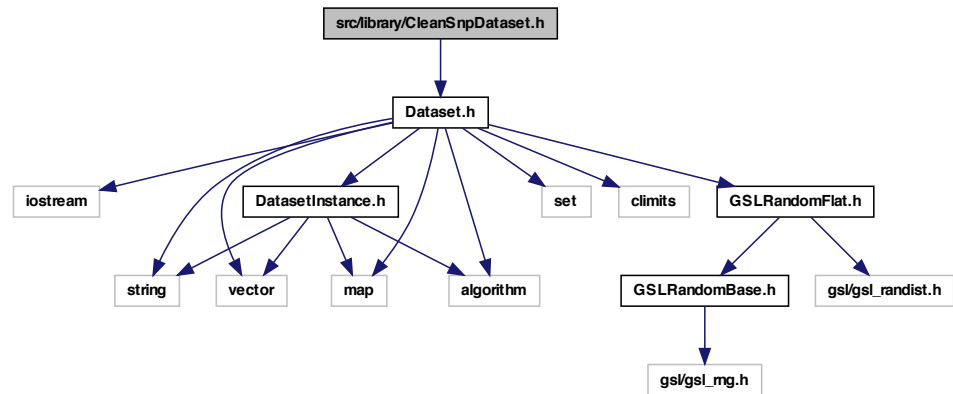


### Classes

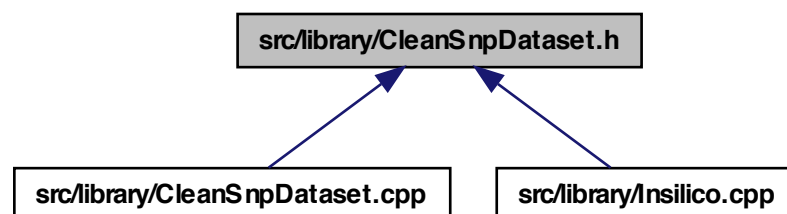
- class [ChiSquared](#)  
*Chi-squared attribute ranking algorithm.*



Include dependency graph for CleanSnpDataset.h:



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



## Classes

- class [CleanSnpDataset](#)  
*Experiemnetal data set reader for large/GWAS data.*

## 7.8 src/library/Dataset.cpp File Reference

```
#include <iostream>
```

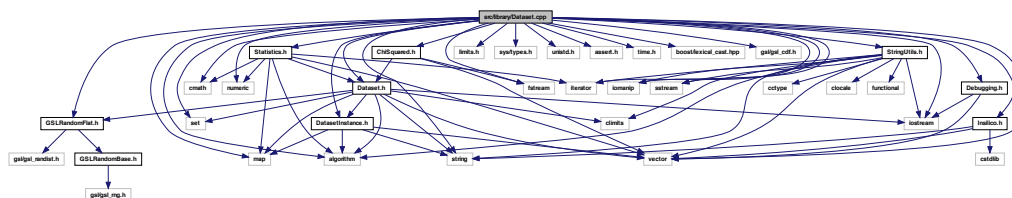


```

#include <iomanip>
#include <fstream>
#include <string>
#include <vector>
#include <set>
#include <map>
#include <iterator>
#include <cmath>
#include <algorithm>
#include <numeric>
#include <sstream>
#include <limits.h>
#include <sys/types.h>
#include <unistd.h>
#include <assert.h>
#include <time.h>
#include <boost/lexical_cast.hpp>
#include "gsl/gsl_cdf.h"
#include "GSLRandomFlat.h"
#include "ChiSquared.h"
#include "Dataset.h"
#include "DatasetInstance.h"
#include "StringUtils.h"
#include "Statistics.h"
#include "Debugging.h"
#include "Insilico.h"

```

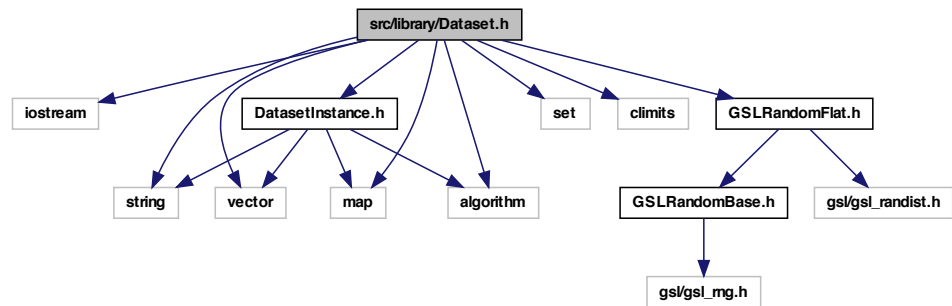
Include dependency graph for Dataset.cpp:



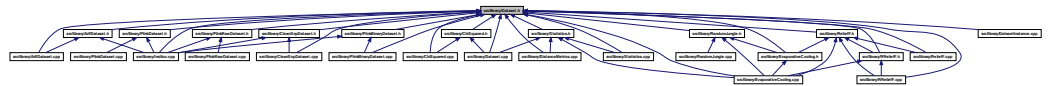
## 7.9 src/library/Dataset.h File Reference

```
#include <iostream>
#include <string>
#include <vector>
#include <map>
#include <set>
#include <algorithm>
#include <climits>
#include "DatasetInstance.h"
#include "GSLRandomFlat.h"
```

Include dependency graph for Dataset.h:



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



## Classes

- class Dataset

*Base class for collections of instances containing attributea and class.*

## Enumerations

- enum `ValueType` { `NUMERIC_VALUE`, `DISCRETE_VALUE`, `MISSING_VALUE`, `NO_VALUE` }
- enum `AttributeType` { `NUMERIC_TYPE`, `DISCRETE_TYPE`, `NO_TYPE` }
- enum `AttributeMutationType` { `TRANSITION_MUTATION`, `TRANSVERSION_MUTATION`, `UNKNOWN_MUTATION` }
- enum `OutputDatasetType` { `TAB_DELIMITED_DATASET`, `CSV_DELIMITED_DATASET`, `ARFF_DATASET`, `NO_OUTPUT_DATASET` }

## Variables

- static const int `INVALID_DISTANCE` = `INT_MAX`  
*return value for invalid distance*
- static const int `INVALID_INDEX` = `INT_MAX`  
*return value for invalid index into attributes*
- static const `AttributeLevel` `INVALID_ATTRIBUTE_VALUE` = `INT_MIN`  
*invalid attribute value*
- static const `NumericLevel` `INVALID_NUMERIC_VALUE` = `INT_MIN`  
*invalid attribute value*
- static const `ClassLevel` `INVALID_DISCRETE_CLASS_VALUE` = `INT_MIN`  
*stored value for missing discrete class*
- static const `NumericLevel` `INVALID_NUMERIC_CLASS_VALUE` = `INT_MIN`  
*stored value for missing numeric class*
- static const `AttributeLevel` `MISSING_ATTRIBUTE_VALUE` = -9  
*stored value for missing discrete attribute*
- static const `NumericLevel` `MISSING_NUMERIC_VALUE` = -9  
*stored value for missing numeric attribute*
- static const `ClassLevel` `MISSING_DISCRETE_CLASS_VALUE` = -9  
*stored value for missing discrete class*
- static const `NumericLevel` `MISSING_NUMERIC_CLASS_VALUE` = -9  
*stored value for missing numeric class*

### 7.9.1 Enumeration Type Documentation

#### 7.9.1.1 enum `AttributeMutationType`

Type of attribute mutation.

**Enumerator:**

**TRANSITION\_MUTATION** transition within family  
**TRANSVERSION\_MUTATION** transversion between families  
**UNKNOWN\_MUTATION** unknown - no allele information

Definition at line 82 of file Dataset.h.

**7.9.1.2 enum AttributeType**

Type of attributes that are stored in data set instances.

**Enumerator:**

**NUMERIC\_TYPE** continuous numeric type  
**DISCRETE\_TYPE** discrete genotype type  
**NO\_TYPE** default no type

Definition at line 71 of file Dataset.h.

**7.9.1.3 enum OutputDatasetType**

Type of data set to write filtered output.

**Enumerator:**

**TAB\_DELIMITED\_DATASET** tab-delimited .txt file  
**CSV\_DELIMITED\_DATASET** comma separated values .csv file  
**ARFF\_DATASET** WEKA ARFF format .arff file.  
**NO\_OUTPUT\_DATASET** no output data set specified

Definition at line 93 of file Dataset.h.

**7.9.1.4 enum ValueType**

Return types for determining a value's type.

**Enumerator:**

**NUMERIC\_VALUE** continuous numeric value  
**DISCRETE\_VALUE** discrete genotype value  
**MISSING\_VALUE** missing value  
**NO\_VALUE** default no value type

Definition at line 59 of file Dataset.h.

## 7.9.2 Variable Documentation

**7.9.2.1** `const AttributeLevel INVALID_ATTRIBUTE_VALUE = INT_MIN` `[static]`

invalid attribute value

Definition at line 38 of file Dataset.h.

**7.9.2.2** `const ClassLevel INVALID_DISCRETE_CLASS_VALUE = INT_MIN`  
`[static]`

stored value for missing discrete class

Definition at line 42 of file Dataset.h.

**7.9.2.3** `const int INVALID_DISTANCE = INT_MAX` `[static]`

return value for invalid distance

Definition at line 33 of file Dataset.h.

**7.9.2.4** `const int INVALID_INDEX = INT_MAX` `[static]`

return value for invalid index into attributes

Definition at line 35 of file Dataset.h.

**7.9.2.5** `const NumericLevel INVALID_NUMERIC_CLASS_VALUE = INT_MIN`  
`[static]`

stored value for missing numeric class

Definition at line 44 of file Dataset.h.

**7.9.2.6** `const NumericLevel INVALID_NUMERIC_VALUE = INT_MIN` `[static]`

invalid attribute value

Definition at line 40 of file Dataset.h.

**7.9.2.7** `const AttributeLevel MISSING_ATTRIBUTE_VALUE = -9` `[static]`

stored value for missing discrete attribute

Definition at line 47 of file Dataset.h.

### 7.9.2.8 `const ClassLevel MISSING_DISCRETE_CLASS_VALUE = -9` [static]

stored value for missing discrete class

Definition at line 51 of file Dataset.h.

### 7.9.2.9 `const NumericLevel MISSING_NUMERIC_CLASS_VALUE = -9` [static]

stored value for missing numeric class

Definition at line 53 of file Dataset.h.

### 7.9.2.10 `const NumericLevel MISSING_NUMERIC_VALUE = -9` [static]

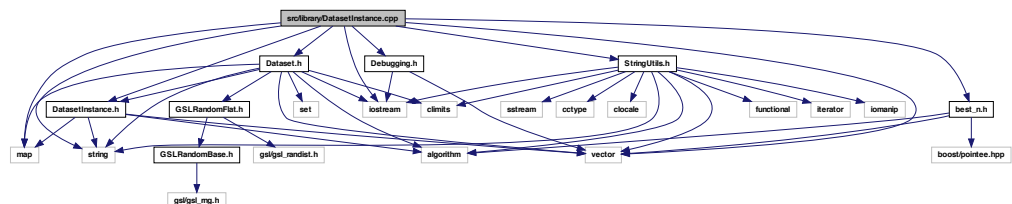
stored value for missing numeric attribute

Definition at line 49 of file Dataset.h.

## 7.10 `src/library/DatasetInstance.cpp` File Reference

```
#include <iostream>
#include <string>
#include <vector>
#include <map>
#include "Dataset.h"
#include "DatasetInstance.h"
#include "StringUtils.h"
#include "best_n.h"
#include "Debugging.h"
```

Include dependency graph for DatasetInstance.cpp:



## Classes

- class [deref\\_less\\_bcw](#)

## Typedefs

- typedef [DistancePair](#) *T*  
*functor for T comparison*

### 7.10.1 Typedef Documentation

#### 7.10.1.1 typedef [DistancePair](#) *T*

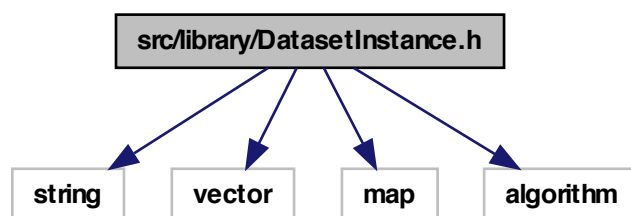
functor for *T* comparison

Definition at line 23 of file DatasetInstance.cpp.

## 7.11 src/library/DatasetInstance.h File Reference

```
#include <string>
#include <vector>
#include <map>
#include <algorithm>
```

Include dependency graph for DatasetInstance.h:



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



## Classes

- class [DatasetInstance](#)  
*Class to hold dataset instances (rows of attributes).*

## Typedefs

- typedef int [AttributeLevel](#)  
*type of discrete attribute values*
- typedef double [NumericLevel](#)  
*type of continuous attributes*
- typedef int [ClassLevel](#)  
*type of instance class labels*
- typedef std::pair< double, std::string > [DistancePair](#)  
*distance pair type: distance, instance ID*
- typedef std::vector< [DistancePair](#) > [DistancePairs](#)  
*vector of distance pairs represents distances to nearest neighbors*
- typedef DistancePairs::const\_iterator [DistancePairsIt](#)  
*distance pairs iterator*

### 7.11.1 Typedef Documentation

#### 7.11.1.1 typedef int AttributeLevel

type of discrete attribute values

Definition at line 24 of file DatasetInstance.h.

#### 7.11.1.2 typedef int ClassLevel

type of instance class labels

Definition at line 28 of file DatasetInstance.h.



### 7.11.1.3 `typedef std::pair<double, std::string> DistancePair`

distance pair type: distance, instance ID

Definition at line 31 of file DatasetInstance.h.

### 7.11.1.4 `typedef std::vector<DistancePair> DistancePairs`

vector of distance pairs represents distances to nearest neighbors

Definition at line 33 of file DatasetInstance.h.

### 7.11.1.5 `typedef DistancePairs::const_iterator DistancePairsIt`

distance pairs iterator

Definition at line 35 of file DatasetInstance.h.

### 7.11.1.6 `typedef double NumericLevel`

type of continuous attributes

Definition at line 26 of file DatasetInstance.h.

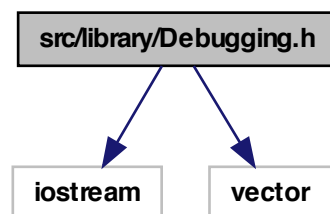
## 7.12 src/library/Debugging.h File Reference

Debugging utilities.

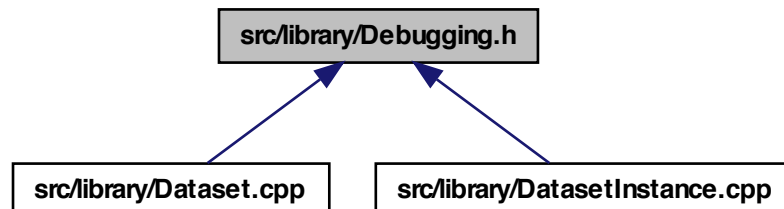
```
#include <iostream>
```

```
#include <vector>
```

Include dependency graph for Debugging.h:



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



## Functions

- `template<class T >`  
`void PrintVector (std::vector< T > vec, std::string title="")`  
*Print a vector of T values with optional title.*
- `template<class T >`  
`void PrintVector (vector< T > vec, string title)`

### 7.12.1 Detailed Description

Debugging utilities.

#### Author

Bill White

#### Version

1.0

Contact: [bill.c.white@gmail.com](mailto:bill.c.white@gmail.com) Created on: 8/q/11

Definition in file [Debugging.h](#).

### 7.12.2 Function Documentation

7.12.2.1 `template<class T > void PrintVector ( std::vector< T > vec, std::string title = " " )`

Print a vector of T values with optional title.

