barry: Your go-to motif accountant

0.0-1

Generated by Doxygen 1.9.1

1 Main Page 1

2 Module Index 3

2.1 Modules . . . 3

3 Class Index 5

3.1 Class List . . . 5

4 File Index 7

4.1 File List . . . 7

5 Module Documentation 9

5.1 Counting . . . 9

5.1.1 Detailed Description . . . 9

5.2 Statistical Models . . . 9

5.2.1 Detailed Description . . . 10

5.3 Network counters . . . 10

5.3.1 Detailed Description . . . 10

5.3.2 Function Documentation . . . 11

5.3.2.1 counter_absdiff() . . . 11

5.3.2.2 counter_ctriads() . . . 11

5.3.2.3 counter_degree() . . . 11

5.3.2.4 counter_density() . . . 11

5.3.2.5 counter_diff() . . . 12

5.3.2.6 counter_edges() . . . 12

5.3.2.7 counter_idegree() . . . 12

5.3.2.8 counter_idegree15() . . . 12

5.3.2.9 counter_isolates() . . . 12

5.3.2.10 counter_istar2() . . . 13

5.3.2.11 counter_mutual() . . . 13

5.3.2.12 counter_nodecov() . . . 13

5.3.2.13 counter_nodeicov() . . . 13

5.3.2.14 counter_nodematch() . . . 13

5.3.2.15 counter_nodeocov() . . . 14

5.3.2.16 counter_odegree() . . . 14

5.3.2.17 counter_odegree15() . . . 14

5.3.2.18 counter_ostar2() . . . 14

5.3.2.19 counter_ttriads() . . . 14

5.3.2.20 NETWORK_COUNTER() . . . 15

5.4 Phylo counters . . . 15

5.4.1 Detailed Description . . . 15

5.4.2 Function Documentation . . . 16

5.4.2.1 counter_co_opt() . . . 16

5.4.2.2 counter_cogain() . . . 16

5.4.2.3 counter_gains() . . . 17
5.4.2.4 counter_gains_k_offspring() . . . 17
5.4.2.5 counter_genes_changing() . . . 17
5.4.2.6 counter_longest() . . . 17
5.4.2.7 counter_loss() . . . 18
5.4.2.8 counter_maxfuns() . . . 18
5.4.2.9 counter_neofun() . . . 18
5.4.2.10 counter_neofun_a2b() . . . 18
5.4.2.11 counter_overall_changes() . . . 19
5.4.2.12 counter_overall_gains() . . . 19
5.4.2.13 counter_overall_loss() . . . 19
5.4.2.14 counter_subfun() . . . 19

6 Namespace Documentation **21**
6.1 barry Namespace Reference . . . 21
6.1.1 Detailed Description . . . 21
6.2 barry::counters Namespace Reference . . . 21
6.2.1 Detailed Description . . . 21
6.3 barry::counters::network Namespace Reference . . . 22
6.4 barry::counters::phylo Namespace Reference . . . 22
6.5 CHECK Namespace Reference . . . 22
6.5.1 Detailed Description . . . 22
6.5.2 Variable Documentation . . . 22
6.5.2.1 BOTH . . . 22
6.5.2.2 NONE . . . 22
6.5.2.3 ONE . . . 22
6.5.2.4 TWO . . . 23
6.6 EXISTS Namespace Reference . . . 23
6.6.1 Detailed Description . . . 23
6.6.2 Variable Documentation . . . 23
6.6.2.1 AS_ONE . . . 23
6.6.2.2 AS_ZERO . . . 23
6.6.2.3 BOTH . . . 24
6.6.2.4 NONE . . . 24
6.6.2.5 ONE . . . 24
6.6.2.6 TWO . . . 24
6.6.2.7 UKNOWN . . . 24

7 Class Documentation **25**
7.1 BArray< Cell_Type, Data_Type > Class Template Reference . . . 25
7.1.1 Detailed Description . . . 27
7.1.2 Constructor & Destructor Documentation . . . 28
7.1.2.1 BArray() [1/6] . . . 28

7.1.2.2 BArray() [2/6] . . . 28
7.1.2.3 BArray() [3/6] . . . 28
7.1.2.4 BArray() [4/6] . . . 29
7.1.2.5 BArray() [5/6] . . . 29
7.1.2.6 BArray() [6/6] . . . 29
7.1.2.7 ∼BArray() . . . 29
7.1.3 Member Function Documentation . . . 29
7.1.3.1 clear() . . . 29
7.1.3.2 col() . . . 30
7.1.3.3 D() [1/2] . . . 30
7.1.3.4 D() [2/2] . . . 30
7.1.3.5 default_val() . . . 30
7.1.3.6 get_cell() . . . 30
7.1.3.7 get_col() . . . 30
7.1.3.8 get_col_vec() [1/2] . . . 31
7.1.3.9 get_col_vec() [2/2] . . . 31
7.1.3.10 get_entries() . . . 31
7.1.3.11 get_row() . . . 31
7.1.3.12 get_row_vec() [1/2] . . . 31
7.1.3.13 get_row_vec() [2/2] . . . 32
7.1.3.14 insert_cell() [1/3] . . . 32
7.1.3.15 insert_cell() [2/3] . . . 32
7.1.3.16 insert_cell() [3/3] . . . 32
7.1.3.17 is_empty() . . . 32
7.1.3.18 ncol() . . . 33
7.1.3.19 nnozero() . . . 33
7.1.3.20 nrow() . . . 33
7.1.3.21 operator()() [1/2] . . . 33
7.1.3.22 operator()() [2/2] . . . 33
7.1.3.23 operator∗=() . . . 33
7.1.3.24 operator+=() [1/3] . . . 34
7.1.3.25 operator+=() [2/3] . . . 34
7.1.3.26 operator+=() [3/3] . . . 34
7.1.3.27 operator-=() [1/3] . . . 34
7.1.3.28 operator-=() [2/3] . . . 34
7.1.3.29 operator-=() [3/3] . . . 34
7.1.3.30 operator/=() . . . 35
7.1.3.31 operator=() [1/2] . . . 35
7.1.3.32 operator=() [2/2] . . . 35
7.1.3.33 operator==() . . . 35
7.1.3.34 out_of_range() . . . 35
7.1.3.35 print() . . . 35

7.1.3.36 reserve() . . . 36
7.1.3.37 resize() . . . 36
7.1.3.38 rm_cell() . . . 36
7.1.3.39 row() . . . 36
7.1.3.40 set_data() . . . 36
7.1.3.41 swap_cells() . . . 37
7.1.3.42 swap_cols() . . . 37
7.1.3.43 swap_rows() . . . 37
7.1.3.44 toggle_cell() . . . 37
7.1.3.45 toggle_lock() . . . 38
7.1.3.46 transpose() . . . 38
7.1.3.47 zero_col() . . . 38
7.1.3.48 zero_row() . . . 38
7.1.4 Friends And Related Function Documentation . . . 38
7.1.4.1 BArrayCell< Cell_Type, Data_Type > . . . 38
7.1.4.2 BArrayCell_const< Cell_Type, Data_Type > . . . 39
7.1.5 Member Data Documentation . . . 39
7.1.5.1 visited . . . 39
7.2 BArrayCell< Cell_Type, Data_Type > Class Template Reference . . . 39
7.2.1 Detailed Description . . . 39
7.2.2 Constructor & Destructor Documentation . . . 40
7.2.2.1 BArrayCell() . . . 40
7.2.2.2 ~BArrayCell() . . . 40
7.2.3 Member Function Documentation . . . 40
7.2.3.1 operator Cell_Type() . . . 40
7.2.3.2 operator*=() . . . 40
7.2.3.3 operator+=() . . . 41
7.2.3.4 operator-=() . . . 41
7.2.3.5 operator/=() . . . 41
7.2.3.6 operator=() . . . 41
7.2.3.7 operator==() . . . 41
7.3 BArrayCell_const< Cell_Type, Data_Type > Class Template Reference . . . 42
7.3.1 Detailed Description . . . 42
7.3.2 Constructor & Destructor Documentation . . . 42
7.3.2.1 BArrayCell_const() . . . 42
7.3.2.2 ~BArrayCell_const() . . . 42
7.3.3 Member Function Documentation . . . 43
7.3.3.1 operator Cell_Type() . . . 43
7.3.3.2 operator"!=() . . . 43
7.3.3.3 operator<() . . . 43
7.3.3.4 operator<=() . . . 43
7.3.3.5 operator==() . . . 43

7.3.3.6 operator>() . . . 44
7.3.3.7 operator>=() . . . 44
7.4 BArrayCol< Cell_Type, Data_Type > Class Template Reference . . . 44
7.4.1 Detailed Description . . . 44
7.4.2 Constructor & Destructor Documentation . . . 45
7.4.2.1 BArrayCol() . . . 45
7.4.2.2 ~BArrayCol() . . . 45
7.4.3 Member Function Documentation . . . 45
7.4.3.1 begin() . . . 45
7.4.3.2 end() . . . 45
7.4.3.3 operator Cell_Type() . . . 46
7.4.3.4 operator*=() . . . 46
7.4.3.5 operator+=() . . . 46
7.4.3.6 operator-=() . . . 46
7.4.3.7 operator/=() . . . 46
7.4.3.8 operator=() . . . 47
7.4.3.9 operator==() . . . 47
7.5 BArrayCol_const< Cell_Type, Data_Type > Class Template Reference . . . 47
7.5.1 Detailed Description . . . 47
7.5.2 Constructor & Destructor Documentation . . . 48
7.5.2.1 ~BArrayCol_const() . . . 48
7.5.3 Member Function Documentation . . . 48
7.5.3.1 BArrayCol() . . . 48
7.5.3.2 operator"!=() . . . 48
7.5.3.3 operator<() . . . 48
7.5.3.4 operator<=() . . . 49
7.5.3.5 operator==() . . . 49
7.5.3.6 operator>() . . . 49
7.5.3.7 operator>=() . . . 49
7.6 Cell< Cell_Type > Class Template Reference . . . 49
7.6.1 Detailed Description . . . 50
7.6.2 Constructor & Destructor Documentation . . . 50
7.6.2.1 Cell() [1/7] . . . 50
7.6.2.2 Cell() [2/7] . . . 51
7.6.2.3 ~Cell() . . . 51
7.6.2.4 Cell() [3/7] . . . 51
7.6.2.5 Cell() [4/7] . . . 51
7.6.2.6 Cell() [5/7] . . . 51
7.6.2.7 Cell() [6/7] . . . 52
7.6.2.8 Cell() [7/7] . . . 52
7.6.3 Member Function Documentation . . . 52
7.6.3.1 add() [1/4] . . . 52