#### Parameters

in	<i>vec</i>	vector of T type values
in	<i>title</i>	optional title to print before the vector

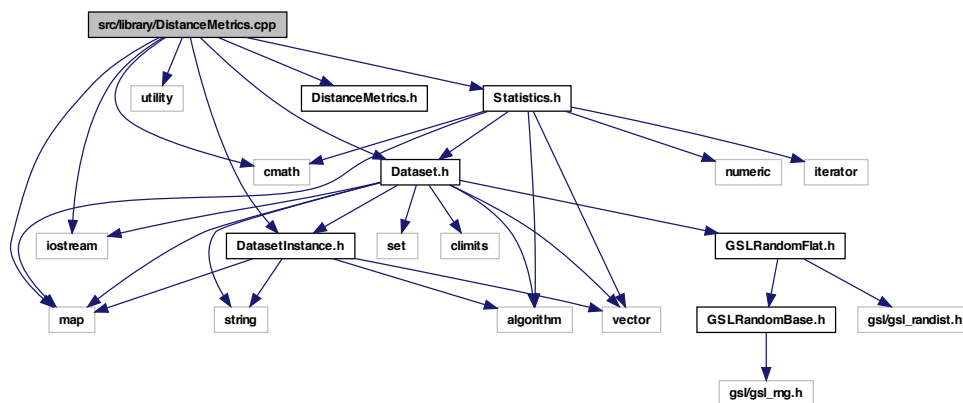
7.12.2.2 `template<class T> void PrintVector ( vector< T > vec, string title )`

Definition at line 28 of file Debugging.h.

## 7.13 src/library/DistanceMetrics.cpp File Reference

```
#include <cmath>
#include <iostream>
#include <map>
#include <utility>
#include "Dataset.h"
#include "DistanceMetrics.h"
#include "DatasetInstance.h"
#include "Statistics.h"
```

Include dependency graph for DistanceMetrics.cpp:



## Functions

- `pair< bool, double > CheckMissing (unsigned int attributeIndex, DatasetInstance *dsi1, DatasetInstance *dsi2)`

*Check for a missing discrete value and return value.*

- `pair< bool, double > CheckMissingNumeric` (unsigned int numericIndex, [DatasetInstance \\*dsi1](#), [DatasetInstance \\*dsi2](#))

*Check for a missing continuous value and return value.*

- double `norm` (double x, double minX, double maxX)

*Normalizes a given value of a numeric attribute.*

- double `diffAMM` (unsigned int attributeIndex, [DatasetInstance \\*dsi1](#), [DatasetInstance \\*dsi2](#))

*Allele mismatch metric.*

- double `diffGMM` (unsigned int attributeIndex, [DatasetInstance \\*dsi1](#), [DatasetInstance \\*dsi2](#))

*Genotype mismatch metric.*

- double `diffManhattan` (unsigned int attributeIndex, [DatasetInstance \\*dsi1](#), [DatasetInstance \\*dsi2](#))

*"Manhattan" distance between continuous attributes.*

- double `diffPredictedValueTau` ([DatasetInstance \\*dsi1](#), [DatasetInstance \\*dsi2](#))

*Same as "Manhattan" distance but uses method calls versus public variables.*

### 7.13.1 Function Documentation

**7.13.1.1** `pair<bool, double> CheckMissing ( unsigned int attributeIndex, DatasetInstance * dsi1, DatasetInstance * dsi2 )`

Check for a missing discrete value and return value.

#### Parameters

in	<i>attributeIndex</i>	index into the vector of attributes
in	<i>dsi1</i>	data set instance 1
in	<i>dsi2</i>	data set instance 2

#### Returns

pair: has missing value?, value for missing (true) or 0.0 (false)

Definition at line 21 of file DistanceMetrics.cpp.

**7.13.1.2** `pair<bool, double> CheckMissingNumeric ( unsigned int numericIndex, DatasetInstance * dsi1, DatasetInstance * dsi2 )`

Check for a missing continuous value and return value.

#### Parameters

in	<i>attributeIndex</i>	index into the vector of attributes
----	-----------------------	-------------------------------------

in	<i>dsi1</i>	data set instance 1
in	<i>dsi2</i>	data set instance 2

**Returns**

pair: has missing value?, value for missing (true) or 0.0 (false)

Definition at line 88 of file DistanceMetrics.cpp.

**7.13.1.3 double diffAMM ( unsigned int *attributeIndex*, DatasetInstance \* *dsi1*, DatasetInstance \* *dsi2* )**

Allele mismatch metric.

**Parameters**

in	<i>attributeIndex</i>	index into the vector of attributes
in	<i>dsi1</i>	data set instance 1
in	<i>dsi2</i>	data set instance 2

**Returns**

diff(ference) between attribute values: 0.0, 0.5, 1.0

Definition at line 135 of file DistanceMetrics.cpp.

**7.13.1.4 double diffGMM ( unsigned int *attributeIndex*, DatasetInstance \* *dsi1*, DatasetInstance \* *dsi2* )**

Genotype mismatch metric.

**Parameters**

in	<i>attributeIndex</i>	index into the vector of attributes
in	<i>dsi1</i>	data set instance 1
in	<i>dsi2</i>	data set instance 2

**Returns**

diff(ference) between attribute values: 0.0 (same) or 1.0 (not same)

Definition at line 150 of file DistanceMetrics.cpp.

**7.13.1.5 double diffManhattan ( unsigned int *attributeIndex*, DatasetInstance \* *dsi1*, DatasetInstance \* *dsi2* )**

"Manhattan" distance between continuous attributes.

**Parameters**

in	<i>attributeIndex</i>	index into the vector of attributes
in	<i>dsi1</i>	data set instance 1
in	<i>dsi2</i>	data set instance 2

**Returns**

absolute value of difference divided by attribute's range

Definition at line 164 of file DistanceMetrics.cpp.

7.13.1.6 `double diffPredictedValueTau ( DatasetInstance * dsi1, DatasetInstance * dsi2 )`

Same as "Manhattan" distance but uses method calls versus public variables.

**Parameters**

in	<i>attributeIndex</i>	index into the vector of attributes
in	<i>dsi1</i>	data set instance 1
in	<i>dsi2</i>	data set instance 2

**Returns**

absolute value of difference divided by attribute's range

Definition at line 189 of file DistanceMetrics.cpp.

7.13.1.7 `double norm ( double x, double minX, double maxX )`

Normalizes a given value of a numeric attribute.

Borrowed from Weka 8/18/11

**Parameters**

in	<i>x</i>	value
in	<i>minX</i>	minimum value for x
in	<i>maxX</i>	maximum value for x

**Returns**

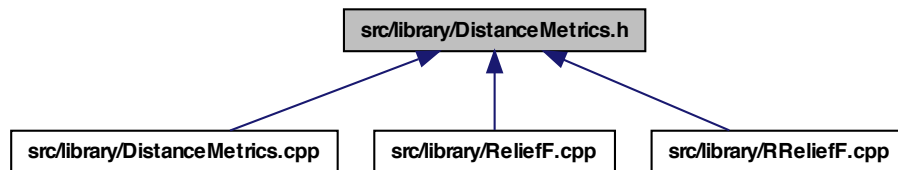
normalized value

Definition at line 127 of file DistanceMetrics.cpp.

## 7.14 src/library/DistanceMetrics.h File Reference

Distance metrics for [ReliefF](#).

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



### Functions

- `std::pair< bool, double >` [CheckMissing](#) (unsigned int attributeIndex, [DatasetInstance](#) \*dsi1, [DatasetInstance](#) \*dsi2)  
*Check for a missing discrete value and return value.*
- `std::pair< bool, double >` [CheckMissingNumeric](#) (unsigned int numericIndex, [DatasetInstance](#) \*dsi1, [DatasetInstance](#) \*dsi2)  
*Check for a missing continuous value and return value.*
- `double` [norm](#) (double x, double minX, double maxX)  
*Normalizes a given value of a numeric attribute.*
- `double` [diffAMM](#) (unsigned int attributeIndex, [DatasetInstance](#) \*dsi1, [DatasetInstance](#) \*dsi2)  
*Allele mismatch metric.*
- `double` [diffGMM](#) (unsigned int attributeIndex, [DatasetInstance](#) \*dsi1, [DatasetInstance](#) \*dsi2)  
*Genotype mismatch metric.*
- `double` [diffManhattan](#) (unsigned int attributeIndex, [DatasetInstance](#) \*dsi1, [DatasetInstance](#) \*dsi2)  
*"Manhattan" distance between continuous attributes.*
- `double` [diffPredictedValueTau](#) ([DatasetInstance](#) \*dsi1, [DatasetInstance](#) \*dsi2)  
*Same as "Manhattan" distance but uses method calls versus public variables.*

### 7.14.1 Detailed Description

Distance metrics for [ReliefF](#).

#### Author

: Bill White

**Version**

1.0

Contact: [bill.c.white@gmail.com](mailto:bill.c.white@gmail.com) Created on 3/29/11Definition in file [DistanceMetrics.h](#).**7.14.2 Function Documentation****7.14.2.1 `std::pair<bool, double> CheckMissing ( unsigned int attributeIndex, DatasetInstance * dsi1, DatasetInstance * dsi2 )`**

Check for a missing discrete value and return value.

**Parameters**

in	<i>attributeIndex</i>	index into the vector of attributes
in	<i>dsi1</i>	data set instance 1
in	<i>dsi2</i>	data set instance 2

**Returns**

pair: has missing value?, value for missing (true) or 0.0 (false)

Definition at line 21 of file DistanceMetrics.cpp.

**7.14.2.2 `std::pair<bool, double> CheckMissingNumeric ( unsigned int numericIndex, DatasetInstance * dsi1, DatasetInstance * dsi2 )`**

Check for a missing continuous value and return value.

**Parameters**

in	<i>attributeIndex</i>	index into the vector of attributes
in	<i>dsi1</i>	data set instance 1
in	<i>dsi2</i>	data set instance 2

**Returns**

pair: has missing value?, value for missing (true) or 0.0 (false)

Definition at line 88 of file DistanceMetrics.cpp.

**7.14.2.3 `double diffAMM ( unsigned int attributeIndex, DatasetInstance * dsi1, DatasetInstance * dsi2 )`**

Allele mismatch metric.



**Parameters**

in	<i>attributeIndex</i>	index into the vector of attributes
in	<i>dsi1</i>	data set instance 1
in	<i>dsi2</i>	data set instance 2

**Returns**

diff(ference) between attribute values: 0.0, 0.5, 1.0

Definition at line 135 of file DistanceMetrics.cpp.

**7.14.2.4** double diffGMM ( unsigned int *attributeIndex*, DatasetInstance \* *dsi1*, DatasetInstance \* *dsi2* )

Genotype mismatch metric.

**Parameters**

in	<i>attributeIndex</i>	index into the vector of attributes
in	<i>dsi1</i>	data set instance 1
in	<i>dsi2</i>	data set instance 2

**Returns**

diff(ference) between attribute values: 0.0 (same) or 1.0 (not same)

Definition at line 150 of file DistanceMetrics.cpp.

**7.14.2.5** double diffManhattan ( unsigned int *attributeIndex*, DatasetInstance \* *dsi1*, DatasetInstance \* *dsi2* )

"Manhattan" distance between continuous attributes.

**Parameters**

in	<i>attributeIndex</i>	index into the vector of attributes
in	<i>dsi1</i>	data set instance 1
in	<i>dsi2</i>	data set instance 2

**Returns**

absolute value of difference divided by attribute's range

Definition at line 164 of file DistanceMetrics.cpp.

#### 7.14.2.6 `double diffPredictedValueTau ( DatasetInstance * dsi1, DatasetInstance * dsi2 )`

Same as "Manhattan" distance but uses method calls versus public variables.

##### Parameters

in	<i>attributeIndex</i>	index into the vector of attributes
in	<i>dsi1</i>	data set instance 1
in	<i>dsi2</i>	data set instance 2

##### Returns

absolute value of difference divided by attribute's range

Definition at line 189 of file DistanceMetrics.cpp.

#### 7.14.2.7 `double norm ( double x, double minX, double maxX )`

Normalizes a given value of a numeric attribute.

Borrowed from Weka 8/18/11

##### Parameters

in	<i>x</i>	value
in	<i>minX</i>	minimum value for x
in	<i>maxX</i>	maximum value for x

##### Returns

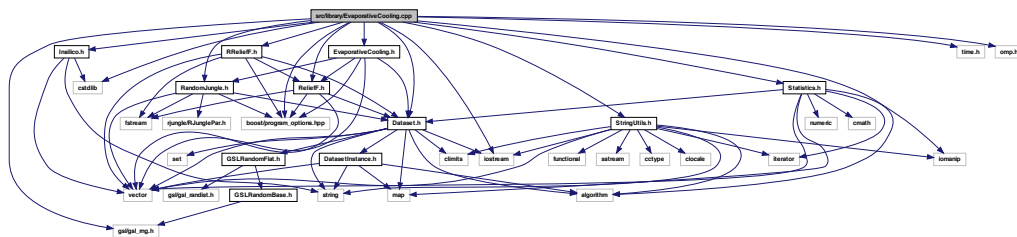
normalized value

Definition at line 127 of file DistanceMetrics.cpp.

## 7.15 `src/library/EvaporativeCooling.cpp` File Reference

```
#include <cstdlib>
#include <iostream>
#include <iomanip>
#include <time.h>
#include <boost/program_options.hpp>
#include <omp.h>
#include <gsl/gsl_rng.h>
#include "EvaporativeCooling.h"
#include "Dataset.h"
```

Include dependency graph for EvaporativeCooling.cpp:



- bool `scoresSortAsc` (const pair< double, string > &p1, const pair< double, string > &p2)
- bool `scoresSortAscByName` (const pair< double, string > &p1, const pair< double, string > &p2)
- bool `scoresSortDesc` (const pair< double, string > &p1, const pair< double, string > &p2)

```
7.15.1.1 bool scoresSortAsc ( const pair< double, string > & p1, const pair< double, string >
& p2 )
```

```
7.15.1.2 bool scoresSortAscByName ( const pair< double, string > & p1, const pair< double,
string > & p2 )
```

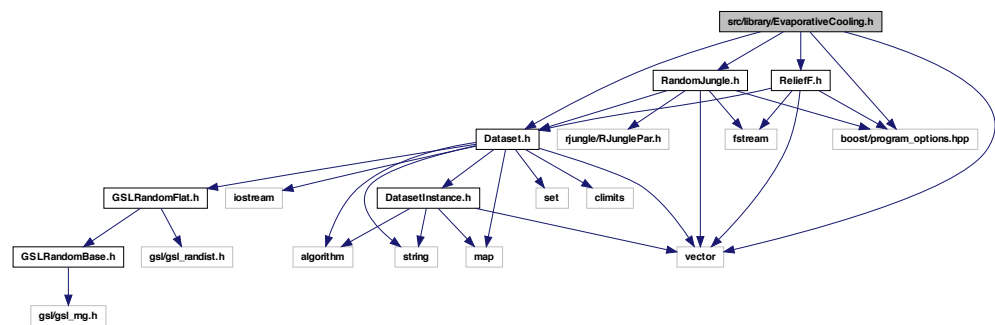
```
7.15.1.3 bool scoresSortDesc ( const pair< double, string > & p1, const pair< double, string
> & p2 )
```

Generated on Fri Jan 6 2012 19:38:13 for Evaporative Cooling by Doxygen

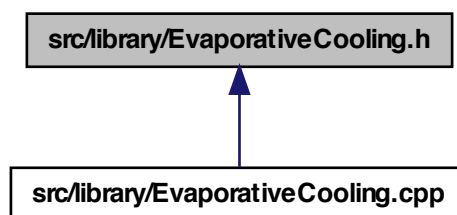
## 7.16 src/library/EvaporativeCooling.h File Reference

```
#include <vector>
#include <boost/program_options.hpp>
#include "Dataset.h"
#include "RandomJungle.h"
#include "ReliefF.h"
```

Include dependency graph for EvaporativeCooling.h:



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



### Classes

- class [EvaporativeCooling](#)

*Evaporative Cooling attribute ranking algorithm.*

## Typedefs

- typedef std::vector< std::pair< double, std::string > > [EcScores](#)  
*evaporative cooling scores - sorted by score key*
- typedef std::vector< std::pair< double, std::string > >::iterator [EcScoresIt](#)  
*evaporative cooling scores iterator - sorted by score key*
- typedef std::vector< std::pair< double, std::string > >::const\_iterator [EcScoresCIt](#)  
*evaporative cooling scores constant iterator - sorted by score key*

## Enumerations

- enum [EcAlgorithmType](#) { [EC\\_ALL](#), [EC\\_RJ](#), [EC\\_RF](#) }

### 7.16.1 Typedef Documentation

#### 7.16.1.1 typedef std::vector<std::pair<double, std::string> > [EcScores](#)

evaporative cooling scores - sorted by score key

Definition at line 35 of file EvaporativeCooling.h.