7.6.3.2 add() [2/4] . . . 52
7.6.3.3 add() [3/4] . . . 52
7.6.3.4 add() [4/4] . . . 53
7.6.3.5 operator Cell_Type() . . . 53
7.6.3.6 operator=() [1/2] . . . 53
7.6.3.7 operator=() [2/2] . . . 53
7.6.4 Member Data Documentation . . . 53
7.6.4.1 value . . . 53
7.6.4.2 visited . . . 54
7.7 ConstBArrayRowIter< Cell_Type, Data_Type > Class Template Reference . . . 54
7.7.1 Detailed Description . . . 55
7.7.2 Constructor & Destructor Documentation . . . 55
7.7.2.1 ConstBArrayRowIter() . . . 55
7.7.2.2 ~ConstBArrayRowIter() . . . 55
7.7.3 Member Data Documentation . . . 55
7.7.3.1 Array . . . 55
7.7.3.2 current_col . . . 55
7.7.3.3 current_row . . . 56
7.7.3.4 iter . . . 56
7.8 Counter< Array_Type, Data_Type > Class Template Reference . . . 56
7.8.1 Detailed Description . . . 57
7.8.2 Constructor & Destructor Documentation . . . 57
7.8.2.1 Counter() [1/3] . . . 57
7.8.2.2 Counter() [2/3] . . . 57
7.8.2.3 Counter() [3/3] . . . 58
7.8.2.4 ~Counter() . . . 58
7.8.3 Member Function Documentation . . . 58
7.8.3.1 count() . . . 58
7.8.3.2 init() . . . 58
7.8.3.3 operator=() . . . 58
7.8.4 Member Data Documentation . . . 59
7.8.4.1 count_fun . . . 59
7.8.4.2 data . . . 59
7.8.4.3 delete_data . . . 59
7.8.4.4 desc . . . 59
7.8.4.5 init_fun . . . 59
7.8.4.6 name . . . 60
7.9 Counters< Array_Type, Data_Type > Class Template Reference . . . 60
7.9.1 Detailed Description . . . 60
7.9.2 Constructor & Destructor Documentation . . . 61
7.9.2.1 Counters() [1/2] . . . 61
7.9.2.2 ~Counters() . . . 61

7.9.2.3 Counters() [2/2] . . . 61
7.9.3 Member Function Documentation . . . 61
7.9.3.1 add_counter() [1/3] . . . 61
7.9.3.2 add_counter() [2/3] . . . 62
7.9.3.3 add_counter() [3/3] . . . 62
7.9.3.4 clear() . . . 62
7.9.3.5 operator=() . . . 62
7.9.3.6 operator[]() . . . 62
7.9.3.7 size() . . . 63
7.10 Entries< Cell_Type > Class Template Reference . . . 63
7.10.1 Detailed Description . . . 63
7.10.2 Constructor & Destructor Documentation . . . 64
7.10.2.1 Entries() [1/2] . . . 64
7.10.2.2 Entries() [2/2] . . . 64
7.10.2.3 ~Entries() . . . 64
7.10.3 Member Function Documentation . . . 64
7.10.3.1 resize() . . . 64
7.10.4 Member Data Documentation . . . 65
7.10.4.1 source . . . 65
7.10.4.2 target . . . 65
7.10.4.3 val . . . 65
7.11 Flock Class Reference . . . 65
7.11.1 Detailed Description . . . 66
7.11.2 Constructor & Destructor Documentation . . . 66
7.11.2.1 Flock() . . . 66
7.11.2.2 ~Flock() . . . 67
7.11.3 Member Function Documentation . . . 67
7.11.3.1 add_data() . . . 67
7.11.3.2 counters_ptr() . . . 67
7.11.3.3 init() . . . 67
7.11.3.4 likelihood_joint() . . . 68
7.11.3.5 nfuns() . . . 68
7.11.3.6 nleafs() . . . 68
7.11.3.7 nnodes() . . . 68
7.11.3.8 nterms() . . . 69
7.11.3.9 ntrees() . . . 69
7.11.3.10 operator()() . . . 69
7.11.3.11 set_seed() . . . 69
7.11.4 Member Data Documentation . . . 70
7.11.4.1 dat . . . 70
7.11.4.2 initialized . . . 70
7.11.4.3 nfunctions . . . 70

7.11.4.4 rengine . . . 70
7.11.4.5 support . . . 70
7.12 FreqTable< T > Class Template Reference . . . 71
7.12.1 Detailed Description . . . 71
7.12.2 Constructor & Destructor Documentation . . . 71
7.12.2.1 FreqTable() . . . 71
7.12.2.2 ~FreqTable() . . . 71
7.12.3 Member Function Documentation . . . 72
7.12.3.1 add() . . . 72
7.12.3.2 as_vector() . . . 72
7.12.3.3 clear() . . . 72
7.12.3.4 get_data() . . . 72
7.12.3.5 get_data_ptr() . . . 72
7.12.3.6 print() . . . 73
7.12.3.7 reserve() . . . 73
7.12.3.8 size() . . . 73
7.13 Geese Class Reference . . . 73
7.13.1 Detailed Description . . . 75
7.13.2 Constructor & Destructor Documentation . . . 75
7.13.2.1 Geese() [1/4] . . . 75
7.13.2.2 Geese() [2/4] . . . 75
7.13.2.3 Geese() [3/4] . . . 76
7.13.2.4 Geese() [4/4] . . . 76
7.13.2.5 ~Geese() . . . 76
7.13.3 Member Function Documentation . . . 76
7.13.3.1 calc_reduced_sequence() . . . 76
7.13.3.2 calc_sequence() . . . 76
7.13.3.3 get_probabilities() . . . 77
7.13.3.4 inherit_support() . . . 77
7.13.3.5 init() . . . 77
7.13.3.6 init_node() . . . 77
7.13.3.7 likelihood() . . . 77
7.13.3.8 likelihood_exhaust() . . . 78
7.13.3.9 nfuns() . . . 78
7.13.3.10 nleafs() . . . 78
7.13.3.11 nnodes() . . . 78
7.13.3.12 nterms() . . . 78
7.13.3.13 observed_counts() . . . 78
7.13.3.14 operator=() [1/2] . . . 79
7.13.3.15 operator=() [2/2] . . . 79
7.13.3.16 predict() . . . 79
7.13.3.17 predict_backend() . . . 79

7.13.3.18 print_observed_counts() . . . 79
7.13.3.19 set_seed() . . . 80
7.13.3.20 simulate() . . . 80
7.13.3.21 update_annotations() . . . 80
7.13.4 Member Data Documentation . . . 80
7.13.4.1 counters . . . 80
7.13.4.2 delete_counters . . . 80
7.13.4.3 delete_rengine . . . 81
7.13.4.4 delete_support . . . 81
7.13.4.5 initialized . . . 81
7.13.4.6 map_to_nodes . . . 81
7.13.4.7 nfunctions . . . 81
7.13.4.8 nodes . . . 81
7.13.4.9 reduced_sequence . . . 82
7.13.4.10 rengine . . . 82
7.13.4.11 sequence . . . 82
7.13.4.12 states . . . 82
7.13.4.13 support . . . 82
7.14 Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type > Class Template Reference . . . 83
7.14.1 Detailed Description . . . 86
7.14.2 Constructor & Destructor Documentation . . . 86
7.14.2.1 Model() [1/3] . . . 86
7.14.2.2 Model() [2/3] . . . 86
7.14.2.3 Model() [3/3] . . . 87
7.14.2.4 ∼Model() . . . 87
7.14.3 Member Function Documentation . . . 87
7.14.3.1 add_array() . . . 87
7.14.3.2 add_counter() [1/3] . . . 88
7.14.3.3 add_counter() [2/3] . . . 88
7.14.3.4 add_counter() [3/3] . . . 88
7.14.3.5 add_rule() [1/3] . . . 88
7.14.3.6 add_rule() [2/3] . . . 89
7.14.3.7 add_rule() [3/3] . . . 89
7.14.3.8 add_rule_dyn() [1/3] . . . 89
7.14.3.9 add_rule_dyn() [2/3] . . . 89
7.14.3.10 add_rule_dyn() [3/3] . . . 90
7.14.3.11 get_norm_const() . . . 90
7.14.3.12 get_pset() . . . 90
7.14.3.13 get_stats() . . . 90
7.14.3.14 likelihood() [1/3] . . . 91
7.14.3.15 likelihood() [2/3] . . . 91

7.14.3.16 likelihood() [3/3] . . . 91
7.14.3.17 likelihood_total() . . . 91
7.14.3.18 nterms() . . . 92
7.14.3.19 operator=() . . . 92
7.14.3.20 print_stats() . . . 92
7.14.3.21 sample() [1/2] . . . 92
7.14.3.22 sample() [2/2] . . . 93
7.14.3.23 set_counters() . . . 93
7.14.3.24 set_keygen() . . . 93
7.14.3.25 set_rengine() . . . 93
7.14.3.26 set_rules() . . . 94
7.14.3.27 set_rules_dyn() . . . 94
7.14.3.28 set_seed() . . . 94
7.14.3.29 size() . . . 94
7.14.3.30 size_unique() . . . 94
7.14.3.31 store_psets() . . . 95
7.14.4 Member Data Documentation . . . 95
7.14.4.1 array_frequency . . . 95
7.14.4.2 arrays2support . . . 95
7.14.4.3 counter_fun . . . 95
7.14.4.4 counters . . . 96
7.14.4.5 delete_rengine . . . 96
7.14.4.6 first_calc_done . . . 96
7.14.4.7 keygen . . . 96
7.14.4.8 keys2support . . . 97
7.14.4.9 n_arrays_per_stats . . . 97
7.14.4.10 normalizing_constants . . . 97
7.14.4.11 params_last . . . 97
7.14.4.12 pset_arrays . . . 98
7.14.4.13 pset_probs . . . 98
7.14.4.14 pset_stats . . . 98
7.14.4.15 rengine . . . 98
7.14.4.16 rules . . . 98
7.14.4.17 rules_dyn . . . 99
7.14.4.18 stats . . . 99
7.14.4.19 support_fun . . . 99
7.14.4.20 target_stats . . . 99
7.14.4.21 with_pset . . . 99
7.15 NetCounterData Class Reference . . . 100
7.15.1 Detailed Description . . . 100
7.15.2 Constructor & Destructor Documentation . . . 100
7.15.2.1 NetCounterData() [1/2] . . . 100

7.15.2.2 NetCounterData() [2/2] . . . 100
7.15.2.3 ~NetCounterData() . . . 101
7.15.3 Member Data Documentation . . . 101
7.15.3.1 indices . . . 101
7.15.3.2 numbers . . . 101
7.16 NetworkData Class Reference . . . 101
7.16.1 Detailed Description . . . 102
7.16.2 Constructor & Destructor Documentation . . . 102
7.16.2.1 NetworkData() [1/3] . . . 102
7.16.2.2 NetworkData() [2/3] . . . 102
7.16.2.3 NetworkData() [3/3] . . . 102
7.16.2.4 ~NetworkData() . . . 103
7.16.3 Member Data Documentation . . . 103
7.16.3.1 directed . . . 103
7.16.3.2 vertex_attr . . . 103
7.17 Node Class Reference . . . 103
7.17.1 Detailed Description . . . 104
7.17.2 Constructor & Destructor Documentation . . . 104
7.17.2.1 Node() [1/5] . . . 105
7.17.2.2 Node() [2/5] . . . 105
7.17.2.3 Node() [3/5] . . . 105
7.17.2.4 Node() [4/5] . . . 105
7.17.2.5 Node() [5/5] . . . 105
7.17.2.6 ~Node() . . . 106
7.17.3 Member Function Documentation . . . 106
7.17.3.1 get_parent() . . . 106
7.17.3.2 is_leaf() . . . 106
7.17.4 Member Data Documentation . . . 106
7.17.4.1 annotations . . . 106
7.17.4.2 array . . . 106
7.17.4.3 arrays . . . 107
7.17.4.4 duplication . . . 107
7.17.4.5 id . . . 107
7.17.4.6 narray . . . 107
7.17.4.7 offspring . . . 107
7.17.4.8 ord . . . 108
7.17.4.9 parent . . . 108
7.17.4.10 probability . . . 108
7.17.4.11 subtree_prob . . . 108
7.17.4.12 visited . . . 108
7.18 NodeData Class Reference . . . 109
7.18.1 Detailed Description . . . 109

7.18.2 Constructor & Destructor Documentation . . . 109
7.18.2.1 NodeData() [1/2] . . . 109
7.18.2.2 NodeData() [2/2] . . . 109
7.18.2.3 ~NodeData() . . . 110
7.18.3 Member Data Documentation . . . 110
7.18.3.1 blengths . . . 110
7.18.3.2 duplication . . . 110
7.18.3.3 states . . . 110
7.19 PowerSet< Array_Type, Data_Rule_Type > Class Template Reference . . . 111
7.19.1 Detailed Description . . . 112
7.19.2 Constructor & Destructor Documentation . . . 112
7.19.2.1 PowerSet() [1/3] . . . 112
7.19.2.2 PowerSet() [2/3] . . . 112
7.19.2.3 PowerSet() [3/3] . . . 113
7.19.2.4 ~PowerSet() . . . 113
7.19.3 Member Function Documentation . . . 113
7.19.3.1 add_rule() [1/3] . . . 113
7.19.3.2 add_rule() [2/3] . . . 113
7.19.3.3 add_rule() [3/3] . . . 113
7.19.3.4 begin() . . . 114
7.19.3.5 calc() . . . 114
7.19.3.6 end() . . . 114
7.19.3.7 get_data() . . . 114
7.19.3.8 get_data_ptr() . . . 114
7.19.3.9 init_support() . . . 115
7.19.3.10 operator[]() . . . 115
7.19.3.11 reset() . . . 115
7.19.3.12 size() . . . 115
7.19.4 Member Data Documentation . . . 115
7.19.4.1 coordinates_free . . . 115
7.19.4.2 coordinates_locked . . . 116
7.19.4.3 data . . . 116
7.19.4.4 EmptyArray . . . 116
7.19.4.5 M . . . 116
7.19.4.6 N . . . 116
7.19.4.7 rules . . . 117
7.19.4.8 rules_deleted . . . 117
7.20 Rule< Array_Type, Data_Type > Class Template Reference . . . 117
7.20.1 Detailed Description . . . 118
7.20.2 Constructor & Destructor Documentation . . . 118
7.20.2.1 Rule() [1/2] . . . 118
7.20.2.2 Rule() [2/2] . . . 118

7.20.2.3 ~Rule() . . . 118
7.20.3 Member Function Documentation . . . 119
7.20.3.1 D() . . . 119
7.20.3.2 operator()() . . . 119
7.21 Rules< Array_Type, Data_Type > Class Template Reference . . . 119
7.21.1 Detailed Description . . . 120
7.21.2 Constructor & Destructor Documentation . . . 120
7.21.2.1 Rules() [1/2] . . . 120
7.21.2.2 Rules() [2/2] . . . 120
7.21.2.3 ~Rules() . . . 121
7.21.3 Member Function Documentation . . . 121
7.21.3.1 add_rule() [1/3] . . . 121
7.21.3.2 add_rule() [2/3] . . . 121
7.21.3.3 add_rule() [3/3] . . . 121
7.21.3.4 clear() . . . 121
7.21.3.5 get_seq() . . . 121
7.21.3.6 operator()() . . . 122
7.21.3.7 operator=() . . . 122
7.21.3.8 size() . . . 123
7.22 StatsCounter< Array_Type, Data_Type > Class Template Reference . . . 123
7.22.1 Detailed Description . . . 124
7.22.2 Constructor & Destructor Documentation . . . 124
7.22.2.1 StatsCounter() [1/2] . . . 124
7.22.2.2 StatsCounter() [2/2] . . . 124
7.22.2.3 ~StatsCounter() . . . 125
7.22.3 Member Function Documentation . . . 125
7.22.3.1 add_counter() [1/2] . . . 125
7.22.3.2 add_counter() [2/2] . . . 125
7.22.3.3 count_all() . . . 125
7.22.3.4 count_current() . . . 125
7.22.3.5 count_init() . . . 126
7.22.3.6 reset_array() . . . 126
7.22.3.7 set_counters() . . . 126
7.22.4 Member Data Documentation . . . 126
7.22.4.1 Array . . . 126
7.22.4.2 counter_deleted . . . 127
7.22.4.3 counters . . . 127
7.22.4.4 current_stats . . . 127
7.22.4.5 EmptyArray . . . 127
7.23 Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type > Class Template Reference . . . 128
7.23.1 Detailed Description . . . 130

7.23.2 Constructor & Destructor Documentation . . . 130
7.23.2.1 Support() [1/3] . . . 130
7.23.2.2 Support() [2/3] . . . 130
7.23.2.3 Support() [3/3] . . . 131
7.23.2.4 ~Support() . . . 131
7.23.3 Member Function Documentation . . . 131
7.23.3.1 add_counter() [1/2] . . . 131
7.23.3.2 add_counter() [2/2] . . . 131
7.23.3.3 add_rule() [1/2] . . . 132
7.23.3.4 add_rule() [2/2] . . . 132
7.23.3.5 add_rule_dyn() [1/2] . . . 132
7.23.3.6 add_rule_dyn() [2/2] . . . 132
7.23.3.7 calc() . . . 132
7.23.3.8 get_counts() . . . 133
7.23.3.9 get_counts_ptr() . . . 133
7.23.3.10 get_current_stats() . . . 133
7.23.3.11 init_support() . . . 133
7.23.3.12 print() . . . 134
7.23.3.13 reset_array() [1/2] . . . 134
7.23.3.14 reset_array() [2/2] . . . 134
7.23.3.15 set_counters() . . . 134
7.23.3.16 set_rules() . . . 134
7.23.3.17 set_rules_dyn() . . . 135
7.23.4 Member Data Documentation . . . 135
7.23.4.1 change_stats . . . 135
7.23.4.2 coordinates_free . . . 135
7.23.4.3 coordinates_locked . . . 135
7.23.4.4 counters . . . 136
7.23.4.5 current_stats . . . 136
7.23.4.6 data . . . 136
7.23.4.7 delete_counters . . . 136
7.23.4.8 delete_rules . . . 137
7.23.4.9 delete_rules_dyn . . . 137
7.23.4.10 EmptyArray . . . 137
7.23.4.11 M . . . 137
7.23.4.12 max_num_elements . . . 138
7.23.4.13 N . . . 138
7.23.4.14 rules . . . 138
7.23.4.15 rules_dyn . . . 138
7.24 vecHasher< T > Struct Template Reference . . . 139
7.24.1 Detailed Description . . . 139
7.24.2 Member Function Documentation . . . 139

7.24.2.1 operator()() . . . 139

8 File Documentation **141**

8.1 include/barry/barray-bones.hpp File Reference . . . 141
8.1.1 Macro Definition Documentation . . . 142
8.1.1.1 BARRAY_BONES_HPP . . . 142
8.2 include/barry/barray-iterator.hpp File Reference . . . 142
8.2.1 Macro Definition Documentation . . . 143
8.2.1.1 BARRAY_ITERATOR_HPP . . . 143
8.3 include/barry/barray-meat-operators.hpp File Reference . . . 143
8.3.1 Macro Definition Documentation . . . 144
8.3.1.1 COL . . . 144
8.3.1.2 ROW . . . 145
8.3.2 Function Documentation . . . 145
8.3.2.1 checkdim_() . . . 145
8.4 include/barry/barray-meat.hpp File Reference . . . 145
8.4.1 Macro Definition Documentation . . . 146
8.4.1.1 COL . . . 146
8.4.1.2 ROW . . . 146
8.5 include/barry/barraycell-bones.hpp File Reference . . . 147
8.6 include/barry/barraycell-meat.hpp File Reference . . . 147
8.7 include/barry/barraycol-bones.hpp File Reference . . . 148
8.8 include/barry/barraycol-meat.hpp File Reference . . . 149
8.8.1 Macro Definition Documentation . . . 149
8.8.1.1 BARRY_BARRAYCOL_MEAT_HPP . . . 150
8.9 include/barry/barry-configuration.hpp File Reference . . . 150
8.9.1 Macro Definition Documentation . . . 150
8.9.1.1 BARRY_CHECK_SUPPORT . . . 151
8.9.1.2 BARRY_ISFINITE . . . 151
8.9.1.3 BARRY_MAX_NUM_ELEMENTS . . . 151
8.9.1.4 BARRY_SAFE_EXP . . . 151
8.9.2 Typedef Documentation . . . 151
8.9.2.1 Map . . . 151
8.10 include/barry/barry.hpp File Reference . . . 152
8.10.1 Macro Definition Documentation . . . 153
8.10.1.1 COUNTER_FUNCTION . . . 153
8.10.1.2 COUNTER_LAMBDA . . . 154
8.10.1.3 RULE_FUNCTION . . . 154
8.10.1.4 RULE_LAMBDA . . . 154
8.11 include/barry/cell-bones.hpp File Reference . . . 154
8.12 include/barry/cell-meat.hpp File Reference . . . 155
8.13 include/barry/col-bones.hpp File Reference . . . 156

8.14 include/barry/counters-bones.hpp File Reference . . . 156
8.15 include/barry/counters-meat.hpp File Reference . . . 157
8.16 include/barry/counters/network.hpp File Reference . . . 158
8.16.1 Macro Definition Documentation . . . 161
8.16.1.1 NET_C_DATA_IDX . . . 161
8.16.1.2 NET_C_DATA_NUM . . . 161
8.16.1.3 NETWORK_COUNTER . . . 161
8.16.1.4 NETWORK_COUNTER_LAMBDA . . . 161
8.16.1.5 NETWORK_RULE . . . 162
8.16.1.6 NETWORK_RULE_LAMBDA . . . 162
8.16.2 Typedef Documentation . . . 162
8.16.2.1 NetCounter . . . 162
8.16.2.2 NetCounters . . . 162
8.16.2.3 NetModel . . . 163
8.16.2.4 NetRule . . . 163
8.16.2.5 NetRules . . . 163
8.16.2.6 NetStatsCounter . . . 163
8.16.2.7 NetSupport . . . 163
8.16.2.8 Network . . . 163
8.16.3 Function Documentation . . . 164
8.16.3.1 rules_zerodiag() . . . 164
8.17 include/barry/counters/phylo.hpp File Reference . . . 164
8.17.1 Macro Definition Documentation . . . 166
8.17.1.1 PHYLO_C_DATA_IDX . . . 166
8.17.1.2 PHYLO_CHECK_MISSING . . . 167
8.17.1.3 PHYLO_COUNTER . . . 167
8.17.1.4 PHYLO_COUNTER_LAMBDA . . . 167
8.17.2 Typedef Documentation . . . 167
8.17.2.1 PhyloArray . . . 167
8.17.2.2 PhyloCounter . . . 168
8.17.2.3 PhyloCounterData . . . 168
8.17.2.4 PhyloCounters . . . 168
8.17.2.5 PhyloModel . . . 168
8.17.2.6 PhyloPowerSet . . . 168
8.17.2.7 PhyloRule . . . 168
8.17.2.8 PhyloRuleData . . . 169
8.17.2.9 PhyloRules . . . 169
8.17.2.10 PhyloStatsCounter . . . 169
8.17.2.11 PhyloSupport . . . 169
8.17.3 Function Documentation . . . 169
8.17.3.1 get_last_name() . . . 169
8.18 include/barry/model-bones.hpp File Reference . . . 170