#### 7.16.1.2 typedef std::vector<std::pair<double, std::string> >::const\_iterator [EcScoresCIt](#)

evaporative cooling scores constant iterator - sorted by score key

Definition at line 39 of file EvaporativeCooling.h.

#### 7.16.1.3 typedef std::vector<std::pair<double, std::string> >::iterator [EcScoresIt](#)

evaporative cooling scores iterator - sorted by score key

Definition at line 37 of file EvaporativeCooling.h.

### 7.16.2 Enumeration Type Documentation

#### 7.16.2.1 enum [EcAlgorithmType](#)

Type of algorithm steps to perform.

##### Enumerator:

**[EC\\_ALL](#)** Run [RandomJungle](#) and [ReliefF](#).

**[EC\\_RJ](#)** Run only [RandomJungle](#).

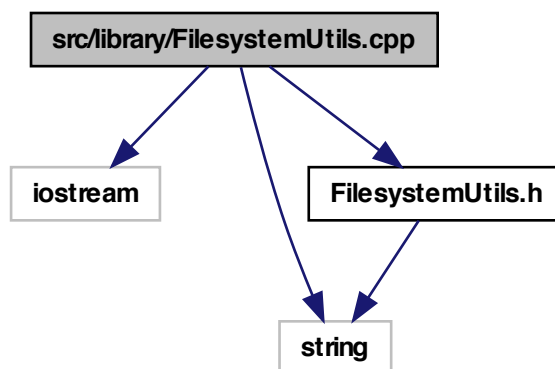
**[EC\\_RF](#)** Run only [ReliefF](#).

Definition at line 45 of file EvaporativeCooling.h.

## 7.17 src/library/FileSystemUtils.cpp File Reference

```
#include <iostream>
#include <string>
#include "FileSystemUtils.h"
```

Include dependency graph for FileSystemUtils.cpp:



### Functions

- string [GetFileBasename](#) (string fileName)
- string [GetFileExtension](#) (string fileName)

#### 7.17.1 Function Documentation

##### 7.17.1.1 string GetFileBasename ( string *fileName* )

Definition at line 8 of file FileSystemUtils.cpp.

##### 7.17.1.2 string GetFileExtension ( string *fileName* )

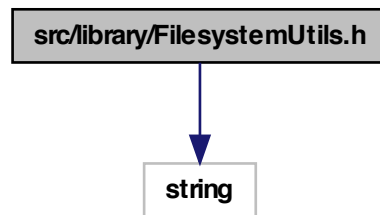
Definition at line 13 of file FileSystemUtils.cpp.

## 7.18 src/library/FileSystemUtils.h File Reference

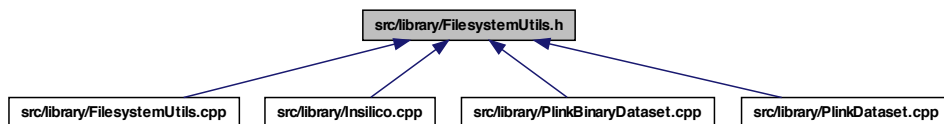
Filesystem utilities.

```
#include <string>
```

Include dependency graph for FileSystemUtils.h:



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



### Functions

- `std::string` [GetFileBasename](#) (`std::string` fullFilename)  
*Get the full filename without the extension.*
- `std::string` [GetFileExtension](#) (`std::string` fullFilename)  
*Get the filename extension.*

### 7.18.1 Detailed Description

Filesystem utilities.

#### Author

Bill White

**Version**

1.0

Contact: [bill.c.white@gmail.com](mailto:bill.c.white@gmail.com) Created on: 4/7/11Definition in file [FilesystemUtils.h](#).**7.18.2 Function Documentation****7.18.2.1 std::string GetFileBasename ( std::string *fullFilename* )**

Get the full filename without the extension.

**Parameters**

in	<i>fullFilename</i>	complete filename
----	---------------------	-------------------

**Returns**

path/filename without extension

**7.18.2.2 std::string GetFileExtension ( std::string *fullFilename* )**

Get the filename extension.

**Parameters**

in	<i>fullFilename</i>	complete filename
----	---------------------	-------------------

**Returns**

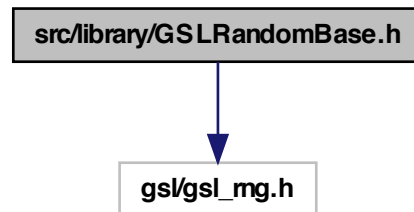
filename extension

**7.19 src/library/GSLRandomBase.h File Reference**

#include "gsl/gsl\_rng.h"



Include dependency graph for GSLRandomBase.h:



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



## Classes

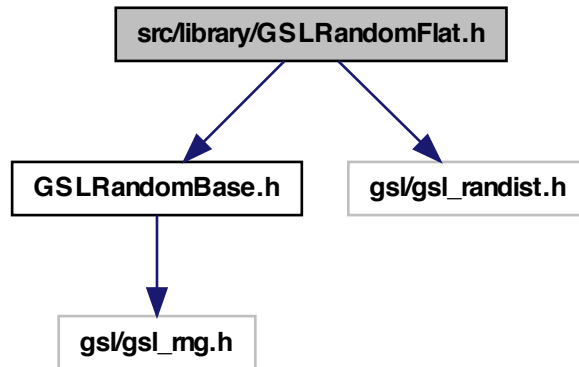
- class **GSLRandomBase**

*A base class for GNU Scientific Library (GSL) random number functions.*

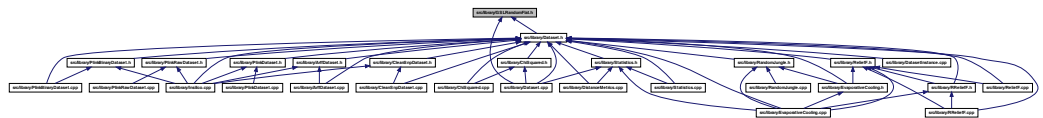
## 7.20 src/library/GSLRandomFlat.h File Reference

```
#include "GSLRandomBase.h"
#include "gsl/gsl_randist.h"
```

Include dependency graph for GSLRandomFlat.h:



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



## Classes

- class [GSLRandomFlat](#)  
*Random numbers in a flat, or uniform distribution.*

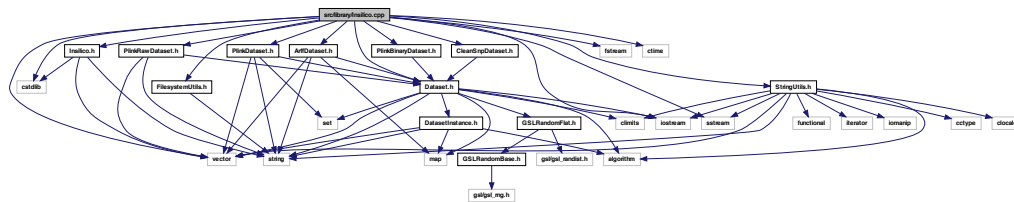
## 7.21 src/library/Insilico.cpp File Reference

```

#include <cstdlib>
#include <iostream>
#include <fstream>
#include <vector>
#include <sstream>
#include <ctime>

```

Include dependency graph for Insilico.cpp:



- string **Timestamp** ()  
*Return a timestamp string for logging purposes.*
- **Dataset \* ChooseSnpsDatasetByExtension** (string snpsFilename, bool isCleanSnps)
- bool **LoadIndividualIds** (string filename, vector< string > &retIds, bool hasHeader)
- bool **GetMatchingIds** (string numericsFilename, string altPhenotypeFilename, vector< string > numericsIds, vector< string > phenolds, vector< string > &matchingIds)

```
7.21.1.1 Dataset* ChooseSnpsDatasetByExtension ( string snpsFilename, bool isCleanSnps
)
```

```
7.21.1.2 bool GetMatchingIds ( string numericsFilename, string altPhenotypeFilename,
vector< string > numericsIds, vector< string > phenolds, vector< string > &
matchingIds )
```

Generated on Fri Jan 6 2012 19:38:13 for Evaporative Cooling by Doxygen

### 7.21.1.3 `bool LoadIndividualIds ( string filename, vector< string > & retIds, bool hasHeader )`

Definition at line 87 of file `Insilico.cpp`.

### 7.21.1.4 `string Timestamp ( )`

Return a timestamp string for logging purposes.

#### Returns

fixed-length, formatted timestamp as a string

Definition at line 31 of file `Insilico.cpp`.

## 7.22 `src/library/Insilico.h` File Reference

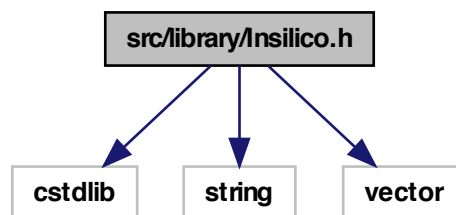
Common functions for Insilico Lab projects.

```
#include <cstdlib>
```

```
#include <string>
```

```
#include <vector>
```

Include dependency graph for `Insilico.h`:



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



## Functions

- `std::string Timestamp ()`  
*Return a timestamp string for logging purposes.*
- `Dataset * ChooseSnpsDatasetByExtension (std::string snpsFilename, bool isCleanSnps=false)`  
*Determines the data set type to instantiate based on the data set filenames's extension.*
- `bool LoadIndividualIds (std::string filename, std::vector< std::string > &retIds, bool hasHeader)`  
*Loads the individual (instance) IDs from the numerics or alternate phenotype file.*
- `bool GetMatchingIds (std::string numericsFilename, std::string altPhenotypeFilename, std::vector< std::string > numericsIds, std::vector< std::string > phenolds, std::vector< std::string > &matchingIds)`  
*Return matching IDs from numeric and/or phenotype file IDs.*

## Variables

- `static const int COMMAND_LINE_ERROR = EXIT_FAILURE`  
*Error codes.*

### 7.22.1 Detailed Description

Common functions for Insilico Lab projects.

#### Author

: Bill White

#### Version

1.0

Contact: [bill.c.white@gmail.com](mailto:bill.c.white@gmail.com) Created on 10/13/11

Definition in file [Insilico.h](#).

### 7.22.2 Function Documentation

#### 7.22.2.1 Dataset\* ChooseSnpsDatasetByExtension ( std::string snpsFilename, bool isCleanSnps = false )

Determines the data set type to instantiate based on the data set filenames's extension.

#### Parameters

in	<i>snpsFilename</i>	SNP data set filename
	<i>isCleanSnps</i>	is this a CleanSnpsDataset

**Returns**

pointer to new dataset or NULL if could not match filename extension

**7.22.2.2** `bool GetMatchingIds ( std::string numericsFilename, std::string altPhenotypeFilename,  
std::vector< std::string > numericsIds, std::vector< std::string > phenolds,  
std::vector< std::string > & matchingIds )`

Return matching IDs from numeric and/or phenotype file IDs.

**Parameters**

in	<i>numericsFilename</i>	
in	<i>altPhenotypeFilename</i>	
in	<i>numericsIds</i>	covar format file ids
in	<i>phenolds</i>	alternate phenotype file ids
out	<i>matchingIds</i>	ids that match between numerics and phenotypes

**Returns**

success

**7.22.2.3** `bool LoadIndividualIds ( std::string filename, std::vector< std::string > & retIds, bool  
hasHeader )`

Loads the individual (instance) IDs from the numerics or alternate phenotype file.

Returns the IDs through reference parameter *retIds*.

**Parameters**

in	<i>filename</i>	filename that contains covar or pheno file IDs
out	<i>vector</i>	of individual (instance) IDs (strings)
in	<i>does</i>	the file have a header row?

**Returns**

success

**7.22.2.4** `std::string Timestamp ( )`

Return a timestamp string for logging purposes.

**Returns**

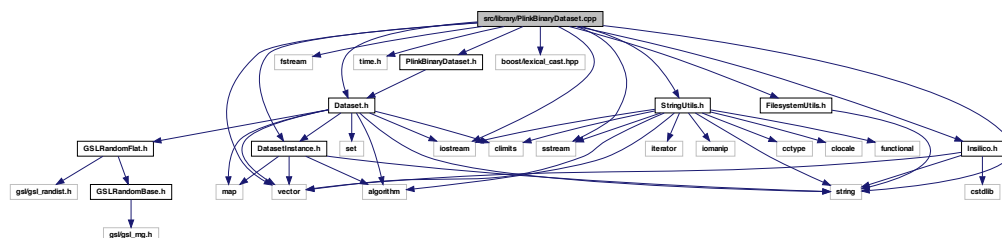
fixed-length, formatted timestamp as a string

### 7.22.3 Variable Documentation

Error codes.

## 7.23 src/library/PlinkBinaryDataset.cpp File Reference

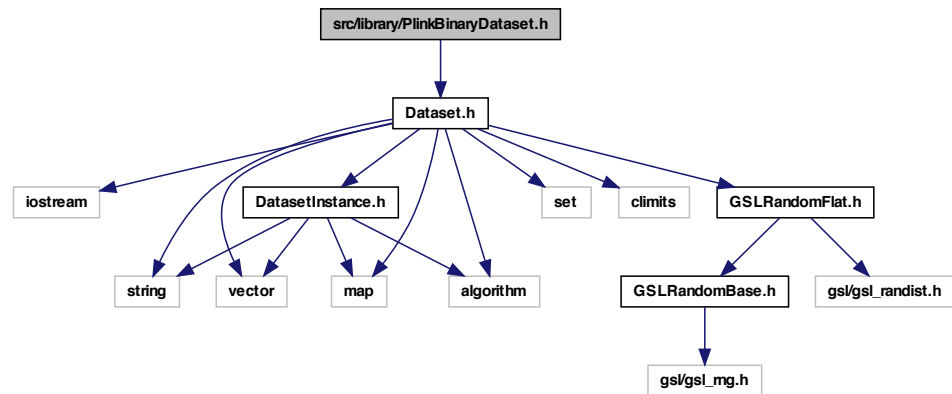
Include dependency graph for PlinkBinaryDataset.cpp:



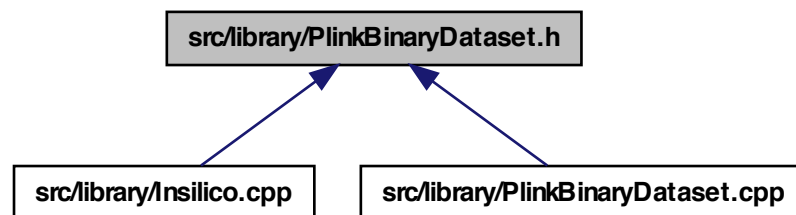
## 7.24 src/library/PlinkBinaryDataset.h File Reference

```
#include "Dataset.h"
```

Include dependency graph for PlinkBinaryDataset.h:



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



### Classes

- class [PlinkBinaryDataset](#)

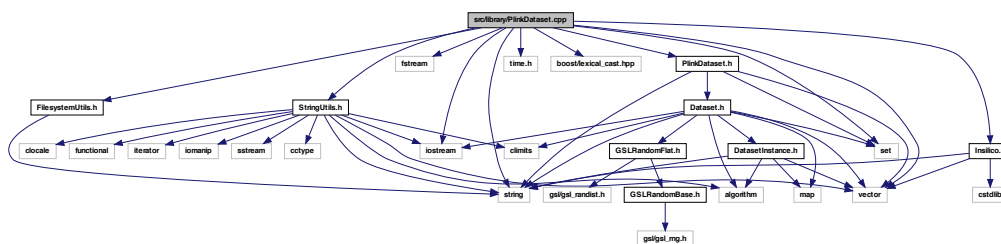
*Plink binary PED/BED file format reader.*



## 7.25 src/library/PlinkDataset.cpp File Reference

```
#include <string>
#include <iostream>
#include <fstream>
#include <vector>
#include <set>
#include <time.h>
#include <boost/lexical_cast.hpp>
#include "StringUtils.h"
#include "FilesystemUtils.h"
#include "PlinkDataset.h"
#include "Insilico.h"
```

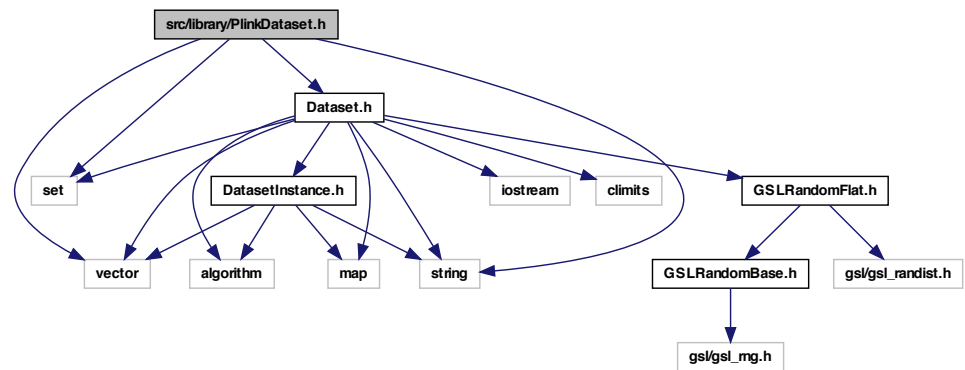
Include dependency graph for PlinkDataset.cpp:



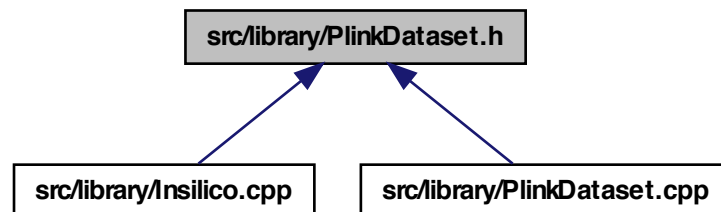
## 7.26 src/library/PlinkDataset.h File Reference

```
#include <set>
#include <vector>
#include <string>
#include "Dataset.h"
```

Include dependency graph for PlinkDataset.h:



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



## Classes

- class [PlinkDataset](#)  
*Plink MAP/PED file format reader.*

## Enumerations

- enum [MapFileType](#) { `MAP3_FILE`, `MAP4_FILE`, `ERROR_FILE` }

### 7.26.1 Enumeration Type Documentation

#### 7.26.1.1 enum MapFileType

PLINK map file types.