8.18.1 Function Documentation . . . 171
8.18.1.1 keygen_default() . . . 171
8.18.1.2 likelihood_() . . . 171
8.18.1.3 update_normalizing_constant() . . . 171
8.19 include/barry/model-meat.hpp File Reference . . . 172
8.20 include/barry/models/geese.hpp File Reference . . . 173
8.21 include/barry/models/geese/flock-bones.hpp File Reference . . . 173
8.22 include/barry/models/geese/flock-meet.hpp File Reference . . . 174
8.23 include/barry/models/geese/geese-bones.hpp File Reference . . . 174
8.23.1 Macro Definition Documentation . . . 175
8.23.1.1 INITIALIZED . . . 175
8.23.2 Function Documentation . . . 175
8.23.2.1 keygen_full() . . . 175
8.23.2.2 RULE_FUNCTION() . . . 176
8.23.2.3 vec_diff() . . . 176
8.23.2.4 vector_caster() . . . 176
8.24 include/barry/models/geese/geese-meat-constructors.hpp File Reference . . . 176
8.24.1 Macro Definition Documentation . . . 177
8.24.1.1 GEESE_MEAT_CONSTRUCTORS_HPP . . . 177
8.25 include/barry/models/geese/geese-meat-likelihood.hpp File Reference . . . 177
8.26 include/barry/models/geese/geese-meat-likelihood_exhaust.hpp File Reference . . . 178
8.27 include/barry/models/geese/geese-meat-predict.hpp File Reference . . . 179
8.28 include/barry/models/geese/geese-meat-simulate.hpp File Reference . . . 180
8.29 include/barry/models/geese/geese-meat.hpp File Reference . . . 181
8.30 include/barry/models/geese/geese-node-bones.hpp File Reference . . . 182
8.31 include/barry/powerset-bones.hpp File Reference . . . 182
8.32 include/barry/powerset-meat.hpp File Reference . . . 184
8.33 include/barry/rules-bones.hpp File Reference . . . 184
8.33.1 Function Documentation . . . 185
8.33.1.1 rule_fun_default() . . . 186
8.34 include/barry/rules-meat.hpp File Reference . . . 186
8.35 include/barry/statscounter-bones.hpp File Reference . . . 187
8.36 include/barry/statscounter-meat.hpp File Reference . . . 188
8.37 include/barry/statsdb.hpp File Reference . . . 188
8.38 include/barry/support-bones.hpp File Reference . . . 189
8.39 include/barry/support-meat.hpp File Reference . . . 191
8.39.1 Macro Definition Documentation . . . 192
8.39.1.1 BARRY_SUPPORT_MEAT_HPP . . . 192
8.40 include/barry/typedefs.hpp File Reference . . . 192
8.40.1 Typedef Documentation . . . 194
8.40.1.1 Col_type . . . 194
8.40.1.2 Counter_fun_type . . . 194

8.40.1.3 Counts_type . . . 194
8.40.1.4 MapVec_type . . . 194
8.40.1.5 Row_type . . . 195
8.40.1.6 Rule_fun_type . . . 195
8.40.1.7 uint . . . 195
8.40.2 Function Documentation . . . 195
8.40.2.1 vec_equal() . . . 195
8.40.2.2 vec_equal_approx() . . . 196
8.40.2.3 vec_inner_prod() . . . 196
8.41 README.md File Reference . . . 196

Index **197**

Chapter 1

Main Page

Barry: your to-go motif accountant

`This repository` contains a C++ template library that essentially counts sufficient statistics on binary arrays. The idea of the library is that this can be used together to build exponential family models as those in Exponential Random Graph Models (ERGMs), but as a generalization that also deals with non square arrays.

Examples

Counting statistics in a graph

In the following code we create an array of size 5x5 of class `Network` (available in the namespace `netcounters`), add/remove ties, print the graph, and count common statistics used in ERGMs:

```
#include <iostream>
#include <ostream>
#include "../include/barry.hpp"
typedef std::vector< unsigned int > vuint;
int main() {
  // Creating network of size six with five ties
  netcounters::Network net(
      6, 6,
      {0, 0, 4, 4, 2, 0, 1},
      {1, 2, 0, 2, 4, 0, 1}
  );

  // How does this looks like?
  std::cout « "Current view" « std::endl;
  net.print();

  // Adding extra ties
  net += {1, 0};
  net(2, 0) = true;

  // And removing a couple
  net(0, 0) = false;
  net -= {1, 1};
  std::cout « "New view" « std::endl;
  net.print();

  // Initializing the data. The program deals with freing the memory
  net.set_data(new netcounters::NetworkData, true);
  // Creating counter object for the network and adding stats to count
  netcounters::NetStatsCounter counter(&net);
  netcounters::counter_edges(counter.counters);
```

```
  netcounters::counter_ttriads(counter.counters);
  netcounters::counter_isolates(counter.counters);
  netcounters::counter_ctriads(counter.counters);
  netcounters::counter_mutual(counter.counters);

  // Counting and printing the results
  std::vector< double > counts = counter.count_all();

  std::cout «
    "Edges             : " « counts[0] « std::endl «
    "Transitive triads : " « counts[1] « std::endl «
    "Isolates          : " « counts[2] « std::endl «
    "C triads          : " « counts[3] « std::endl «
    "Mutuals           : " « counts[4] « std::endl;

  return 0;
}
```

Compiling this program using g++

```
g++ -std=c++11 -Wall -pedantic 08-counts.cpp -o counts && ./counts
```

Yields the following output:

```
Current view
[  0,]  1  1  1  .  .  .
[  1,]  .  1  .  .  .  .
[  2,]  .  .  .  .  1  .
[  3,]  .  .  .  .  .  .
[  4,]  1  .  1  .  .  .
[  5,]  .  .  .  .  .  .
New view
[  0,]  .  1  1  .  .  .
[  1,]  1  .  .  .  .  .
[  2,]  1  .  .  .  1  .
[  3,]  .  .  .  .  .  .
[  4,]  1  .  1  .  .  .
[  5,]  .  .  .  .  .  .
Edges             : 7
Transitive triads : 3
Isolates          : 2
C triads          : 1
Mutuals           : 3
```

Code of Conduct

Please note that the `barry` project is released with a Contributor Code of Conduct. By contributing to this project, you agree to abide by its terms.

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Counting 9
Statistical Models 9
Network counters 10
Phylo counters 15

Chapter 3

Class Index

3.1 Class List

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

BArray< Cell_Type, Data_Type >
: Baseline class for binary arrays . . . 25

BArrayCell< Cell_Type, Data_Type > . . . 39

BArrayCell_const< Cell_Type, Data_Type > . . . 42

BArrayCol< Cell_Type, Data_Type > . . . 44

BArrayCol_const< Cell_Type, Data_Type > . . . 47

Cell< Cell_Type >
: Entries in BArray. For now, it only has two members: . . . 49

ConstBArrayRowIter< Cell_Type, Data_Type > . . . 54

Counter< Array_Type, Data_Type >
: A counter function based on change statistics . . . 56

Counters< Array_Type, Data_Type >
: Vector of counters . . . 60

Entries< Cell_Type >
: A wrapper class to store `source`, `target`, `val` from a `BArray` object . . . 63

Flock
: A Flock is a group of Geese . . . 65

FreqTable< T >
: Database of statistics . . . 71

Geese
: Annotated Phylo Model . . . 73

Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >
: General framework for discrete exponential models. This class allows generating discrete exponential models in the form of a linear exponential model: . . . 83

NetCounterData
: Data class used to store arbitrary uint or double vectors . . . 100

NetworkData
: Data class for Networks . . . 101

Node
: A single node for the model . . . 103

NodeData
: Data definition for the `PhyloArray` class . . . 109

PowerSet< Array_Type, Data_Rule_Type >
: Powerset of a binary array . . . 111

Rule< Array_Type, Data_Type >
: Rule for determining if a cell should be included in a sequence . . . 117

Rules< Array_Type, Data_Type >
Vector of objects of class Rule 119
StatsCounter< Array_Type, Data_Type >
Count stats for a single Array 123
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >
Compute the support of sufficient statistics 128
vecHasher< T > 139

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

include/barry/barray-bones.hpp . . . 141
include/barry/barray-iterator.hpp . . . 142
include/barry/barray-meat-operators.hpp . . . 143
include/barry/barray-meat.hpp . . . 145
include/barry/barraycell-bones.hpp . . . 147
include/barry/barraycell-meat.hpp . . . 147
include/barry/barraycol-bones.hpp . . . 148
include/barry/barraycol-meat.hpp . . . 149
include/barry/barry-configuration.hpp . . . 150
include/barry/barry.hpp . . . 152
include/barry/cell-bones.hpp . . . 154
include/barry/cell-meat.hpp . . . 155
include/barry/col-bones.hpp . . . 156
include/barry/counters-bones.hpp . . . 156
include/barry/counters-meat.hpp . . . 157
include/barry/model-bones.hpp . . . 170
include/barry/model-meat.hpp . . . 172
include/barry/powerset-bones.hpp . . . 182
include/barry/powerset-meat.hpp . . . 184
include/barry/rules-bones.hpp . . . 184
include/barry/rules-meat.hpp . . . 186
include/barry/statscounter-bones.hpp . . . 187
include/barry/statscounter-meat.hpp . . . 188
include/barry/statsdb.hpp . . . 188
include/barry/support-bones.hpp . . . 189
include/barry/support-meat.hpp . . . 191
include/barry/typedefs.hpp . . . 192
include/barry/counters/network.hpp . . . 158
include/barry/counters/phylo.hpp . . . 164
include/barry/models/geese.hpp . . . 173
include/barry/models/geese/flock-bones.hpp . . . 173
include/barry/models/geese/flock-meet.hpp . . . 174
include/barry/models/geese/geese-bones.hpp . . . 174
include/barry/models/geese/geese-meat-constructors.hpp . . . 176
include/barry/models/geese/geese-meat-likelihood.hpp . . . 177

include/barry/models/geese/geese-meat-likelihood_exhaust.hpp 178
include/barry/models/geese/geese-meat-predict.hpp 179
include/barry/models/geese/geese-meat-simulate.hpp 180
include/barry/models/geese/geese-meat.hpp 181
include/barry/models/geese/geese-node-bones.hpp 182

Chapter 5

Module Documentation

5.1 Counting

Classes

- class NetworkData

  Data class for Networks.
- class NodeData

  Data definition for the `PhyloArray` *class.*
- class Counter< Array_Type, Data_Type >

  A counter function based on change statistics.

5.1.1 Detailed Description

`barry` includes a flexible way to generate counters based on change statistics. Since most of the time we are counting many motifs in a graph, change statistics make a reasonable (and efficient) way to make such counts.

In particular, let the motif be defined as $s(y)$, with y as the binary array. The change statistic when adding cell y_{ij}, i.e. when the cell moves from being emty to have a one, is defined as

$$\delta(y_{ij}) = s^{+}_{ij}(y) - s^{-}_{ij}(y),$$

where $s^{+}_{ij}(y)$ and $s^{-}_{ij}(y)$ represent the motif statistic with and without the ij-cell. For example, in the case of networks, the change statistic for the number of edges is always 1.

To count statistics in an array, the [Counter] class will empty the array, initialize the counters, and then start counting while adding at each step a single cell, until matching the original array.

5.2 Statistical Models

Statistical models available in `barry`.

Classes

- class Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >

  General framework for discrete exponential models. This class allows generating discrete exponential models in the form of a linear exponential model:
- class Flock

  A Flock is a group of Geese.
- class Geese

  Annotated Phylo Model.

5.2.1 Detailed Description

Statistical models available in `barry`.

5.3 Network counters

Counters for network models.

Functions

- void counter_edges (NetCounters *counters)

  Number of edges.
- void counter_isolates (NetCounters *counters)

  Number of isolated vertices.
- void counter_mutual (NetCounters *counters)

  Number of mutual ties.
- void counter_istar2 (NetCounters *counters)
- void counter_ostar2 (NetCounters *counters)
- void counter_ttriads (NetCounters *counters)
- void counter_ctriads (NetCounters *counters)
- void counter_density (NetCounters *counters)
- void counter_idegree15 (NetCounters *counters)
- void counter_odegree15 (NetCounters *counters)
- void counter_absdiff (NetCounters *counters, uint attr_id, double alpha=1.0)

  Sum of absolute attribute difference between ego and alter.
- void counter_diff (NetCounters *counters, uint attr_id, double alpha=1.0, double tail_head=true)

  Sum of attribute difference between ego and alter to pow(alpha)
- NETWORK_COUNTER (init_single_attr)
- void counter_nodeicov (NetCounters *counters, uint attr_id)
- void counter_nodeocov (NetCounters *counters, uint attr_id)
- void counter_nodecov (NetCounters *counters, uint attr_id)
- void counter_nodematch (NetCounters *counters, uint attr_id)
- void counter_idegree (NetCounters *counters, std::vector< uint > d)

  Counts number of vertices with a given in-degree.
- void counter_odegree (NetCounters *counters, std::vector< uint > d)

  Counts number of vertices with a given out-degree.
- void counter_degree (NetCounters *counters, std::vector< uint > d)

  Counts number of vertices with a given out-degree.

5.3.1 Detailed Description

Counters for network models.

Parameters

counters	A pointer to a `NetCounters` object (`Counters<Network, NetCounterData>`).

5.3.2 Function Documentation

5.3.2.1 counter_absdiff()

```
void counter_absdiff (
            NetCounters * counters,
            uint attr_id,
            double alpha = 1.0 )  [inline]
```

Sum of absolute attribute difference between ego and alter.

Definition at line 419 of file network.hpp.

5.3.2.2 counter_ctriads()

```
void counter_ctriads (
            NetCounters * counters )  [inline]
```

Definition at line 322 of file network.hpp.

5.3.2.3 counter_degree()

```
void counter_degree (
            NetCounters * counters,
            std::vector< uint > d )  [inline]
```

Counts number of vertices with a given out-degree.

Definition at line 690 of file network.hpp.

5.3.2.4 counter_density()

```
void counter_density (
            NetCounters * counters )  [inline]
```

Definition at line 361 of file network.hpp.

5.3.2.5 counter_diff()

```
void counter_diff (
            NetCounters * counters,
            uint attr_id,
            double alpha = 1.0,
            double tail_head = true )  [inline]
```

Sum of attribute difference between ego and alter to pow(alpha)

Definition at line 461 of file network.hpp.

5.3.2.6 counter_edges()

```
void counter_edges (
            NetCounters * counters )  [inline]
```

Number of edges.

Definition at line 128 of file network.hpp.

5.3.2.7 counter_idegree()

```
void counter_idegree (
            NetCounters * counters,
            std::vector< uint > d )  [inline]
```

Counts number of vertices with a given in-degree.

Definition at line 604 of file network.hpp.

5.3.2.8 counter_idegree15()

```
void counter_idegree15 (
            NetCounters * counters )  [inline]
```

Definition at line 377 of file network.hpp.

5.3.2.9 counter_isolates()

```
void counter_isolates (
            NetCounters * counters )  [inline]
```

Number of isolated vertices.

Definition at line 142 of file network.hpp.

5.3.2.10 counter_istar2()

```
void counter_istar2 (
            NetCounters * counters )  [inline]
```

Definition at line 210 of file network.hpp.

5.3.2.11 counter_mutual()

```
void counter_mutual (
            NetCounters * counters )  [inline]
```

Number of mutual ties.

Definition at line 172 of file network.hpp.

5.3.2.12 counter_nodecov()

```
void counter_nodecov (
            NetCounters * counters,
            uint attr_id )  [inline]
```

Definition at line 558 of file network.hpp.

5.3.2.13 counter_nodeicov()

```
void counter_nodeicov (
            NetCounters * counters,
            uint attr_id )  [inline]
```

Definition at line 520 of file network.hpp.

5.3.2.14 counter_nodematch()

```
void counter_nodematch (
            NetCounters * counters,
            uint attr_id )  [inline]
```

Definition at line 578 of file network.hpp.

5.3.2.15 counter_nodeocov()

```
void counter_nodeocov (
            NetCounters * counters,
            uint attr_id )  [inline]
```

Definition at line 539 of file network.hpp.

5.3.2.16 counter_odegree()

```
void counter_odegree (
            NetCounters * counters,
            std::vector< uint > d )  [inline]
```

Counts number of vertices with a given out-degree.

Definition at line 646 of file network.hpp.

5.3.2.17 counter_odegree15()

```
void counter_odegree15 (
            NetCounters * counters )  [inline]
```

Definition at line 397 of file network.hpp.

5.3.2.18 counter_ostar2()

```
void counter_ostar2 (
            NetCounters * counters )  [inline]
```

Definition at line 228 of file network.hpp.

5.3.2.19 counter_ttriads()

```
void counter_ttriads (
            NetCounters * counters )  [inline]
```

Definition at line 247 of file network.hpp.

5.3.2.20 NETWORK_COUNTER()

```
NETWORK_COUNTER (
              init_single_attr  )
```

Definition at line 503 of file network.hpp.

5.4 Phylo counters

Counters for phylogenetic modeling.

Functions

- void counter_overall_gains (PhyloCounters *counters, bool duplication=true)

  Overall functional gains.
- void counter_gains (PhyloCounters *counters, std::vector< uint > nfun, bool duplication=true)

  Functional gains for a specific function (`nfun`).
- void counter_gains_k_offspring (PhyloCounters *counters, std::vector< uint > nfun, uint k=1u, bool duplication=true)

  k genes gain function nfun
- void counter_genes_changing (PhyloCounters *counters, bool duplication=true)

  Keeps track of how many genes are changing (either 0, 1, or 2 if dealing with regular trees.)
- void counter_overall_loss (PhyloCounters *counters, bool duplication=true)

  Overall functional loss.
- void counter_maxfuns (PhyloCounters *counters, uint lb, uint ub, bool duplication=true)

  Cap the number of functions per gene.
- void counter_loss (PhyloCounters *counters, std::vector< uint > nfun, bool duplication=true)

  Total count of losses for an specific function.
- void counter_overall_changes (PhyloCounters *counters, bool duplication=true)

  Total number of changes. Use this statistic to account for "preservation".
- void counter_subfun (PhyloCounters *counters, uint nfunA, uint nfunB, bool duplication=true)

  Total count of Sub-functionalization events.
- void counter_cogain (PhyloCounters *counters, uint nfunA, uint nfunB, bool duplication=true)

  Co-evolution (joint gain or loss)
- void counter_longest (PhyloCounters *counters)

  Longest branch mutates (either by gain or by loss)
- void counter_neofun (PhyloCounters *counters, uint nfunA, uint nfunB, bool duplication=true)

  Total number of neofunctionalization events.
- void counter_neofun_a2b (PhyloCounters *counters, uint nfunA, uint nfunB, bool duplication=true)

  Total number of neofunctionalization events.
- void counter_co_opt (PhyloCounters *counters, uint nfunA, uint nfunB, bool duplication=true)

  Function co-opting.

5.4.1 Detailed Description

Counters for phylogenetic modeling.

Parameters

counters	A pointer to a `PhyloCounters` object (`Counters<PhyloArray, PhyloCounterData>`).

5.4.2 Function Documentation

5.4.2.1 counter_co_opt()

```
void counter_co_opt (
            PhyloCounters * counters,
            uint nfunA,
            uint nfunB,
            bool duplication = true )  [inline]
```

Function co-opting.

Function co-opting of functions A and B happens when, for example, function B is gained as a new featured leveraging what function A already does; without losing function A. The sufficient statistic is defined as follows:

$$x_{pa}(1-x_{pb})\sum_{i<j}\left[x_{ia}^{p}(1-x_{ib}^{p})x_{ja}^{p}x_{jb}^{p}+x_{ja}^{p}(1-x_{jb}^{p})x_{ia}^{p}x_{ib}^{p}\right]$$

This algorithm implements the change statistic.

Definition at line 1081 of file phylo.hpp.

5.4.2.2 counter_cogain()

```
void counter_cogain (
            PhyloCounters * counters,
            uint nfunA,
            uint nfunB,
            bool duplication = true )  [inline]
```

Co-evolution (joint gain or loss)

Needs to specify pairs of functions (`nfunA`, `nfunB`).

Definition at line 711 of file phylo.hpp.

5.4.2.3 counter_gains()

```
void counter_gains (
            PhyloCounters * counters,
            std::vector< uint > nfun,
            bool duplication = true )  [inline]
```

Functional gains for a specific function (`nfun`).

Definition at line 149 of file phylo.hpp.

5.4.2.4 counter_gains_k_offspring()

```
void counter_gains_k_offspring (
            PhyloCounters * counters,
            std::vector< uint > nfun,
            uint k = 1u,
            bool duplication = true )  [inline]
```

k genes gain function nfun

Definition at line 191 of file phylo.hpp.