**Enumerator:**

**MAP3\_FILE** map 3 simplified format

**MAP4\_FILE** map 4 standard format

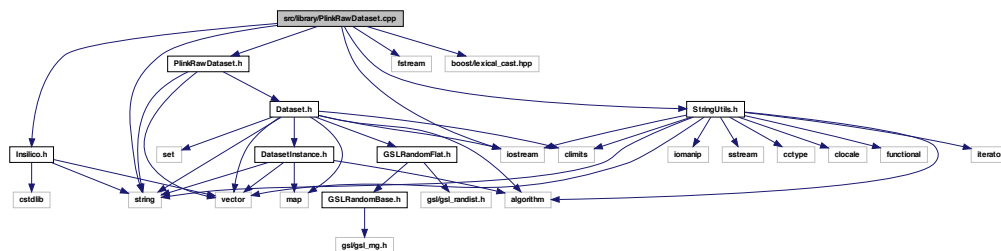
**ERROR\_FILE** default

Definition at line 28 of file PlinkDataset.h.

## 7.27 src/library/PlinkRawDataset.cpp File Reference

```
#include <string>
#include <iostream>
#include <fstream>
#include <boost/lexical_cast.hpp>
#include "StringUtils.h"
#include "PlinkRawDataset.h"
#include "Insilico.h"
```

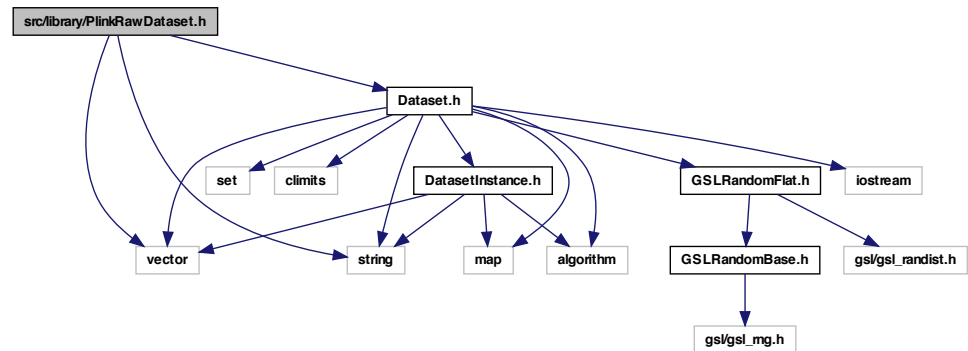
Include dependency graph for PlinkRawDataset.cpp:



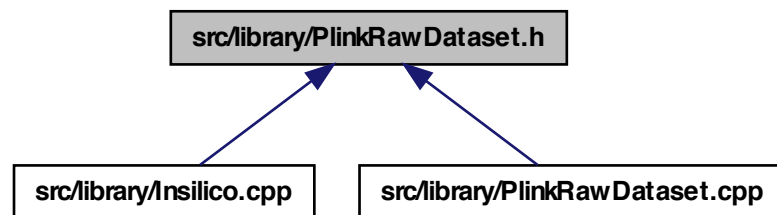
## 7.28 src/library/PlinkRawDataset.h File Reference

```
#include <string>
#include <vector>
#include "Dataset.h"
```

Include dependency graph for PlinkRawDataset.h:



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



## Classes

- class [PlinkRawDataset](#)  
*Plink recodeA/RAW file format reader.*

## 7.29 src/library/RandomJungle.cpp File Reference

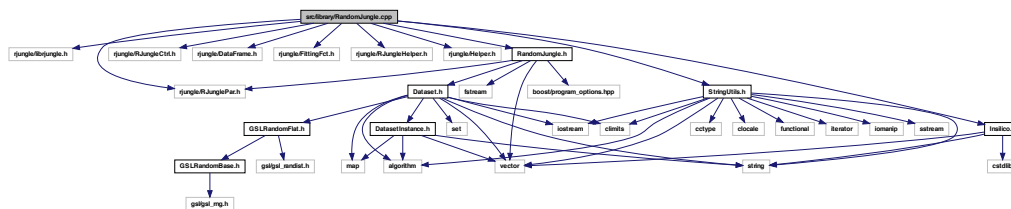
```

#include "rjungle/librjungle.h"
#include "rjungle/RJunglePar.h"
#include "rjungle/RJungleCtrl.h"

```

```
#include "rjungle/DataFrame.h"
#include "rjungle/FittingFct.h"
#include "rjungle/RJungleHelper.h"
#include "rjungle/Helper.h"
#include "RandomJungle.h"
#include "StringUtils.h"
#include "Insilico.h"
```

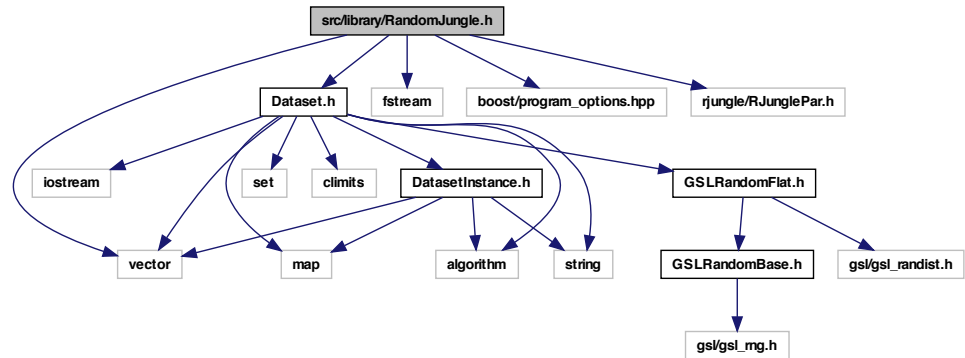
Include dependency graph for RandomJungle.cpp:



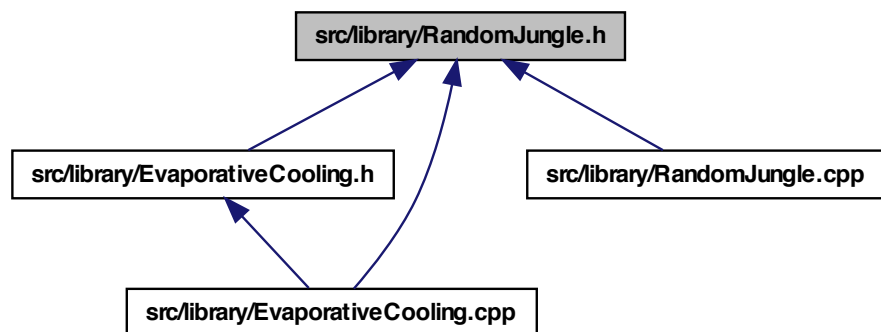
## 7.30 src/library/RandomJungle.h File Reference

```
#include <vector>
#include <fstream>
#include "Dataset.h"
#include <boost/program_options.hpp>
#include "rjungle/RJunglePar.h"
```

Include dependency graph for RandomJungle.h:



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



## Classes

- class [RandomJungle](#)  
*RandomJungle* attribute ranking algorithm.

## 7.31 src/library/ReliefF.cpp File Reference

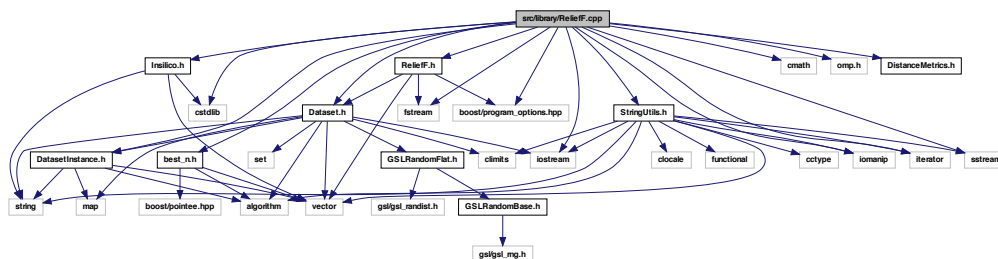
```
#include <cstdlib>
```

```

#include <iostream>
#include <fstream>
#include <iomanip>
#include <iterator>
#include <cmath>
#include <sstream>
#include <omp.h>
#include <boost/program_options.hpp>
#include "ReliefF.h"
#include "Dataset.h"
#include "DatasetInstance.h"
#include "StringUtils.h"
#include "DistanceMetrics.h"
#include "best_n.h"
#include "Insilico.h"

```

Include dependency graph for ReliefF.cpp:



## Classes

- class [deref\\_less](#)

## Typedefs

- typedef vector< pair< double, unsigned int > > [ScoresMap](#)  
*scores map: score->attribute index*
- typedef vector< pair< double, unsigned int > >::iterator [ScoresMapIt](#)  
*scores map iterator*
- typedef vector< pair< unsigned int, double > > [AttributeIndex](#)

*attribute index map: attribute index->score*

- typedef vector< pair< unsigned int, double > >::const\_iterator [AttributeIndexIt](#)

*attribute index map iterator*

- typedef pair< unsigned int, [DatasetInstance](#) \* > T

*functor for T comparison*

## Functions

- bool [scoreSort](#) (const pair< double, string > &p1, const pair< double, string > &p2)

*attribute score sorting functor*

- bool [attributeSort](#) (const pair< unsigned int, double > &p1, const pair< unsigned int, double > &p2)

*attribute index sorting functor*

- void [librelieff\\_is\\_present](#) (void)

*HACK FOR AUTOTOOLS LIBRARY DETECTION.*

### 7.31.1 Typedef Documentation

#### 7.31.1.1 typedef vector<pair<unsigned int, double> > [AttributeIndex](#)

attribute index map: attribute index->score

Definition at line 38 of file ReliefF.cpp.

#### 7.31.1.2 typedef vector<pair<unsigned int, double> >::const\_iterator [AttributeIndexIt](#)

attribute index map iterator

Definition at line 40 of file ReliefF.cpp.

#### 7.31.1.3 typedef vector<pair<double, unsigned int> > [ScoresMap](#)

scores map: score->attribute index

Definition at line 34 of file ReliefF.cpp.

#### 7.31.1.4 typedef vector<pair<double, unsigned int> >::iterator [ScoresMapIt](#)

scores map iterator

Definition at line 36 of file ReliefF.cpp.



### 7.31.1.5 typedef pair<unsigned int, DatasetInstance\*> T

functor for T comparison

Definition at line 55 of file ReliefF.cpp.

## 7.31.2 Function Documentation

### 7.31.2.1 bool attributeSort ( const pair< unsigned int, double > & *p1*, const pair< unsigned int, double > & *p2* )

attribute index sorting functor

Definition at line 49 of file ReliefF.cpp.

### 7.31.2.2 void librelieff\_is\_present ( void )

HACK FOR AUTOTOOLS LIBRARY DETECTION.

Definition at line 1062 of file ReliefF.cpp.

### 7.31.2.3 bool scoreSort ( const pair< double, string > & *p1*, const pair< double, string > & *p2* )

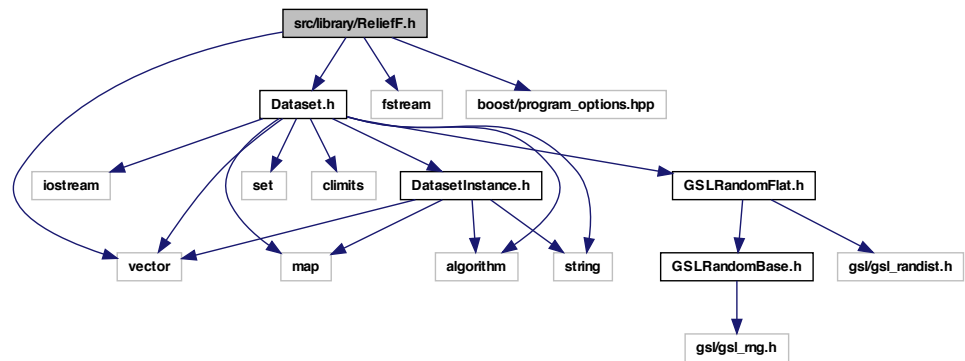
attribute score sorting functor

Definition at line 43 of file ReliefF.cpp.

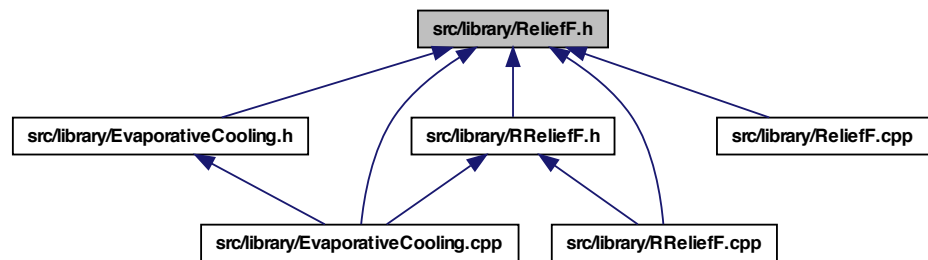
## 7.32 src/library/ReliefF.h File Reference

```
#include <vector>
#include <fstream>
#include <boost/program_options.hpp>
#include "Dataset.h"
```

Include dependency graph for ReliefF.h:



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



## Classes

- class [ReliefF](#)  
*ReliefF* attribute ranking algorithm.

## Enumerations

- enum [AnalysisType](#) {  
[SNP\\_ONLY\\_ANALYSIS](#), [SNP\\_CLEAN\\_ANALYSIS](#), [NUMERIC\\_ONLY\\_ANALYSIS](#),  
[INTEGRATED\\_ANALYSIS](#),  
[DIAGNOSTIC\\_ANALYSIS](#), [REGRESSION\\_ANALYSIS](#), [NO\\_ANALYSIS](#) }

## Functions

- void [librelieff\\_is\\_present](#) (void)  
*HACK FOR AUTOTOOLS LIBRARY DETECTION.*

### 7.32.1 Enumeration Type Documentation

#### 7.32.1.1 enum AnalysisType

Type of analysis to perform.