5.4.2.5 counter_genes_changing()

```
void counter_genes_changing (
            PhyloCounters * counters,
            bool duplication = true )  [inline]
```

Keeps track of how many genes are changing (either 0, 1, or 2 if dealing with regular trees.)

Definition at line 265 of file phylo.hpp.

5.4.2.6 counter_longest()

```
void counter_longest (
            PhyloCounters * counters )  [inline]
```

Longest branch mutates (either by gain or by loss)

Definition at line 770 of file phylo.hpp.

5.4.2.7 counter_loss()

```
void counter_loss (
            PhyloCounters * counters,
            std::vector< uint > nfun,
            bool duplication = true )  [inline]
```

Total count of losses for an specific function.

Definition at line 512 of file phylo.hpp.

5.4.2.8 counter_maxfuns()

```
void counter_maxfuns (
            PhyloCounters * counters,
            uint lb,
            uint ub,
            bool duplication = true )  [inline]
```

Cap the number of functions per gene.

Definition at line 428 of file phylo.hpp.

5.4.2.9 counter_neofun()

```
void counter_neofun (
            PhyloCounters * counters,
            uint nfunA,
            uint nfunB,
            bool duplication = true )  [inline]
```

Total number of neofunctionalization events.

Needs to specify pairs of function.

Definition at line 875 of file phylo.hpp.

5.4.2.10 counter_neofun_a2b()

```
void counter_neofun_a2b (
            PhyloCounters * counters,
            uint nfunA,
            uint nfunB,
            bool duplication = true )  [inline]
```

Total number of neofunctionalization events.

Needs to specify pairs of function.

Definition at line 960 of file phylo.hpp.

5.4.2.11 counter_overall_changes()

```
void counter_overall_changes (
            PhyloCounters * counters,
            bool duplication = true )  [inline]
```

Total number of changes. Use this statistic to account for "preservation".

Definition at line 559 of file phylo.hpp.

5.4.2.12 counter_overall_gains()

```
void counter_overall_gains (
            PhyloCounters * counters,
            bool duplication = true )  [inline]
```

Overall functional gains.

Total number of gains (irrespective of the function).

Definition at line 109 of file phylo.hpp.

5.4.2.13 counter_overall_loss()

```
void counter_overall_loss (
            PhyloCounters * counters,
            bool duplication = true )  [inline]
```

Overall functional loss.

Definition at line 382 of file phylo.hpp.

5.4.2.14 counter_subfun()

```
void counter_subfun (
            PhyloCounters * counters,
            uint nfunA,
            uint nfunB,
            bool duplication = true )  [inline]
```

Total count of Sub-functionalization events.

It requires to specify data = {funA, funB}

Definition at line 625 of file phylo.hpp.

Chapter 6

Namespace Documentation

6.1 barry Namespace Reference

barry: Your go-to motif accountant

Namespaces

- counters

  Tree class and TreeIterator class.

6.1.1 Detailed Description

barry: Your go-to motif accountant

6.2 barry::counters Namespace Reference

Tree class and TreeIterator class.

Namespaces

- network
- phylo

6.2.1 Detailed Description

Tree class and TreeIterator class.

6.3 barry::counters::network Namespace Reference

6.4 barry::counters::phylo Namespace Reference

6.5 CHECK Namespace Reference

Integer constants used to specify which cell should be check.

Variables

- const int BOTH = -1
- const int NONE = 0
- const int ONE = 1
- const int TWO = 2

6.5.1 Detailed Description

Integer constants used to specify which cell should be check.

6.5.2 Variable Documentation

6.5.2.1 BOTH

```
const int CHECK::BOTH = -1
```

Definition at line 30 of file typedefs.hpp.

6.5.2.2 NONE

```
const int CHECK::NONE = 0
```

Definition at line 31 of file typedefs.hpp.

6.5.2.3 ONE

```
const int CHECK::ONE = 1
```

Definition at line 32 of file typedefs.hpp.

6.5.2.4 TWO

```
const int CHECK::TWO = 2
```

Definition at line 33 of file typedefs.hpp.

6.6 EXISTS Namespace Reference

Integer constants used to specify which cell should be check to exist or not.

Variables

- const int BOTH = -1
- const int NONE = 0
- const int ONE = 1
- const int TWO = 1
- const int UKNOWN = -1
- const int AS_ZERO = 0
- const int AS_ONE = 1

6.6.1 Detailed Description

Integer constants used to specify which cell should be check to exist or not.

6.6.2 Variable Documentation

6.6.2.1 AS_ONE

```
const int EXISTS::AS_ONE = 1
```

Definition at line 48 of file typedefs.hpp.

6.6.2.2 AS_ZERO

```
const int EXISTS::AS_ZERO = 0
```

Definition at line 47 of file typedefs.hpp.

6.6.2.3 BOTH

```
const int EXISTS::BOTH = -1
```

Definition at line 41 of file typedefs.hpp.

6.6.2.4 NONE

```
const int EXISTS::NONE = 0
```

Definition at line 42 of file typedefs.hpp.

6.6.2.5 ONE

```
const int EXISTS::ONE = 1
```

Definition at line 43 of file typedefs.hpp.

6.6.2.6 TWO

```
const int EXISTS::TWO = 1
```

Definition at line 44 of file typedefs.hpp.

6.6.2.7 UKNOWN

```
const int EXISTS::UKNOWN = -1
```

Definition at line 46 of file typedefs.hpp.

Chapter 7

Class Documentation

7.1 BArray< Cell_Type, Data_Type > Class Template Reference

Baseline class for binary arrays.

`#include <barray-bones.hpp>`

Public Member Functions

- bool operator== (const BArray< Cell_Type, Data_Type > &Array_)
- ~BArray ()
- void out_of_range (uint i, uint j) const
- Cell_Type get_cell (uint i, uint j, bool check_bounds=true) const
- const Row_type< Cell_Type > * get_row (uint i, bool check_bounds=true) const
- const Col_type< Cell_Type > * get_col (uint i, bool check_bounds=true) const
- std::vector< Cell_Type > get_col_vec (uint i, bool check_bounds=true) const
- std::vector< Cell_Type > get_row_vec (uint i, bool check_bounds=true) const
- void get_col_vec (std::vector< Cell_Type > *x, uint i, bool check_bounds=true) const
- void get_row_vec (std::vector< Cell_Type > *x, uint i, bool check_bounds=true) const
- const Row_type< Cell_Type > & row (uint i, bool check_bounds=true) const
- const Col_type< Cell_Type > & col (uint i, bool check_bounds=true) const
- Entries< Cell_Type > get_entries () const

  Get the edgelist.
- void transpose ()
- void clear (bool hard=true)
- void resize (uint N_, uint M_)
- void reserve ()
- void print () const

Constructors

Parameters

N_	*Number of rows*
M_	*Number of columns*
source	*An unsigned vector ranging from 0 to N_*
target	*An unsigned int vector ranging from 0 to M_*
target	*When `true` tries to add repeated observations.*

- BArray ()
  Zero-size array.
- BArray (uint N_, uint M_)
  Empty array.
- BArray (uint N_, uint M_, const std::vector< uint > &source, const std::vector< uint > &target, const std::vector< Cell_Type > &value, bool add=true)
  Edgelist with data.
- BArray (uint N_, uint M_, const std::vector< uint > &source, const std::vector< uint > &target, bool add=true)
  Edgelist with no data (simpler)
- BArray (const BArray< Cell_Type, Data_Type > &Array_, bool copy_data=false)
  Copy constructor.
- BArray< Cell_Type, Data_Type > & operator= (const BArray< Cell_Type, Data_Type > &Array_)
  Assignment constructor.
- BArray (BArray< Cell_Type, Data_Type > &&x) noexcept
  Move operator.
- BArray< Cell_Type, Data_Type > & operator= (BArray< Cell_Type, Data_Type > &&x) noexcept
  Move assignment.

- void set_data (Data_Type *data_, bool delete_data_=false)
  Set the data object.
- Data_Type * D ()
- const Data_Type * D () const

Queries

`is_empty` queries a single cell. `nrow`, `ncol`, and `nnozero` return the number of rows, columns, and non-zero cells respectively.

Parameters

i,j	*Coordinates*
check_bounds	*If `false` avoids checking bounds.*

- bool is_empty (uint i, uint j, bool check_bounds=true) const
- uint nrow () const noexcept
- uint ncol () const noexcept
- uint nnozero () const noexcept
- Cell< Cell_Type > default_val () const

Cell-wise insertion/deletion

Parameters

i,j	*Row,column*
check_bounds	*When `true` and out of range, the function throws an error.*
check_exists	*Wither check if the cell exists (before trying to delete/add), or, in the case of `swap_cells`, check if either of both cells exists/don't exist.*

- BArray< Cell_Type, Data_Type > & operator+= (const std::pair< uint, uint > &coords)
- BArray< Cell_Type, Data_Type > & operator-= (const std::pair< uint, uint > &coords)

- BArrayCell< Cell_Type, Data_Type > operator() (uint i, uint j, bool check_bounds=true)
- const BArrayCell_const< Cell_Type, Data_Type > operator() (uint i, uint j, bool check_bounds=true) const
- void rm_cell (uint i, uint j, bool check_bounds=true, bool check_exists=true)
- void insert_cell (uint i, uint j, const Cell< Cell_Type > &v, bool check_bounds, bool check_exists)
- void insert_cell (uint i, uint j, Cell< Cell_Type > &&v, bool check_bounds, bool check_exists)
- void insert_cell (uint i, uint j, Cell_Type v, bool check_bounds, bool check_exists)
- void swap_cells (uint i0, uint j0, uint i1, uint j1, bool check_bounds=true, int check_exists=CHECK::BOTH, int *report=nullptr)
- void toggle_cell (uint i, uint j, bool check_bounds=true, int check_exists=EXISTS::UKNOWN)
- void toggle_lock (uint i, uint j, bool check_bounds=true)

Column/row wise interchange

- void swap_rows (uint i0, uint i1, bool check_bounds=true)
- void swap_cols (uint j0, uint j1, bool check_bounds=true)
- void zero_row (uint i, bool check_bounds=true)
- void zero_col (uint j, bool check_bounds=true)

Arithmetic operators

- BArray< Cell_Type, Data_Type > & operator+= (const BArray< Cell_Type, Data_Type > &rhs)
- BArray< Cell_Type, Data_Type > & operator+= (const Cell_Type &rhs)
- BArray< Cell_Type, Data_Type > & operator-= (const BArray< Cell_Type, Data_Type > &rhs)
- BArray< Cell_Type, Data_Type > & operator-= (const Cell_Type &rhs)
- BArray< Cell_Type, Data_Type > & operator/= (const Cell_Type &rhs)
- BArray< Cell_Type, Data_Type > & operator*= (const Cell_Type &rhs)

Public Attributes

- bool visited = false

Friends

- class BArrayCell< Cell_Type, Data_Type >
- class BArrayCell_const< Cell_Type, Data_Type >

7.1.1 Detailed Description

template<typename Cell_Type = bool, typename Data_Type = bool>
class BArray< Cell_Type, Data_Type >

Baseline class for binary arrays.