##### Enumerator:

**SNP\_ONLY\_ANALYSIS** discrete analysis  
**SNP\_CLEAN\_ANALYSIS** discrete analysis - no filtering  
**NUMERIC\_ONLY\_ANALYSIS** continuous attributes  
**INTEGRATED\_ANALYSIS** discrete and continuous analysis  
**DIAGNOSTIC\_ANALYSIS** diagnostic mode - no [ReliefF](#) analysis  
**REGRESSION\_ANALYSIS** regression [ReliefF](#) analysis  
**NO\_ANALYSIS** no analysis specified

Definition at line 35 of file ReliefF.h.

### 7.32.2 Function Documentation

#### 7.32.2.1 void librelieff\_is\_present ( void )

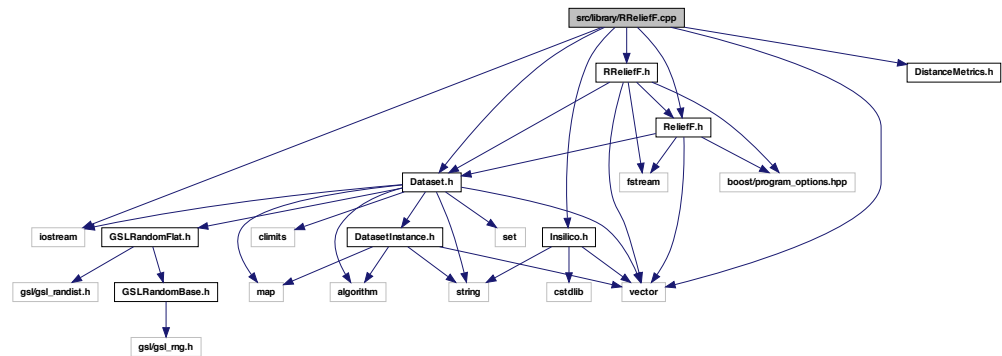
HACK FOR AUTOTOOLS LIBRARY DETECTION.

Definition at line 1062 of file ReliefF.cpp.

## 7.33 src/library/RReliefF.cpp File Reference

```
#include <iostream>
#include <vector>
#include "ReliefF.h"
#include "RReliefF.h"
#include "Dataset.h"
#include "DistanceMetrics.h"
#include "Insilico.h"
```

Include dependency graph for RReliefF.cpp:



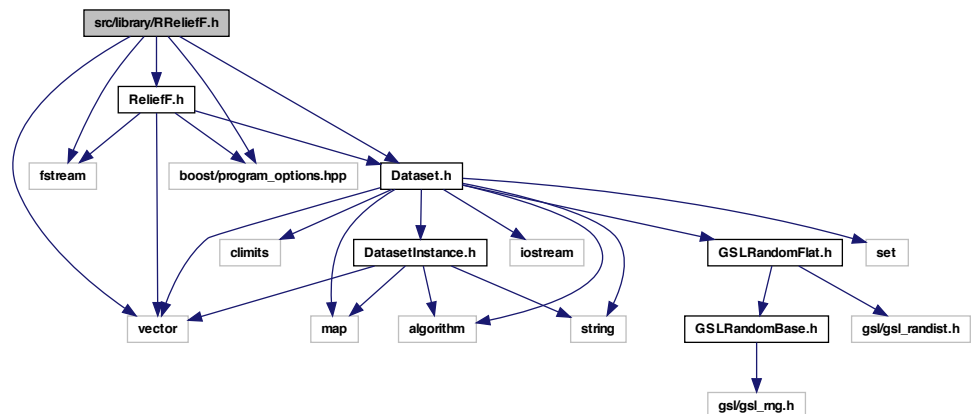
### 7.34 src/library/RReliefF.h File Reference

```

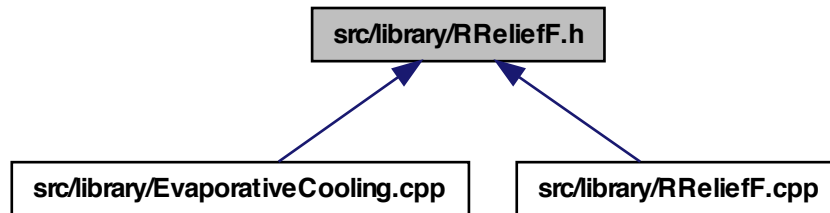
#include <vector>
#include <fstream>
#include "ReliefF.h"
#include "Dataset.h"
#include <boost/program_options.hpp>

```

Include dependency graph for RReliefF.h:



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



## Classes

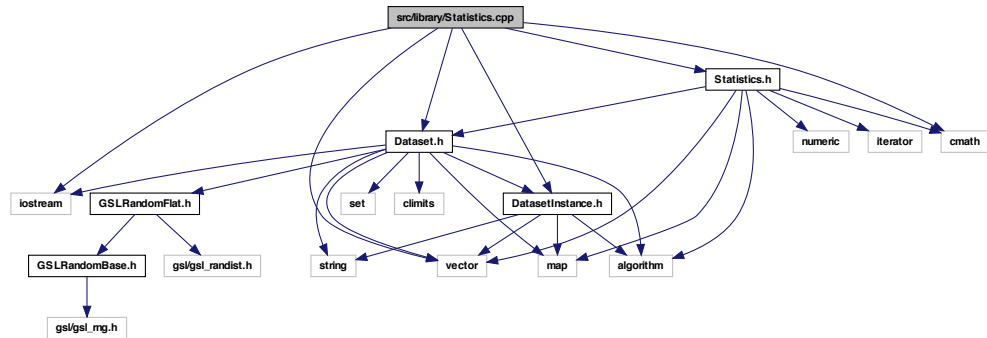
- class `RReliefF`

*Regression `ReliefF` attribute ranking algorithm.*

## 7.35 src/library/Statistics.cpp File Reference

```
#include <iostream>
#include <vector>
#include <cmath>
#include "Dataset.h"
#include "DatasetInstance.h"
#include "Statistics.h"
```

Include dependency graph for Statistics.cpp:



## Defines

- #define [DEBUG\\_Z](#) 0
- #define [DEBUG\\_E](#) 0

## Functions

- bool [ZTransform](#) (const [VectorDouble](#) &inputValues, [VectorDouble](#) &outputValues)
- *ZTransform input values.*
- double [Entropy](#) (const vector< [AttributeLevel](#) > &sequenceValues)
- double [ConditionalEntropy](#) (const vector< [AttributeLevel](#) > &sequenceValues, const vector< [AttributeLevel](#) > &givenValues)
- bool [ConstructAttributeCart](#) (const vector< [AttributeLevel](#) > &a, const vector< [AttributeLevel](#) > &b, vector< [AttributeLevel](#) > &ab)
- double [KendallTau](#) (vector< string > X, vector< string > Y)
- double [KendallTau](#) (vector< double > X, vector< double > Y)
- double [KendallTau](#) (vector< int > X, vector< int > Y)

### 7.35.1 Define Documentation

#### 7.35.1.1 #define [DEBUG\\_E](#) 0

Definition at line 18 of file Statistics.cpp.

#### 7.35.1.2 #define [DEBUG\\_Z](#) 0

Definition at line 17 of file Statistics.cpp.

### 7.35.2 Function Documentation

**7.35.2.1** `double ConditionalEntropy ( const vector< AttributeLevel > & sequenceValues,  
const vector< AttributeLevel > & givenValues )`

Definition at line 105 of file Statistics.cpp.

**7.35.2.2** `bool ConstructAttributeCart ( const vector< AttributeLevel > & a, const vector<  
AttributeLevel > & b, vector< AttributeLevel > & ab )`

Definition at line 174 of file Statistics.cpp.

**7.35.2.3** `double Entropy ( const vector< AttributeLevel > & sequenceValues )`

Definition at line 80 of file Statistics.cpp.

**7.35.2.4** `double KendallTau ( vector< string > X, vector< string > Y )`

Definition at line 187 of file Statistics.cpp.

**7.35.2.5** `double KendallTau ( vector< int > X, vector< int > Y )`

Definition at line 253 of file Statistics.cpp.

**7.35.2.6** `double KendallTau ( vector< double > X, vector< double > Y )`

Definition at line 219 of file Statistics.cpp.

**7.35.2.7** `bool ZTransform ( const VectorDouble & inputValues, VectorDouble &  
outputValues )`

ZTransform input values.

#### Parameters

in	<i>inputValues</i>	const vector of double input values
out	<i>outputValues</i>	transformed input values to z-scores with mean=0, stddev=1

#### Returns

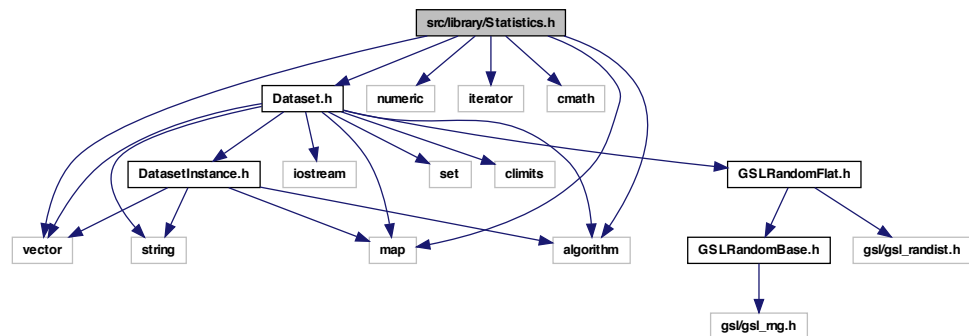
success

Definition at line 20 of file Statistics.cpp.

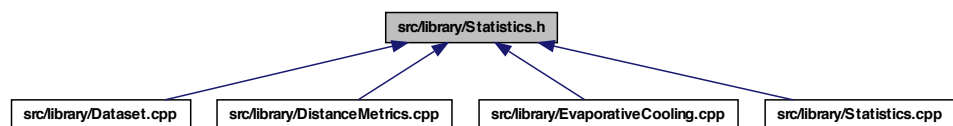
### 7.36 src/library/Statistics.h File Reference

```
#include <vector>
#include <map>
#include <numeric>
#include <iterator>
#include <cmath>
#include <algorithm>
#include "Dataset.h"
```

Include dependency graph for Statistics.h:



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



#### Typedefs

- typedef std::vector< double > [VectorDouble](#)  
*vector of doubles type*
- typedef std::vector< double >::const\_iterator [VectorDoubleIt](#)  
*vector of doubles iterator*



- typedef std::map< unsigned int, unsigned int > [Histogram](#)  
*histogram type is a map: value->count*
- typedef std::map< unsigned int, unsigned int >::const\_iterator [HistogramIt](#)  
*histogram iterator*

## Functions

- bool [ZTransform](#) (const [VectorDouble](#) &inputValues, [VectorDouble](#) &outputValues)  
*ZTransform input values.*
- double [Entropy](#) (const std::vector< [AttributeLevel](#) > &attributeValues)  
*Calculates the entropy of a sequence of unsigned integers.*
- double [ConditionalEntropy](#) (const std::vector< [AttributeLevel](#) > &attributeValues, const std::vector< [AttributeLevel](#) > &givenValues)  
*Calculates the conditional entropy of a sequence of unsigned integers based (conditioned) on another sequence of unsigned integers (the givens).*
- bool [ConstructAttributeCart](#) (const std::vector< [AttributeLevel](#) > &a, const std::vector< [AttributeLevel](#) > &b, std::vector< [AttributeLevel](#) > &ab)  
*Create a new attribute that is the cartesian product of a and b.*
- double [KendallTau](#) (std::vector< std::string > X, std::vector< std::string > Y)  
*Compute KendallTau for two ranked vectors of strings.*
- double [KendallTau](#) (std::vector< double > X, std::vector< double > Y)  
*Compute KendallTau for two ranked vectors of doubles.*
- double [KendallTau](#) (std::vector< int > X, std::vector< int > Y)  
*Compute KendallTau for two ranked vectors of integers.*
- template<class T >  
std::pair< double, double > [VarStd](#) (std::vector< T > &values)  
*Calculate variance and standard deviation of a vector of values.*

### 7.36.1 Typedef Documentation

#### 7.36.1.1 typedef std::map<unsigned int, unsigned int> [Histogram](#)

histogram type is a map: value->count

Definition at line 30 of file Statistics.h.

#### 7.36.1.2 typedef std::map<unsigned int, unsigned int>::const\_iterator [HistogramIt](#)

histogram iterator

Definition at line 32 of file Statistics.h.

### 7.36.1.3 `typedef std::vector<double> VectorDouble`

vector of doubles type

Definition at line 26 of file Statistics.h.

### 7.36.1.4 `typedef std::vector<double>::const_iterator VectorDoubleIt`

vector of doubles iterator

Definition at line 28 of file Statistics.h.

## 7.36.2 Function Documentation

### 7.36.2.1 `double ConditionalEntropy ( const std::vector< AttributeLevel > & attributeValues, const std::vector< AttributeLevel > & givenValues )`

Calculates the conditional entropy of a sequence of unsigned integers based (conditioned) on another sequence of unsigned integers (the givens).

$P(\text{sequenceValues} \mid \text{givenValues})$

#### Parameters

in	<i>attributeValues</i>	vector of values
in	<i>givenValues</i>	vector of givens

#### Returns

conditioniional entropy as a double-precision float

### 7.36.2.2 `bool ConstructAttributeCart ( const std::vector< AttributeLevel > & a, const std::vector< AttributeLevel > & b, std::vector< AttributeLevel > & ab )`

Create a new attribute that is the cartesian product of a and b.

NOTE: works for genotypes; need to verify for missign data levels, etc.

#### Parameters

in	<i>a</i>	attributes vector a
in	<i>b</i>	attributes vector b
out	<i>vector</i>	ab, the cartesian product of a and b

#### Returns

success

**7.36.2.3 double Entropy ( const std::vector< AttributeLevel > & attributeValues )**

Calculates the entropy of a sequence of unsigned integers.

**Parameters**

in	<i>attributeValues</i>	vector of sequence values - unsigned ints - positive categorical
----	------------------------	--

**Returns**

entropy as a double-precision float

**7.36.2.4 double KendallTau ( std::vector< double > X, std::vector< double > Y )**

Compute KendallTau for two ranked vectors of doubles.

Why Kenall Tau - G. E. NOETHER <http://www.rsscse-edu.org.uk/tsj/bts/noether/text.html>

**Parameters**

in	X	ranked attribute vector X
in	Y	ranked attribute vector Y

**Returns**

Kendall Tau value (-1, 1)

**7.36.2.5 double KendallTau ( std::vector< std::string > X, std::vector< std::string > Y )**

Compute KendallTau for two ranked vectors of strings.

Why Kenall Tau - G. E. NOETHER <http://www.rsscse-edu.org.uk/tsj/bts/noether/text.html>

**Parameters**

in	X	ranked attribute vector X
in	Y	ranked attribute vector Y

**Returns**

Kendall Tau value (-1, 1)

**7.36.2.6 double KendallTau ( std::vector< int > X, std::vector< int > Y )**

Compute KendallTau for two ranked vectors of integers.

Why Kenall Tau - G. E. NOETHER <http://www.rsscse-edu.org.uk/tsj/bts/noether/text.html>

**Parameters**

in	X	ranked attribute vector X
in	Y	ranked attribute vector Y

**Returns**

Kendall Tau value (-1, 1)

**7.36.2.7** `template<class T> std::pair<double, double> VarStd ( std::vector< T> & values )`

Calculate variance and standard deviation of a vector of values.

**Parameters**

in	<i>ranked</i>	attribute lists X and Y
----	---------------	-------------------------

**Returns**

Kendall Tau value (-1, 1)

Definition at line 101 of file Statistics.h.

**7.36.2.8** `bool ZTransform ( const VectorDouble & inputValues, VectorDouble & outputValues )`

ZTransform input values.

**Parameters**

in	<i>inputValues</i>	const vector of double input values
out	<i>outputValues</i>	transformed input values to z-scores with mean=0, stddev=1

**Returns**

success

Definition at line 20 of file Statistics.cpp.