`BArray` class objects are arbitrary arrays in which non-empty cells hold data of type `Cell_Type`. The non-empty cells are stored by row and indexed using `unordered_map`s, i.e. `std::vector< std::unordered_map<unsigned int,Cell_Type> >`.

Template Parameters

Cell_Type	Type of cell (any type).
Data_Type	Data type of the array (bool default).

Definition at line 22 of file barray-bones.hpp.

7.1.2 Constructor & Destructor Documentation

7.1.2.1 BArray() [1/6]

```
template<typename Cell_Type = bool, typename Data_Type = bool>
BArray< Cell_Type, Data_Type >::BArray ( )  [inline]
```

Zero-size array.

Definition at line 60 of file barray-bones.hpp.

7.1.2.2 BArray() [2/6]

```
template<typename Cell_Type = bool, typename Data_Type = bool>
BArray< Cell_Type, Data_Type >::BArray (
            uint N_,
            uint M_ )  [inline]
```

Empty array.

Definition at line 63 of file barray-bones.hpp.

7.1.2.3 BArray() [3/6]

```
template<typename Cell_Type = bool, typename Data_Type = bool>
BArray< Cell_Type, Data_Type >::BArray (
            uint N_,
            uint M_,
            const std::vector< uint > & source,
            const std::vector< uint > & target,
            const std::vector< Cell_Type > & value,
            bool add = true )
```

Edgelist with data.

7.1.2.4 BArray() [4/6]

```
template<typename Cell_Type = bool, typename Data_Type = bool>
BArray< Cell_Type, Data_Type >::BArray (
            uint N_,
            uint M_,
            const std::vector< uint > & source,
            const std::vector< uint > & target,
            bool add = true )
```

Edgelist with no data (simpler)

7.1.2.5 BArray() [5/6]

```
template<typename Cell_Type = bool, typename Data_Type = bool>
BArray< Cell_Type, Data_Type >::BArray (
            const BArray< Cell_Type, Data_Type > & Array_,
            bool copy_data = false )
```

Copy constructor.

7.1.2.6 BArray() [6/6]

```
template<typename Cell_Type = bool, typename Data_Type = bool>
BArray< Cell_Type, Data_Type >::BArray (
            BArray< Cell_Type, Data_Type > && x )  [noexcept]
```

Move operator.

7.1.2.7 ~BArray()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
BArray< Cell_Type, Data_Type >::~BArray ( )
```

7.1.3 Member Function Documentation

7.1.3.1 clear()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
void BArray< Cell_Type, Data_Type >::clear (
            bool hard = true )
```

7.1.3.2 col()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
const Col_type< Cell_Type >& BArray< Cell_Type, Data_Type >::col (
            uint i,
            bool check_bounds = true ) const
```

7.1.3.3 D() [1/2]

```
template<typename Cell_Type = bool, typename Data_Type = bool>
Data_Type* BArray< Cell_Type, Data_Type >::D ( )
```

7.1.3.4 D() [2/2]

```
template<typename Cell_Type = bool, typename Data_Type = bool>
const Data_Type* BArray< Cell_Type, Data_Type >::D ( ) const
```

7.1.3.5 default_val()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
Cell<Cell_Type> BArray< Cell_Type, Data_Type >::default_val ( ) const
```

7.1.3.6 get_cell()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
Cell_Type BArray< Cell_Type, Data_Type >::get_cell (
            uint i,
            uint j,
            bool check_bounds = true ) const
```

7.1.3.7 get_col()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
const Col_type< Cell_Type >* BArray< Cell_Type, Data_Type >::get_col (
            uint i,
            bool check_bounds = true ) const
```

7.1.3.8 get_col_vec() [1/2]

```
template<typename Cell_Type = bool, typename Data_Type = bool>
void BArray< Cell_Type, Data_Type >::get_col_vec (
            std::vector< Cell_Type > * x,
            uint i,
            bool check_bounds = true ) const
```

7.1.3.9 get_col_vec() [2/2]

```
template<typename Cell_Type = bool, typename Data_Type = bool>
std::vector< Cell_Type > BArray< Cell_Type, Data_Type >::get_col_vec (
            uint i,
            bool check_bounds = true ) const
```

7.1.3.10 get_entries()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
Entries<Cell_Type> BArray< Cell_Type, Data_Type >::get_entries ( ) const
```

Get the edgelist.

`Entries` is a class with three objects: Two `std::vector` with the row and column coordinates respectively, and one `std::vector` with the corresponding value of the cell.

Returns

Entries<Cell_Type>

7.1.3.11 get_row()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
const Row_type< Cell_Type >* BArray< Cell_Type, Data_Type >::get_row (
            uint i,
            bool check_bounds = true ) const
```

7.1.3.12 get_row_vec() [1/2]

```
template<typename Cell_Type = bool, typename Data_Type = bool>
void BArray< Cell_Type, Data_Type >::get_row_vec (
            std::vector< Cell_Type > * x,
            uint i,
            bool check_bounds = true ) const
```

7.1.3.13 get_row_vec() [2/2]

```
template<typename Cell_Type = bool, typename Data_Type = bool>
std::vector< Cell_Type > BArray< Cell_Type, Data_Type >::get_row_vec (
            uint i,
            bool check_bounds = true ) const
```

7.1.3.14 insert_cell() [1/3]

```
template<typename Cell_Type = bool, typename Data_Type = bool>
void BArray< Cell_Type, Data_Type >::insert_cell (
            uint i,
            uint j,
            Cell< Cell_Type > && v,
            bool check_bounds,
            bool check_exists )
```

7.1.3.15 insert_cell() [2/3]

```
template<typename Cell_Type = bool, typename Data_Type = bool>
void BArray< Cell_Type, Data_Type >::insert_cell (
            uint i,
            uint j,
            Cell_Type v,
            bool check_bounds,
            bool check_exists )
```

7.1.3.16 insert_cell() [3/3]

```
template<typename Cell_Type = bool, typename Data_Type = bool>
void BArray< Cell_Type, Data_Type >::insert_cell (
            uint i,
            uint j,
            const Cell< Cell_Type > & v,
            bool check_bounds,
            bool check_exists )
```

7.1.3.17 is_empty()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
bool BArray< Cell_Type, Data_Type >::is_empty (
            uint i,
            uint j,
            bool check_bounds = true ) const
```

7.1.3.18 ncol()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
uint BArray< Cell_Type, Data_Type >::ncol ( ) const  [noexcept]
```

7.1.3.19 nnozero()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
uint BArray< Cell_Type, Data_Type >::nnozero ( ) const  [noexcept]
```

7.1.3.20 nrow()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
uint BArray< Cell_Type, Data_Type >::nrow ( ) const  [noexcept]
```

7.1.3.21 operator()() [1/2]

```
template<typename Cell_Type = bool, typename Data_Type = bool>
BArrayCell<Cell_Type,Data_Type> BArray< Cell_Type, Data_Type >::operator() (
            uint i,
            uint j,
            bool check_bounds = true )
```

7.1.3.22 operator()() [2/2]

```
template<typename Cell_Type = bool, typename Data_Type = bool>
const BArrayCell_const<Cell_Type,Data_Type> BArray< Cell_Type, Data_Type >::operator() (
            uint i,
            uint j,
            bool check_bounds = true ) const
```

7.1.3.23 operator*=()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
BArray<Cell_Type,Data_Type>& BArray< Cell_Type, Data_Type >::operator*= (
            const Cell_Type & rhs )
```

7.1.3.24 operator+=() [1/3]

```
template<typename Cell_Type = bool, typename Data_Type = bool>
BArray<Cell_Type,Data_Type>& BArray< Cell_Type, Data_Type >::operator+= (
            const BArray< Cell_Type, Data_Type > & rhs )
```

7.1.3.25 operator+=() [2/3]

```
template<typename Cell_Type = bool, typename Data_Type = bool>
BArray<Cell_Type,Data_Type>& BArray< Cell_Type, Data_Type >::operator+= (
            const Cell_Type & rhs )
```

7.1.3.26 operator+=() [3/3]

```
template<typename Cell_Type = bool, typename Data_Type = bool>
BArray<Cell_Type,Data_Type>& BArray< Cell_Type, Data_Type >::operator+= (
            const std::pair< uint, uint > & coords )
```

7.1.3.27 operator-=() [1/3]

```
template<typename Cell_Type = bool, typename Data_Type = bool>
BArray<Cell_Type,Data_Type>& BArray< Cell_Type, Data_Type >::operator-= (
            const BArray< Cell_Type, Data_Type > & rhs )
```

7.1.3.28 operator-=() [2/3]

```
template<typename Cell_Type = bool, typename Data_Type = bool>
BArray<Cell_Type,Data_Type>& BArray< Cell_Type, Data_Type >::operator-= (
            const Cell_Type & rhs )
```

7.1.3.29 operator-=() [3/3]

```
template<typename Cell_Type = bool, typename Data_Type = bool>
BArray<Cell_Type,Data_Type>& BArray< Cell_Type, Data_Type >::operator-= (
            const std::pair< uint, uint > & coords )
```

7.1.3.30 operator/=()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
BArray<Cell_Type,Data_Type>& BArray< Cell_Type, Data_Type >::operator/= (
            const Cell_Type & rhs )
```

7.1.3.31 operator=() [1/2]

```
template<typename Cell_Type = bool, typename Data_Type = bool>
BArray<Cell_Type,Data_Type>& BArray< Cell_Type, Data_Type >::operator= (
            BArray< Cell_Type, Data_Type > && x )  [noexcept]
```

Move assignment.

7.1.3.32 operator=() [2/2]

```
template<typename Cell_Type = bool, typename Data_Type = bool>
BArray<Cell_Type,Data_Type>& BArray< Cell_Type, Data_Type >::operator= (
            const BArray< Cell_Type, Data_Type > & Array_ )
```

Assignment constructor.

7.1.3.33 operator==()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
bool BArray< Cell_Type, Data_Type >::operator== (
            const BArray< Cell_Type, Data_Type > & Array_ )
```

7.1.3.34 out_of_range()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
void BArray< Cell_Type, Data_Type >::out_of_range (
            uint i,
            uint j ) const
```

7.1.3.35 print()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
void BArray< Cell_Type, Data_Type >::print ( ) const
```

7.1.3.36 reserve()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
void BArray< Cell_Type, Data_Type >::reserve ( )
```

7.1.3.37 resize()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
void BArray< Cell_Type, Data_Type >::resize (
            uint N_,
            uint M_ )
```

7.1.3.38 rm_cell()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
void BArray< Cell_Type, Data_Type >::rm_cell (
            uint i,
            uint j,
            bool check_bounds = true,
            bool check_exists = true )
```

7.1.3.39 row()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
const Row_type< Cell_Type >& BArray< Cell_Type, Data_Type >::row (
            uint i,
            bool check_bounds = true ) const
```

7.1.3.40 set_data()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
void BArray< Cell_Type, Data_Type >::set_data (
            Data_Type * data_,
            bool delete_data_ = false )
```

Set the data object.