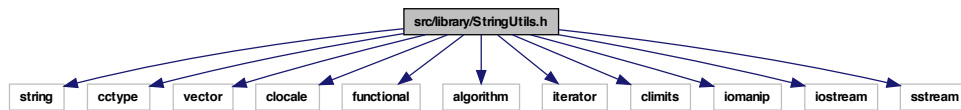
## 7.37 src/library/StringUtils.h File Reference

Various string-related utilities.

```
#include <string>
#include <cctype>
#include <vector>
#include <locale>
```

```
#include <functional>
#include <algorithm>
#include <iterator>
#include <climits>
#include <iomanip>
#include <iostream>
#include <sstream>
```

Include dependency graph for StringUtils.h:



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



## Classes

- class [insilico::is\\_classified< Type, charT >](#)
- class [insilico::do\\_to\\_upper< charT >](#)
- class [insilico::do\\_to\\_lower< charT >](#)

## Namespaces

- namespace [insilico](#)

## Functions

- template<typename stringT >  
stringT [insilico::trim\\_left](#) (const stringT &s, const std::locale &loc=std::locale())
- template<typename stringT >  
stringT [insilico::trim\\_right](#) (const stringT &s, const std::locale &loc=std::locale())
- template<typename stringT >  
stringT [insilico::trim](#) (const stringT &s, const std::locale &loc=std::locale())

- `template<typename Container , typename stringT >`  
`void insilico::split (Container &cont, const stringT &s, const std::locale &loc=std::locale())`
- `template<typename Container , typename stringT >`  
`void insilico::split (Container &cont, const stringT &s, const stringT &delim)`
- `template<typename Container , typename stringT , typename Pred >`  
`void insilico::split_if (Container &cont, const stringT &s, const Pred &pred)`
- `template<typename It , typename stringT >`  
`stringT insilico::join (const It &begin, const It &end, const stringT &delim)`
- `template<typename stringT >`  
`stringT insilico::to_upper (const stringT &str, const std::locale &loc=std::locale())`
- `template<typename stringT >`  
`stringT insilico::to_lower (const stringT &str, const std::locale &loc=std::locale())`
- `std::string insilico::trim_left (const char *s, const std::locale &loc=std::locale())`
- `std::wstring insilico::trim_left (const wchar_t *s, const std::locale &loc=std::locale())`
- `std::string insilico::trim_right (const char *s, const std::locale &loc=std::locale())`
- `std::wstring insilico::trim_right (const wchar_t *s, const std::locale &loc=std::locale())`
- `std::string insilico::trim (const char *s, const std::locale &loc=std::locale())`
- `std::wstring insilico::trim (const wchar_t *s, const std::locale &loc=std::locale())`
- `template<typename Container >`  
`void insilico::split (Container &cont, const char *s, const std::locale &loc=std::locale())`
- `template<typename Container >`  
`void insilico::split (Container &cont, const wchar_t *s, const std::locale &loc=std::locale())`
- `template<typename Container >`  
`void insilico::split (Container &cont, const std::string &s, const char *delim)`
- `template<typename Container >`  
`void insilico::split (Container &cont, const char *s, const std::string &delim)`
- `template<typename Container >`  
`void insilico::split (Container &cont, const char *s, const char *delim)`
- `template<typename Container >`  
`void insilico::split (Container &cont, const std::wstring &s, const wchar_t *delim)`
- `template<typename Container >`  
`void insilico::split (Container &cont, const wchar_t *s, const std::wstring &delim)`
- `template<typename Container >`  
`void insilico::split (Container &cont, const wchar_t *s, const wchar_t *delim)`
- `template<typename Container , typename Pred >`  
`void insilico::split_if (Container &cont, const char *s, const Pred &pred)`
- `template<typename Container , typename Pred >`  
`void insilico::split_if (Container &cont, const wchar_t *s, const Pred &pred)`
- `template<typename It >`  
`std::string insilico::join (const It &begin, const It &end, const char *delim)`
- `template<typename It >`  
`std::wstring insilico::join (const It &begin, const It &end, const wchar_t *delim)`
- `std::string insilico::to_upper (const char *s, const std::locale &loc=std::locale())`
- `std::wstring insilico::to_upper (const wchar_t *s, const std::locale &loc=std::locale())`
- `std::string insilico::to_lower (const char *s, const std::locale &loc=std::locale())`
- `std::wstring insilico::to_lower (const wchar_t *s, const std::locale &loc=std::locale())`
- `template<typename T >`  
`std::string insilico::get_bits (T value)`

- `template<typename T >`  
`std::string insilico::zeroPadNumber (T num, int padSize)`

### 7.37.1 Detailed Description

Various string-related utilities. This is originally from Nate Barney circa Moore Lab days 2003-2007. His function naming follows lowercase with underscores style, while my additions are camelCase.

#### Author

Bill White, Nate Barney

#### Version

1.0

Contact: [bill.c.white@gmail.com](mailto:bill.c.white@gmail.com) Created on: 10/7/04

Definition in file [StringUtils.h](#).

# Index

- ~ArffDataset
  - ArffDataset, [19](#)
- ~ChiSquared
  - ChiSquared, [24](#)
- ~CleanSnpDataDataset
  - CleanSnpDataDataset, [30](#)
- ~Dataset
  - Dataset, [40](#)
- ~DatasetInstance
  - DatasetInstance, [72](#)
- ~EvaporativeCooling
  - EvaporativeCooling, [86](#)
- ~GSLRandomBase
  - GSLRandomBase, [92](#)
- ~GSLRandomFlat
  - GSLRandomFlat, [96](#)
- ~PlinkBinaryDataset
  - PlinkBinaryDataset, [102](#)
- ~PlinkDataset
  - PlinkDataset, [111](#)
- ~PlinkRawDataset
  - PlinkRawDataset, [116](#)
- ~RReliefF
  - RReliefF, [137](#)
- ~RandomJungle
  - RandomJungle, [121](#)
- ~ReliefF
  - ReliefF, [127](#)
- AddInfluenceFactorD
  - DatasetInstance, [72](#)
- AddNumeric
  - DatasetInstance, [72](#)
- algorithmType
  - EvaporativeCooling, [88](#)
- alternatePhenotypesFilename
  - Dataset, [63](#)
- AnalysisType
  - ReliefF.h, [189](#)
- analysisType
  - EvaporativeCooling, [88](#)
- ReliefF, [130](#)
- ARFF\_DATASET
  - Dataset.h, [150](#)
- ARFF\_DATE\_TYPE
  - ArffDataset.h, [141](#)
- ARFF\_ERROR\_TYPE
  - ArffDataset.h, [141](#)
- ARFF\_NOMINAL\_TYPE
  - ArffDataset.h, [141](#)
- ARFF\_NUMERIC\_TYPE
  - ArffDataset.h, [141](#)
- ARFF\_STRING\_TYPE
  - ArffDataset.h, [141](#)
- ArffAttributeType
  - ArffDataset.h, [141](#)
- ArffDataset, [15](#)
  - ~ArffDataset, [19](#)
  - ArffDataset, [19](#)
  - attributeTypes, [21](#)
  - GetAttributeLevel, [19](#)
  - GetDiscreteClassLevel, [19](#)
  - GetNumericClassLevel, [20](#)
  - GetTypeOf, [20](#)
  - LoadSnps, [20](#)
  - missingAttributeValuesToCheck, [21](#)
  - missingClassValuesToCheck, [21](#)
  - nominalValues, [21](#)
  - PrintNominalsMapping, [20](#)
  - relationName, [21](#)
- ArffDataset.h
  - ARFF\_DATE\_TYPE, [141](#)
  - ARFF\_ERROR\_TYPE, [141](#)
  - ARFF\_NOMINAL\_TYPE, [141](#)
  - ARFF\_NUMERIC\_TYPE, [141](#)
  - ARFF\_STRING\_TYPE, [141](#)
  - ArffAttributeType, [141](#)
- attributeAlleleCounts
  - Dataset, [63](#)
- attributeAlleles
  - Dataset, [63](#)
- AttributeIndex



- ReliefF.cpp, 186
- AttributeIndexIt
  - ReliefF.cpp, 186
- AttributeInteractionInformation
  - Dataset, 41
- AttributeLevel
  - DatasetInstance.h, 154
- attributeLevelsSeen
  - Dataset, 64
- attributeMinorAllele
  - Dataset, 64
- attributeMutationMap
  - Dataset, 64
- AttributeMutationType
  - Dataset.h, 149
- attributeMutationTypes
  - Dataset, 64
- attributeNames
  - Dataset, 64
- attributes
  - DatasetInstance, 77
- attributesMask
  - Dataset, 64
- attributesMaskPushed
  - Dataset, 64
- attributeSort
  - ReliefF.cpp, 187
- AttributeType
  - Dataset.h, 150
- attributeTypes
  - ArffDataset, 21
- best\_n
  - insilico, 10
- bestNeighbors
  - DatasetInstance, 77
- bestNeighborsDiffClass
  - DatasetInstance, 77
- bestNeighborsSameClass
  - DatasetInstance, 77
- CalculateGainMatrix
  - Dataset, 41
- CalculateInteractionInformation
  - Dataset, 41
- CheckHardyWeinbergEquilibrium
  - Dataset, 41
- CheckMissing
  - DistanceMetrics.cpp, 158
  - DistanceMetrics.h, 162
- CheckMissingNumeric
  - DistanceMetrics.cpp, 158
  - DistanceMetrics.h, 162
- ChiSquared, 21
  - ~ChiSquared, 24
  - ChiSquared, 24
  - chiSquaredValues, 26
  - ClearTables, 24
  - ComputeScore, 24
  - ComputeScores, 25
  - dataset, 26
  - expectedContingencyTable, 26
  - GetFrequencyCounts, 25
  - numClasses, 26
  - numLevels, 26
  - observedFreqTable, 26
  - PrepareForAttribute, 25
  - PrintScores, 25
  - PrintTables, 25
  - scores, 27
  - WriteScores, 26
- chiSquaredValues
  - ChiSquared, 26
- ChooseSnpsDatasetByExtension
  - Insilico.cpp, 173
  - Insilico.h, 175
- classColumn
  - Dataset, 65
- classIndexes
  - Dataset, 65
- classLabel
  - DatasetInstance, 77
- ClassLevel
  - DatasetInstance.h, 154
- CleanSnpsDataset, 27
  - ~CleanSnpsDataset, 30
  - CleanSnpsDataset, 30
  - LoadSnps, 30
- ClearInfluenceFactors
  - DatasetInstance, 72
- ClearTables
  - ChiSquared, 24
- COMMAND\_LINE\_ERROR
  - Insilico.h, 177
- ComputeAttributeScores
  - RandomJungle, 121
  - ReliefF, 127
  - RReliefF, 137
- ComputeAttributeScoresCleanSnps
  - ReliefF, 128

- ComputeAttributeScoresIteratively
  - ReliefF, [128](#)
- ComputeECScores
  - EvaporativeCooling, [86](#)
- ComputeFreeEnergy
  - EvaporativeCooling, [86](#)
- ComputeInstanceToInstanceDistance
  - ReliefF, [128](#)
- ComputeScore
  - ChiSquared, [24](#)
- ComputeScores
  - ChiSquared, [25](#)
- ComputeWeightByDistanceFactors
  - ReliefF, [128](#)
- ConditionalEntropy
  - Statistics.cpp, [193](#)
  - Statistics.h, [196](#)
- ConstructAttributeCart
  - Statistics.cpp, [193](#)
  - Statistics.h, [196](#)
- continuousPhenotypeMinMax
  - Dataset, [65](#)
- CSV\_DELIMITED\_DATASET
  - Dataset.h, [150](#)
- Dataset, [31](#)
  - ~Dataset, [40](#)
  - alternatePhenotypesFilename, [63](#)
  - attributeAlleleCounts, [63](#)
  - attributeAlleles, [63](#)
  - AttributeInteractionInformation, [41](#)
  - attributeLevelsSeen, [64](#)
  - attributeMinorAllele, [64](#)
  - attributeMutationMap, [64](#)
  - attributeMutationTypes, [64](#)
  - attributeNames, [64](#)
  - attributesMask, [64](#)
  - attributesMaskPushed, [64](#)
  - CalculateGainMatrix, [41](#)
  - CalculateInteractionInformation, [41](#)
  - CheckHardyWeinbergEquilibrium, [41](#)
  - classColumn, [65](#)
  - classIndexes, [65](#)
  - continuousPhenotypeMinMax, [65](#)
  - Dataset, [40](#)
  - ExtractAttributes, [42](#)
  - genotypeCounts, [65](#)
  - GetAlternatePhenotypesFilename, [42](#)
  - GetAttribute, [42](#)
  - GetAttributeIndexFromName, [43](#)
  - GetAttributeLevel, [43](#)
  - GetAttributeMAF, [43](#)
  - GetAttributeMutationType, [44](#)
  - GetAttributeNames, [44](#)
  - GetAttributeRowCol, [44](#)
  - GetAttributeValues, [45](#)
  - GetAttributeValueType, [45](#)
  - GetClassColumn, [46](#)
  - GetClassIndexes, [46](#)
  - GetClassProbability, [46](#)
  - GetClassValues, [46](#)
  - GetClassValueType, [47](#)
  - GetDiscreteClassLevel, [47](#)
  - GetInstance, [47](#)
  - GetInstanceIds, [48](#)
  - GetInstanceIndexForID, [48](#)
  - GetIntForGenotype, [48](#)
  - GetMeanForNumeric, [49](#)
  - GetMinMaxForContinuousPhenotype, [49](#)
  - GetMinMaxForNumeric, [49](#)
  - GetNumeric, [49](#)
  - GetNumericClassLevel, [50](#)
  - GetNumericIndexFromName, [50](#)
  - GetNumericRowCol, [50](#)
  - GetNumericsFilename, [51](#)
  - GetNumericsNames, [51](#)
  - GetNumericValues, [51](#)
  - GetProbabilityValueGivenClass, [52](#)
  - GetRandomInstance, [52](#)
  - GetSnpsFilename, [52](#)
  - GetVariableNames, [52](#)
  - HasAlternatePhenotypes, [53](#)
  - hasAlternatePhenotypes, [65](#)
  - HasContinuousPhenotypes, [53](#)
  - hasContinuousPhenotypes, [65](#)
  - HasGenotypes, [53](#)
  - hasGenotypes, [65](#)
  - HasNumerics, [53](#)
  - hasNumerics, [66](#)
  - instanceIds, [66](#)
  - instanceIdsToLoad, [66](#)
  - instances, [66](#)
  - instancesMask, [66](#)
  - instancesMaskPushed, [66](#)
  - IsLoadableInstanceID, [53](#)
  - levelCounts, [66](#)
  - levelCountsByClass, [66](#)
  - LoadAlternatePhenotypes, [53](#)
  - LoadDataset, [54](#)

- LoadNumerics, [54](#)
- LoadSnps, [54](#)
- MaskGetAllVariableNames, [55](#)
- MaskGetAttributeIndices, [55](#)
- MaskGetAttributeMask, [55](#)
- MaskGetInstanceIds, [56](#)
- MaskGetInstanceIndices, [56](#)
- MaskGetInstanceMask, [56](#)
- MaskIncludeAllAttributes, [56](#)
- MaskIncludeAllInstances, [56](#)
- maskIsPushed, [67](#)
- MaskPopAll, [57](#)
- MaskPushAll, [57](#)
- MaskRemoveAttribute, [57](#)
- MaskRemoveInstance, [57](#)
- MaskSearchAttribute, [58](#)
- MaskSearchInstance, [58](#)
- MaskWriteNewDataset, [58](#)
- missingNumericValues, [67](#)
- missingValues, [67](#)
- NumAttributes, [59](#)
- NumClasses, [59](#)
- numericsFilename, [67](#)
- numericsIds, [67](#)
- numericsMask, [67](#)
- numericsMaskPushed, [67](#)
- numericsMinMax, [67](#)
- numericsNames, [68](#)
- NumInstances, [59](#)
- NumLevels, [59](#)
- NumNumerics, [59](#)
- NumVariables, [59](#)
- phenotypesIds, [68](#)
- Print, [60](#)
- PrintAttributeLevelsSeen, [60](#)
- PrintClassIndexInfo, [60](#)
- PrintLevelCounts, [60](#)
- PrintMaskStats, [60](#)
- PrintMissingValuesStats, [60](#)
- PrintNumericsStats, [60](#)
- PrintRecodeMap, [60](#)
- PrintStats, [61](#)
- PrintStatsSimple, [61](#)
- rng, [68](#)
- RunSnpDiagnosticTests, [61](#)
- SNPHWE, [61](#)
- snpsFilename, [68](#)
- SwapAttributes, [62](#)
- UpdateAllLevelCounts, [62](#)
- UpdateLevelCounts, [62](#)
- WriteLevelCounts, [62](#)
- WriteNewDataset, [63](#)
- dataset
  - ChiSquared, [26](#)
  - DatasetInstance, [78](#)
  - EvaporativeCooling, [88](#)
  - RandomJungle, [121](#)
  - ReliefF, [130](#)
- Dataset.h
  - ARFF\_DATASET, [150](#)
  - AttributeMutationType, [149](#)
  - AttributeType, [150](#)
  - CSV\_DELIMITED\_DATASET, [150](#)
  - DISCRETE\_TYPE, [150](#)
  - DISCRETE\_VALUE, [150](#)
  - INVALID\_ATTRIBUTE\_VALUE, [151](#)
  - INVALID\_DISCRETE\_CLASS\_VALUE, [151](#)
  - INVALID\_DISTANCE, [151](#)
  - INVALID\_INDEX, [151](#)
  - INVALID\_NUMERIC\_CLASS\_VALUE, [151](#)
  - INVALID\_NUMERIC\_VALUE, [151](#)
  - MISSING\_VALUE, [150](#)
  - MISSING\_ATTRIBUTE\_VALUE, [151](#)
  - MISSING\_DISCRETE\_CLASS\_VALUE, [151](#)
  - MISSING\_NUMERIC\_CLASS\_VALUE, [152](#)
  - MISSING\_NUMERIC\_VALUE, [152](#)
  - NO\_OUTPUT\_DATASET, [150](#)
  - NO\_TYPE, [150](#)
  - NO\_VALUE, [150](#)
  - NUMERIC\_TYPE, [150](#)
  - NUMERIC\_VALUE, [150](#)
  - OutputDatasetType, [150](#)
  - TAB\_DELIMITED\_DATASET, [150](#)
  - TRANSITION\_MUTATION, [150](#)
  - TRANSVERSION\_MUTATION, [150](#)
  - UNKNOWN\_MUTATION, [150](#)
  - ValueType, [150](#)
- DatasetInstance, [68](#)
  - ~DatasetInstance, [72](#)
  - AddInfluenceFactorD, [72](#)
  - AddNumeric, [72](#)
  - attributes, [77](#)
  - bestNeighborIds, [77](#)
  - bestNeighborIdsDiffClass, [77](#)
  - bestNeighborIdsSameClass, [77](#)
  - classLabel, [77](#)

- ClearInfluenceFactors, 72
- dataset, 78
- DatasetInstance, 72
- GetAttribute, 73
- GetClass, 73
- GetDatasetPtr, 73
- GetInfluenceFactorD, 73
- GetNNearestInstances, 73, 74
- GetNumeric, 74
- GetPredictedValueTau, 75
- LoadInstanceFromVector, 75
- neighborInfluenceFactorDs, 78
- NumAttributes, 75
- numerics, 78
- NumNumerics, 75
- predictedValueTau, 78
- Print, 75
- PrintDistancePairs, 75
- SetClass, 76
- SetDistanceSums, 76
- SetPredictedValueTau, 76
- SwapAttributes, 77
- DatasetInstance.cpp
  - T, 153
- DatasetInstance.h
  - AttributeLevel, 154
  - ClassLevel, 154
  - DistancePair, 154
  - DistancePairs, 155
  - DistancePairsIt, 155
  - NumericLevel, 155
- DEBUG\_E
  - Statistics.cpp, 192
- DEBUG\_Z
  - Statistics.cpp, 192
- Debugging.h
  - PrintVector, 156, 157
- deref\_less, 78
  - operator(), 79
- deref\_less\_bcw, 79
  - operator(), 79
- DIAGNOSTIC\_ANALYSIS
  - ReliefF.h, 189
- diffAMM
  - DistanceMetrics.cpp, 159
  - DistanceMetrics.h, 162
- diffGMM
  - DistanceMetrics.cpp, 159
  - DistanceMetrics.h, 163
- diffManhattan
  - DistanceMetrics.cpp, 159
  - DistanceMetrics.h, 163
- diffPredictedValueTau
  - DistanceMetrics.cpp, 160
  - DistanceMetrics.h, 163
- DISCRETE\_TYPE
  - Dataset.h, 150
- DISCRETE\_VALUE
  - Dataset.h, 150
- DistanceMetrics.cpp
  - CheckMissing, 158
  - CheckMissingNumeric, 158
  - diffAMM, 159
  - diffGMM, 159
  - diffManhattan, 159
  - diffPredictedValueTau, 160
  - norm, 160
- DistanceMetrics.h
  - CheckMissing, 162
  - CheckMissingNumeric, 162
  - diffAMM, 162
  - diffGMM, 163
  - diffManhattan, 163
  - diffPredictedValueTau, 163
  - norm, 164
- DistancePair
  - DatasetInstance.h, 154
- DistancePairs
  - DatasetInstance.h, 155
- DistancePairsIt
  - DatasetInstance.h, 155
- do\_to\_lower
  - insilico::do\_to\_lower, 80
- do\_to\_upper
  - insilico::do\_to\_upper, 81
- doRemovePercent
  - ReliefF, 130
- EC\_ALL
  - EvaporativeCooling.h, 167
- EC\_RF
  - EvaporativeCooling.h, 167
- EC\_RJ
  - EvaporativeCooling.h, 167
- EcAlgorithmType
  - EvaporativeCooling.h, 167
- EcScores
  - EvaporativeCooling.h, 167
- ecScores
  - EvaporativeCooling, 88

- EcScoresClt
  - EvaporativeCooling.h, [167](#)
- EcScoresIt
  - EvaporativeCooling.h, [167](#)
- Entropy
  - Statistics.cpp, [193](#)
  - Statistics.h, [196](#)
- ERROR\_FILE
  - PlinkDataset.h, [181](#)
- evaporatedAttributes
  - EvaporativeCooling, [89](#)
- EvaporativeCooling, [82](#)
  - ~EvaporativeCooling, [86](#)
  - algorithmType, [88](#)
  - analysisType, [88](#)
  - ComputeECScores, [86](#)
  - ComputeFreeEnergy, [86](#)
  - dataset, [88](#)
  - ecScores, [88](#)
  - evaporatedAttributes, [89](#)
  - EvaporativeCooling, [86](#)
  - freeEnergyScores, [89](#)
  - GetAlgorithmType, [86](#)
  - GetECScores, [86](#)
  - GetRandomJungleScores, [87](#)
  - GetReliefFScores, [87](#)
  - numRFThreads, [89](#)
  - numTargetAttributes, [89](#)
  - numToRemovePerIteration, [89](#)
  - outFilesPrefix, [89](#)
  - paramsMap, [89](#)
  - PrintAllScoresTabular, [87](#)
  - PrintAttributeScores, [87](#)
  - PrintKendallTaus, [87](#)
  - randomJungle, [89](#)
  - reliefF, [90](#)
  - RemoveWorstAttributes, [87](#)
  - rfScores, [90](#)
  - rjScores, [90](#)
  - RunReliefF, [88](#)
  - WriteAttributeScores, [88](#)
- EvaporativeCooling.cpp
  - scoresSortAsc, [165](#)
  - scoresSortAscByName, [165](#)
  - scoresSortDesc, [165](#)
- EvaporativeCooling.h
  - EC\_ALL, [167](#)
  - EC\_RF, [167](#)
  - EC\_RJ, [167](#)
  - EcAlgorithmType, [167](#)
  - EcScores, [167](#)
  - EcScoresClt, [167](#)
  - EcScoresIt, [167](#)
- expectedContingencyTable
  - ChiSquared, [26](#)
- ExtractAttributes
  - Dataset, [42](#)
- filenameBase
  - PlinkBinaryDataset, [106](#)
  - PlinkDataset, [113](#)
- FilesystemUtils.cpp
  - GetFileBasename, [168](#)
  - GetFileExtension, [168](#)
- FilesystemUtils.h
  - GetFileBasename, [170](#)
  - GetFileExtension, [170](#)
- finalScores
  - ReliefF, [130](#)
- freeEnergyScores
  - EvaporativeCooling, [89](#)
- genotypeCounts
  - Dataset, [65](#)
- get\_bits
  - insilico, [11](#)
- GetAlgorithmType
  - EvaporativeCooling, [86](#)
- GetAlternatePhenotypesFilename
  - Dataset, [42](#)
- GetAttribute
  - Dataset, [42](#)
  - DatasetInstance, [73](#)
- GetAttributeIndexFromName
  - Dataset, [43](#)
- GetAttributeLevel
  - ArffDataset, [19](#)
  - Dataset, [43](#)
  - PlinkBinaryDataset, [102](#)
- GetAttributeMAF
  - Dataset, [43](#)
  - PlinkBinaryDataset, [102](#)
  - PlinkDataset, [111](#)
- GetAttributeMutationType
  - Dataset, [44](#)
  - PlinkBinaryDataset, [103](#)
  - PlinkDataset, [111](#)
- GetAttributeNames
  - Dataset, [44](#)
- GetAttributeRowCol

- Dataset, [44](#)
- GetAttributeValues
  - Dataset, [45](#)
- GetAttributeValueType
  - Dataset, [45](#)
  - PlinkBinaryDataset, [103](#)
- GetClass
  - DatasetInstance, [73](#)
- GetClassColumn
  - Dataset, [46](#)
- GetClassIndexes
  - Dataset, [46](#)
- GetClassProbability
  - Dataset, [46](#)
- GetClassValues
  - Dataset, [46](#)
- GetClassValueType
  - Dataset, [47](#)
  - PlinkBinaryDataset, [104](#)
  - PlinkDataset, [112](#)
  - PlinkRawDataset, [117](#)
- GetDatasetPtr
  - DatasetInstance, [73](#)
- GetDiscreteClassLevel
  - ArffDataset, [19](#)
  - Dataset, [47](#)
  - PlinkBinaryDataset, [104](#)
  - PlinkDataset, [112](#)
  - PlinkRawDataset, [117](#)
- GetECScores
  - EvaporativeCooling, [86](#)
- GetFileBasename
  - FilesystemUtils.cpp, [168](#)
  - FilesystemUtils.h, [170](#)
- GetFileExtension
  - FilesystemUtils.cpp, [168](#)
  - FilesystemUtils.h, [170](#)
- GetFrequencyCounts
  - ChiSquared, [25](#)
- GetInfluenceFactorD
  - DatasetInstance, [73](#)
- GetInstance
  - Dataset, [47](#)
- GetInstancelds
  - Dataset, [48](#)
- GetInstanceIndexForID
  - Dataset, [48](#)
- GetIntForGenotype
  - Dataset, [48](#)
- GetMatchingIds
  - Insilico.cpp, [173](#)
  - Insilico.h, [176](#)
- GetMeanForNumeric
  - Dataset, [49](#)
- GetMinMaxForContinuousPhenotype
  - Dataset, [49](#)
- GetMinMaxForNumeric
  - Dataset, [49](#)
- GetNNearestInstances
  - DatasetInstance, [73](#), [74](#)
- GetNumeric
  - Dataset, [49](#)
  - DatasetInstance, [74](#)
- GetNumericClassLevel
  - ArffDataset, [20](#)
  - Dataset, [50](#)
  - PlinkBinaryDataset, [104](#)
  - PlinkDataset, [112](#)
  - PlinkRawDataset, [117](#)
- GetNumericIndexFromName
  - Dataset, [50](#)
- GetNumericRowCol
  - Dataset, [50](#)
- GetNumericsFilename
  - Dataset, [51](#)
- GetNumericsNames
  - Dataset, [51](#)
- GetNumericValues
  - Dataset, [51](#)
- GetPredictedValueTau
  - DatasetInstance, [75](#)
- GetProbabilityValueGivenClass
  - Dataset, [52](#)
- GetRandomInstance
  - Dataset, [52](#)
- GetRandomJungleScores
  - EvaporativeCooling, [87](#)
- GetReliefFScores
  - EvaporativeCooling, [87](#)
- GetScores
  - RandomJungle, [121](#)
  - ReliefF, [128](#)
- GetSnpsFilename
  - Dataset, [52](#)
- GetTypeOf
  - ArffDataset, [20](#)
- GetVariableNames
  - Dataset, [52](#)
- GSLRandomBase, [90](#)
  - ~GSLRandomBase, [92](#)

- GSLRandomBase, 92
- nextRandVal, 92
- rStatePtr\_, 93
- state, 93
- GSLRandomFlat, 93
  - ~GSLRandomFlat, 96
  - GSLRandomFlat, 96
  - lower\_, 96
  - nextRandVal, 96
  - upper\_, 96
- HasAlternatePhenotypes
  - Dataset, 53
- hasAlternatePhenotypes
  - Dataset, 65
- HasContinuousPhenotypes
  - Dataset, 53
- hasContinuousPhenotypes
  - Dataset, 65
- HasGenotypes
  - Dataset, 53
- hasGenotypes
  - Dataset, 65
- HasNumerics
  - Dataset, 53
- hasNumerics
  - Dataset, 66
- Histogram
  - Statistics.h, 195
- HistogramIt
  - Statistics.h, 195
- insilico, 9
  - best\_n, 10
  - get\_bits, 11
  - join, 11
  - split, 11, 12
  - split\_if, 12, 13
  - to\_lower, 13
  - to\_upper, 13
  - trim, 13, 14
  - trim\_left, 14
  - trim\_right, 14
  - zeroPadNumber, 14
- Insilico.cpp
  - ChooseSnpsDatasetByExtension, 173
  - GetMatchingIds, 173
  - LoadIndividualIds, 173
  - Timestamp, 174
- Insilico.h
  - ChooseSnpsDatasetByExtension, 175
  - COMMAND\_LINE\_ERROR, 177
  - GetMatchingIds, 176
  - LoadIndividualIds, 176
  - Timestamp, 176
- insilico::do\_to\_lower, 79
  - do\_to\_lower, 80
  - m\_ctype, 80
  - operator(), 80
- insilico::do\_to\_upper, 80
  - do\_to\_upper, 81
  - m\_ctype, 81
  - operator(), 81
- insilico::is\_classified, 96
  - is\_classified, 97
  - m\_ctype, 97
  - operator(), 97
- instancelds
  - Dataset, 66
- instanceldsToLoad
  - Dataset, 66
- instances
  - Dataset, 66
- instancesMask
  - Dataset, 66
- instancesMaskPushed
  - Dataset, 66
- INTEGRATED\_ANALYSIS
  - ReliefF.h, 189
- INVALID\_ATTRIBUTE\_VALUE
  - Dataset.h, 151
- INVALID\_DISCRETE\_CLASS\_VALUE
  - Dataset.h, 151
- INVALID\_DISTANCE
  - Dataset.h, 151
- INVALID\_INDEX
  - Dataset.h, 151
- INVALID\_NUMERIC\_CLASS\_VALUE
  - Dataset.h, 151
- INVALID\_NUMERIC\_VALUE
  - Dataset.h, 151
- is\_classified
  - insilico::is\_classified, 97
- IsLoadableInstanceID
  - Dataset, 53
- join
  - insilico, 11
- k

- ReliefF, [130](#)
- KendallTau
  - Statistics.cpp, [193](#)
  - Statistics.h, [197](#)
- levelCounts
  - Dataset, [66](#)
- levelCountsByClass
  - Dataset, [66](#)
- librelieff\_is\_present
  - ReliefF.cpp, [187](#)
  - ReliefF.h, [189](#)
- LoadAlternatePhenotypes
  - Dataset, [53](#)
- LoadDataset
  - Dataset, [54](#)
- LoadIndividualIds
  - Insilico.cpp, [173](#)
  - Insilico.h, [176](#)
- LoadInstanceFromVector
  - DatasetInstance, [75](#)
- LoadNumerics
  - Dataset, [54](#)
- LoadSnps
  - ArffDataset, [20](#)
  - CleanSnpDataset, [30](#)
  - Dataset, [54](#)
  - PlinkBinaryDataset, [105](#)
  - PlinkDataset, [113](#)
  - PlinkRawDataset, [118](#)
- lower\_
  - GSLRandomFlat, [96](#)
- m
  - ReliefF, [130](#)
- m\_ctype
  - insilico::do\_to\_lower, [80](#)
  - insilico::do\_to\_upper, [81](#)
  - insilico::is\_classified, [97](#)
- MAP3\_FILE
  - PlinkDataset.h, [181](#)
- MAP4\_FILE
  - PlinkDataset.h, [181](#)
- MapFileType
  - PlinkDataset.h, [181](#)
- MaskGetAllVariableNames
  - Dataset, [55](#)
- MaskGetAttributeIndices
  - Dataset, [55](#)
- MaskGetAttributeMask
  - Dataset, [55](#)
- MaskGetInstanceIds
  - Dataset, [56](#)
- MaskGetInstanceIndices
  - Dataset, [56](#)
- MaskGetInstanceMask
  - Dataset, [56](#)
- MaskIncludeAllAttributes
  - Dataset, [56](#)
- MaskIncludeAllInstances
  - Dataset, [56](#)
- maskIsPushed
  - Dataset, [67](#)
- MaskPopAll
  - Dataset, [57](#)
- MaskPushAll
  - Dataset, [57](#)
- MaskRemoveAttribute
  - Dataset, [57](#)
- MaskRemoveInstance
  - Dataset, [57](#)
- MaskSearchAttribute
  - Dataset, [58](#)
- MaskSearchInstance
  - Dataset, [58](#)
- MaskWriteNewDataset
  - Dataset, [58](#)
- MISSING\_VALUE
  - Dataset.h, [150](#)
- MISSING\_ATTRIBUTE\_VALUE
  - Dataset.h, [151](#)
- MISSING\_DISCRETE\_CLASS\_VALUE
  - Dataset.h, [151](#)
- MISSING\_NUMERIC\_CLASS\_VALUE
  - Dataset.h, [152](#)
- MISSING\_NUMERIC\_VALUE
  - Dataset.h, [152](#)
- missingAttributeValuesToCheck
  - ArffDataset, [21](#)
  - PlinkBinaryDataset, [106](#)
- missingClassValuesToCheck
  - ArffDataset, [21](#)
  - PlinkBinaryDataset, [106](#)
  - PlinkDataset, [113](#)
- missingNumericValues
  - Dataset, [67](#)
- missingValues
  - Dataset, [67](#)
- neighborInfluenceFactorDs



- DatasetInstance, 78
- nextRandVal
  - GSLRandomBase, 92
  - GSLRandomFlat, 96
- NO\_ANALYSIS
  - ReliefF.h, 189
- NO\_OUTPUT\_DATASET
  - Dataset.h, 150
- NO\_TYPE
  - Dataset.h, 150
- NO\_VALUE
  - Dataset.h, 150
- nominalValues
  - ArffDataset, 21
- norm
  - DistanceMetrics.cpp, 160
  - DistanceMetrics.h, 164
- NumAttributes
  - Dataset, 59
  - DatasetInstance, 75
- numAttributesRead
  - PlinkBinaryDataset, 106
- NumClasses
  - Dataset, 59
- numClasses
  - ChiSquared, 26
- numClassesRead
  - PlinkBinaryDataset, 106
- numDiff
  - ReliefF, 130
- NUMERIC\_ONLY\_ANALYSIS
  - ReliefF.h, 189
- NUMERIC\_TYPE
  - Dataset.h, 150
- NUMERIC\_VALUE
  - Dataset.h, 150
- NumericLevel
  - DatasetInstance.h, 155
- numerics
  - DatasetInstance, 78
- numericsFilename
  - Dataset, 67
- numericsIds
  - Dataset, 67
- numericsMask
  - Dataset, 67
- numericsMaskPushed
  - Dataset, 67
- numericsMinMax
  - Dataset, 67
- numericsNames
  - Dataset, 68
- NumInstances
  - Dataset, 59
- numInstancesRead
  - PlinkBinaryDataset, 106
- NumLevels
  - Dataset, 59
- numLevels
  - ChiSquared, 26
- numMetric
  - ReliefF, 131
- NumNumerics
  - Dataset, 59
  - DatasetInstance, 75
- numRFThreads
  - EvaporativeCooling, 89
- numTargetAttributes
  - EvaporativeCooling, 89
- numToRemovePerIteration
  - EvaporativeCooling, 89
- NumVariables
  - Dataset, 59
- observedFreqTable
  - ChiSquared, 26
- one\_over\_m\_times\_k
  - ReliefF, 131
- operator()
  - deref\_less, 79
  - deref\_less\_bcw, 79
  - insilico::do\_to\_lower, 80
  - insilico::do\_to\_upper, 81
  - insilico::is\_classified, 97
- outFilesPrefix
  - EvaporativeCooling, 89
- OutputDatasetType
  - Dataset.h, 150
- paramsMap
  - EvaporativeCooling, 89
- phenotypesIds
  - Dataset, 68
- PlinkBinaryDataset, 98
  - ~PlinkBinaryDataset, 102
  - filenameBase, 106
  - GetAttributeLevel, 102
  - GetAttributeMAF, 102
  - GetAttributeMutationType, 103
  - GetAttributeValueType, 103

- GetClassValueType, [104](#)
- GetDiscreteClassLevel, [104](#)
- GetNumericClassLevel, [104](#)
- LoadSnps, [105](#)
- missingAttributeValuesToCheck, [106](#)
- missingClassValuesToCheck, [106](#)
- numAttributesRead, [106](#)
- numClassesRead, [106](#)
- numInstancesRead, [106](#)
- PlinkBinaryDataset, [102](#)
- ReadBimFile, [105](#)
- ReadFamFile, [105](#)
- validAttributeValues, [106](#)
- PlinkDataset, [107](#)
  - ~PlinkDataset, [111](#)
  - filenameBase, [113](#)
  - GetAttributeMAF, [111](#)
  - GetAttributeMutationType, [111](#)
  - GetClassValueType, [112](#)
  - GetDiscreteClassLevel, [112](#)
  - GetNumericClassLevel, [112](#)
  - LoadSnps, [113](#)
  - missingClassValuesToCheck, [113](#)
  - PlinkDataset, [111](#)
- PlinkDataset.h
  - ERROR\_FILE, [181](#)
  - MAP3\_FILE, [181](#)
  - MAP4\_FILE, [181](#)
  - MapFileType, [181](#)
- PlinkRawDataset, [113](#)
  - ~PlinkRawDataset, [116](#)
  - GetClassValueType, [117](#)
  - GetDiscreteClassLevel, [117](#)
  - GetNumericClassLevel, [117](#)
  - LoadSnps, [118](#)
  - PlinkRawDataset, [116](#)
- PreComputeDistances
  - ReliefF, [128](#)
- PreComputeDistancesByMap
  - ReliefF, [129](#)
- predictedValueTau
  - DatasetInstance, [78](#)
- PrepareForAttribute
  - ChiSquared, [25](#)
- Print
  - Dataset, [60](#)
  - DatasetInstance, [75](#)
- PrintAllScoresTabular
  - EvaporativeCooling, [87](#)
- PrintAttributeLevelsSeen
  - Dataset, [60](#)
- PrintAttributeScores
  - EvaporativeCooling, [87](#)
  - ReliefF, [129](#)
- PrintClassIndexInfo
  - Dataset, [60](#)
- PrintDistancePairs
  - DatasetInstance, [75](#)
- PrintKendallTaus
  - EvaporativeCooling, [87](#)
- PrintLevelCounts
  - Dataset, [60](#)
- PrintMaskStats
  - Dataset, [60](#)
- PrintMissingValuesStats
  - Dataset, [60](#)
- PrintNominalsMapping
  - ArffDataset, [20](#)
- PrintNumericsStats
  - Dataset, [60](#)
- PrintRecodeMap
  - Dataset, [60](#)
- PrintScores
  - ChiSquared, [25](#)
- PrintStats
  - Dataset, [61](#)
- PrintStatsSimple
  - Dataset, [61](#)
- PrintTables
  - ChiSquared, [25](#)
- PrintVector
  - Debugging.h, [156](#), [157](#)
- ProcessExclusionFile
  - ReliefF, [129](#)
- RandomJungle, [118](#)
  - ~RandomJungle, [121](#)
  - ComputeAttributeScores, [121](#)
  - dataset, [121](#)
  - GetScores, [121](#)
  - RandomJungle, [121](#)
  - ReadScores, [121](#)
  - rjParams, [121](#)
  - scores, [122](#)
- randomJungle
  - EvaporativeCooling, [89](#)
- randomlySelect
  - ReliefF, [131](#)
- ReadBimFile
  - PlinkBinaryDataset, [105](#)

- ReadFamFile
  - PlinkBinaryDataset, 105
- ReadScores
  - RandomJungle, 121
- REGRESSION\_ANALYSIS
  - ReliefF.h, 189
- relationName
  - ArffDataset, 21
- ReliefF, 122
  - ~ReliefF, 127
  - analysisType, 130
  - ComputeAttributeScores, 127
  - ComputeAttributeScoresCleanSnps, 128
  - ComputeAttributeScoresIteratively, 128
  - ComputeInstanceToInstanceDistance, 128
  - ComputeWeightByDistanceFactors, 128
  - dataset, 130
  - doRemovePercent, 130
  - finalScores, 130
  - GetScores, 128
  - k, 130
  - m, 130
  - numDiff, 130
  - numMetric, 131
  - one\_over\_m\_times\_k, 131
  - PreComputeDistances, 128
  - PreComputeDistancesByMap, 129
  - PrintAttributeScores, 129
  - ProcessExclusionFile, 129
  - randomlySelect, 131
  - ReliefF, 127
  - removePercentage, 131
  - removePerIteration, 131
  - ResetForNextIteration, 129
  - scoreNames, 131
  - snpDiff, 131
  - snpMetric, 132
  - W, 132
  - weightByDistanceMethod, 132
  - weightByDistanceSigma, 132
  - WriteAttributeScores, 129
- reliefF
  - EvaporativeCooling, 90
- ReliefF.cpp
  - AttributeIndex, 186
  - AttributeIndexIt, 186
  - attributeSort, 187
  - librelieff\_is\_present, 187
  - ScoresMap, 186
- ScoresMapIt, 186
  - scoreSort, 187
  - T, 186
- ReliefF.h
  - AnalysisType, 189
  - DIAGNOSTIC\_ANALYSIS, 189
  - INTEGRATED\_ANALYSIS, 189
  - librelieff\_is\_present, 189
  - NO\_ANALYSIS, 189
  - NUMERIC\_ONLY\_ANALYSIS, 189
  - REGRESSION\_ANALYSIS, 189
  - SNP\_CLEAN\_ANALYSIS, 189
  - SNP\_ONLY\_ANALYSIS, 189
- removePercentage
  - ReliefF, 131
- removePerIteration
  - ReliefF, 131
- RemoveWorstAttributes
  - EvaporativeCooling, 87
- ResetForNextIteration
  - ReliefF, 129
- rfScores
  - EvaporativeCooling, 90
- rjParams
  - RandomJungle, 121
- rjScores
  - EvaporativeCooling, 90
- rng
  - Dataset, 68
- RReliefF, 133
  - ~RReliefF, 137
  - ComputeAttributeScores, 137
  - RReliefF, 136
- rStatePtr\_
  - GSLRandomBase, 93
- RunReliefF
  - EvaporativeCooling, 88
- RunSnpDiagnosticTests
  - Dataset, 61
- scoreNames
  - ReliefF, 131
- scores
  - ChiSquared, 27
  - RandomJungle, 122
- ScoresMap
  - ReliefF.cpp, 186
- ScoresMapIt
  - ReliefF.cpp, 186
- scoreSort

- ReliefF.cpp, 187
- scoresSortAsc
  - EvaporativeCooling.cpp, 165
- scoresSortAscByName
  - EvaporativeCooling.cpp, 165
- scoresSortDesc
  - EvaporativeCooling.cpp, 165
- SetClass
  - DatasetInstance, 76
- SetDistanceSums
  - DatasetInstance, 76
- SetPredictedValueTau
  - DatasetInstance, 76
- SNP\_CLEAN\_ANALYSIS
  - ReliefF.h, 189
- SNP\_ONLY\_ANALYSIS
  - ReliefF.h, 189
- snpDiff
  - ReliefF, 131
- SNPHWE
  - Dataset, 61
- snpMetric
  - ReliefF, 132
- snpsFilename
  - Dataset, 68
- split
  - insilico, 11, 12
- split\_if
  - insilico, 12, 13
- src/library/ArffDataset.cpp, 139
- src/library/ArffDataset.h, 140
- src/library/best\_n.h, 141
- src/library/ChiSquared.cpp, 143
- src/library/ChiSquared.h, 144
- src/library/CleanSnpDataset.cpp, 145
- src/library/CleanSnpDataset.h, 145
- src/library/Dataset.cpp, 146
- src/library/Dataset.h, 148
- src/library/DatasetInstance.cpp, 152
- src/library/DatasetInstance.h, 153
- src/library/Debugging.h, 155
- src/library/DistanceMetrics.cpp, 157
- src/library/DistanceMetrics.h, 161
- src/library/EvaporativeCooling.cpp, 164
- src/library/EvaporativeCooling.h, 166
- src/library/FilesystemUtils.cpp, 168
- src/library/FilesystemUtils.h, 169
- src/library/GSLRandomBase.h, 170
- src/library/GSLRandomFlat.h, 171
- src/library/Insilico.cpp, 172
- src/library/Insilico.h, 174
- src/library/PlinkBinaryDataset.cpp, 177
- src/library/PlinkBinaryDataset.h, 178
- src/library/PlinkDataset.cpp, 179
- src/library/PlinkDataset.h, 179
- src/library/PlinkRawDataset.cpp, 181
- src/library/PlinkRawDataset.h, 181
- src/library/RandomJungle.cpp, 182
- src/library/RandomJungle.h, 183
- src/library/ReliefF.cpp, 184
- src/library/ReliefF.h, 187
- src/library/RReliefF.cpp, 189
- src/library/RReliefF.h, 190
- src/library/Statistics.cpp, 191
- src/library/Statistics.h, 194
- src/library/StringUtils.h, 198
- state
  - GSLRandomBase, 93
- Statistics.cpp
  - ConditionalEntropy, 193
  - ConstructAttributeCart, 193
  - DEBUG\_E, 192
  - DEBUG\_Z, 192
  - Entropy, 193
  - KendallTau, 193
  - ZTransform, 193
- Statistics.h
  - ConditionalEntropy, 196
  - ConstructAttributeCart, 196
  - Entropy, 196
  - Histogram, 195
  - HistogramIlt, 195
  - KendallTau, 197
  - VarStd, 198
  - VectorDouble, 195
  - VectorDoubleIlt, 196
  - ZTransform, 198
- SwapAttributes
  - Dataset, 62
  - DatasetInstance, 77
- T
  - DatasetInstance.cpp, 153
  - ReliefF.cpp, 186
- TAB\_DELIMITED\_DATASET
  - Dataset.h, 150
- Timestamp
  - Insilico.cpp, 174
  - Insilico.h, 176
- to\_lower

- insilico, [13](#)
- to\_upper
  - insilico, [13](#)
- TRANSITION\_MUTATION
  - Dataset.h, [150](#)
- TRANSVERSION\_MUTATION
  - Dataset.h, [150](#)
- trim
  - insilico, [13](#), [14](#)
- trim\_left
  - insilico, [14](#)
- trim\_right
  - insilico, [14](#)
- UNKNOWN\_MUTATION
  - Dataset.h, [150](#)
- UpdateAllLevelCounts
  - Dataset, [62](#)
- UpdateLevelCounts
  - Dataset, [62](#)
- upper\_
  - GSLRandomFlat, [96](#)
- validAttributeValues
  - PlinkBinaryDataset, [106](#)
- ValueType
  - Dataset.h, [150](#)
- VarStd
  - Statistics.h, [198](#)
- VectorDouble
  - Statistics.h, [195](#)
- VectorDoublelt
  - Statistics.h, [196](#)
- W
  - ReliefF, [132](#)
- weightByDistanceMethod
  - ReliefF, [132](#)
- weightByDistanceSigma
  - ReliefF, [132](#)
- WriteAttributeScores
  - EvaporativeCooling, [88](#)
  - ReliefF, [129](#)
- WriteLevelCounts
  - Dataset, [62](#)
- WriteNewDataset
  - Dataset, [63](#)
- WriteScores
  - ChiSquared, [26](#)
- zeroPadNumber

- insilico, [14](#)
- ZTransform
  - Statistics.cpp, [193](#)
  - Statistics.h, [198](#)