Parameters

*data_*	
*delete_↩ data_*	

7.1.3.41 swap_cells()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
void BArray< Cell_Type, Data_Type >::swap_cells (
            uint i0,
            uint j0,
            uint i1,
            uint j1,
            bool check_bounds = true,
            int check_exists = CHECK::BOTH,
            int * report = nullptr )
```

7.1.3.42 swap_cols()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
void BArray< Cell_Type, Data_Type >::swap_cols (
            uint j0,
            uint j1,
            bool check_bounds = true )
```

7.1.3.43 swap_rows()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
void BArray< Cell_Type, Data_Type >::swap_rows (
            uint i0,
            uint i1,
            bool check_bounds = true )
```

7.1.3.44 toggle_cell()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
void BArray< Cell_Type, Data_Type >::toggle_cell (
            uint i,
            uint j,
            bool check_bounds = true,
            int check_exists = EXISTS::UKNOWN )
```

7.1.3.45 toggle_lock()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
void BArray< Cell_Type, Data_Type >::toggle_lock (
            uint i,
            uint j,
            bool check_bounds = true )
```

7.1.3.46 transpose()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
void BArray< Cell_Type, Data_Type >::transpose ( )
```

7.1.3.47 zero_col()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
void BArray< Cell_Type, Data_Type >::zero_col (
            uint j,
            bool check_bounds = true )
```

7.1.3.48 zero_row()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
void BArray< Cell_Type, Data_Type >::zero_row (
            uint i,
            bool check_bounds = true )
```

7.1.4 Friends And Related Function Documentation

7.1.4.1 BArrayCell< Cell_Type, Data_Type >

```
template<typename Cell_Type = bool, typename Data_Type = bool>
friend class BArrayCell< Cell_Type, Data_Type >  [friend]
```

Definition at line 1 of file barray-bones.hpp.

7.1.4.2 BArrayCell_const< Cell_Type, Data_Type >

```
template<typename Cell_Type = bool, typename Data_Type = bool>
friend class BArrayCell_const< Cell_Type, Data_Type >  [friend]
```

Definition at line 1 of file barray-bones.hpp.

7.1.5 Member Data Documentation

7.1.5.1 visited

```
template<typename Cell_Type = bool, typename Data_Type = bool>
bool BArray< Cell_Type, Data_Type >::visited = false
```

This is as a reference, if we need to iterate through the cells and we need to keep track which were visited, we use this as a reference. So that if cell.visited = true and visited = true, it means that we haven't been here yet. Ideally, any routine using this->visited should switch it at the beginning of the routine.

Definition at line 45 of file barray-bones.hpp.

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

- include/barry/barray-bones.hpp

7.2 BArrayCell< Cell_Type, Data_Type > Class Template Reference

```
#include <barraycell-bones.hpp>
```

Public Member Functions

- BArrayCell (BArray< Cell_Type, Data_Type > *Array_, uint i_, uint j_, bool check_bounds=true)
- ~BArrayCell ()
- void operator= (const Cell_Type &val)
- void operator+= (const Cell_Type &val)
- void operator-= (const Cell_Type &val)
- void operator*= (const Cell_Type &val)
- void operator/= (const Cell_Type &val)
- operator Cell_Type () const
- bool operator== (const Cell_Type &val) const

7.2.1 Detailed Description

template<typename Cell_Type = bool, typename Data_Type = bool>
class BArrayCell< Cell_Type, Data_Type >

Definition at line 7 of file barraycell-bones.hpp.

7.2.2 Constructor & Destructor Documentation

7.2.2.1 BArrayCell()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
BArrayCell< Cell_Type, Data_Type >::BArrayCell (
            BArray< Cell_Type, Data_Type > * Array_,
            uint i_,
            uint j_,
            bool check_bounds = true )  [inline]
```

Definition at line 16 of file barraycell-bones.hpp.

7.2.2.2 ~BArrayCell()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
BArrayCell< Cell_Type, Data_Type >::~BArrayCell ( )  [inline]
```

Definition at line 28 of file barraycell-bones.hpp.

7.2.3 Member Function Documentation

7.2.3.1 operator Cell_Type()

```
template<typename Cell_Type , typename Data_Type >
BArrayCell< Cell_Type, Data_Type >::operator Cell_Type  [inline]
```

Definition at line 58 of file barraycell-meat.hpp.

7.2.3.2 operator*=()

```
template<typename Cell_Type , typename Data_Type >
void BArrayCell< Cell_Type, Data_Type >::operator*= (
            const Cell_Type & val )  [inline]
```

Definition at line 40 of file barraycell-meat.hpp.

7.2.3.3 operator+=()

```
template<typename Cell_Type , typename Data_Type >
void BArrayCell< Cell_Type, Data_Type >::operator+= (
            const Cell_Type & val )  [inline]
```

Definition at line 18 of file barraycell-meat.hpp.

7.2.3.4 operator-=()

```
template<typename Cell_Type , typename Data_Type >
void BArrayCell< Cell_Type, Data_Type >::operator-= (
            const Cell_Type & val )  [inline]
```

Definition at line 29 of file barraycell-meat.hpp.

7.2.3.5 operator/=()

```
template<typename Cell_Type , typename Data_Type >
void BArrayCell< Cell_Type, Data_Type >::operator/= (
            const Cell_Type & val )  [inline]
```

Definition at line 49 of file barraycell-meat.hpp.

7.2.3.6 operator=()

```
template<typename Cell_Type , typename Data_Type >
void BArrayCell< Cell_Type, Data_Type >::operator= (
            const Cell_Type & val )  [inline]
```

Definition at line 7 of file barraycell-meat.hpp.

7.2.3.7 operator==()

```
template<typename Cell_Type , typename Data_Type >
bool BArrayCell< Cell_Type, Data_Type >::operator== (
            const Cell_Type & val ) const  [inline]
```

Definition at line 63 of file barraycell-meat.hpp.

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

- include/barry/barraycell-bones.hpp
- include/barry/barraycell-meat.hpp

7.3 BArrayCell_const< Cell_Type, Data_Type > Class Template Reference

`#include <barraycell-bones.hpp>`

Public Member Functions

- BArrayCell_const (const BArray< Cell_Type, Data_Type > *Array_, uint i_, uint j_, bool check_bounds=true)
- ~BArrayCell_const ()
- operator Cell_Type () const
- bool operator== (const Cell_Type &val) const
- bool operator!= (const Cell_Type &val) const
- bool operator< (const Cell_Type &val) const
- bool operator> (const Cell_Type &val) const
- bool operator<= (const Cell_Type &val) const
- bool operator>= (const Cell_Type &val) const

7.3.1 Detailed Description

template<typename Cell_Type = bool, typename Data_Type = bool>
class BArrayCell_const< Cell_Type, Data_Type >

Definition at line 41 of file barraycell-bones.hpp.

7.3.2 Constructor & Destructor Documentation

7.3.2.1 BArrayCell_const()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
BArrayCell_const< Cell_Type, Data_Type >::BArrayCell_const (
            const BArray< Cell_Type, Data_Type > * Array_,
            uint i_,
            uint j_,
            bool check_bounds = true )  [inline]
```

Definition at line 50 of file barraycell-bones.hpp.

7.3.2.2 ~BArrayCell_const()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
BArrayCell_const< Cell_Type, Data_Type >::~BArrayCell_const ( )  [inline]
```

Definition at line 62 of file barraycell-bones.hpp.

7.3.3 Member Function Documentation

7.3.3.1 operator Cell_Type()

```
template<typename Cell_Type , typename Data_Type >
BArrayCell_const< Cell_Type, Data_Type >::operator Cell_Type  [inline]
```

Definition at line 68 of file barraycell-meat.hpp.

7.3.3.2 operator"!=()

```
template<typename Cell_Type , typename Data_Type >
bool BArrayCell_const< Cell_Type, Data_Type >::operator!= (
            const Cell_Type & val ) const  [inline]
```

Definition at line 78 of file barraycell-meat.hpp.

7.3.3.3 operator<()

```
template<typename Cell_Type , typename Data_Type >
bool BArrayCell_const< Cell_Type, Data_Type >::operator< (
            const Cell_Type & val ) const  [inline]
```

Definition at line 83 of file barraycell-meat.hpp.

7.3.3.4 operator<=()

```
template<typename Cell_Type , typename Data_Type >
bool BArrayCell_const< Cell_Type, Data_Type >::operator<= (
            const Cell_Type & val ) const  [inline]
```

Definition at line 93 of file barraycell-meat.hpp.

7.3.3.5 operator==()

```
template<typename Cell_Type , typename Data_Type >
bool BArrayCell_const< Cell_Type, Data_Type >::operator== (
            const Cell_Type & val ) const  [inline]
```

Definition at line 73 of file barraycell-meat.hpp.

7.3.3.6 operator>()

```
template<typename Cell_Type , typename Data_Type >
bool BArrayCell_const< Cell_Type, Data_Type >::operator> (
            const Cell_Type & val ) const  [inline]
```

Definition at line 88 of file barraycell-meat.hpp.

7.3.3.7 operator>=()

```
template<typename Cell_Type , typename Data_Type >
bool BArrayCell_const< Cell_Type, Data_Type >::operator>= (
            const Cell_Type & val ) const  [inline]
```

Definition at line 98 of file barraycell-meat.hpp.

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

- include/barry/barraycell-bones.hpp
- include/barry/barraycell-meat.hpp

7.4 BArrayCol< Cell_Type, Data_Type > Class Template Reference

```
#include <barraycol-bones.hpp>
```

Public Member Functions

- BArrayCol (BArray< Cell_Type, Data_Type > *Array_, uint i_, bool check_bounds=true)
- ~BArrayCol ()
- void operator= (const Cell_Type &val)
- void operator+= (const Cell_Type &val)
- void operator-= (const Cell_Type &val)
- void operator*= (const Cell_Type &val)
- void operator/= (const Cell_Type &val)
- operator Cell_Type () const
- bool operator== (const Cell_Type &val) const
- Col_type< Cell_Type >::iterator begin () noexcept
- Col_type< Cell_Type >::iterator end () noexcept

7.4.1 Detailed Description

template<typename Cell_Type = bool, typename Data_Type = bool>
class BArrayCol< Cell_Type, Data_Type >

Definition at line 7 of file barraycol-bones.hpp.

7.4.2 Constructor & Destructor Documentation

7.4.2.1 BArrayCol()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
BArrayCol< Cell_Type, Data_Type >::BArrayCol (
            BArray< Cell_Type, Data_Type > * Array_,
            uint i_,
            bool check_bounds = true )  [inline]
```

Definition at line 16 of file barraycol-bones.hpp.

7.4.2.2 ~BArrayCol()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
BArrayCol< Cell_Type, Data_Type >::~BArrayCol ( )  [inline]
```

Definition at line 26 of file barraycol-bones.hpp.

7.4.3 Member Function Documentation

7.4.3.1 begin()

```
template<typename Cell_Type , typename Data_Type >
Col_type< Cell_Type >::iterator BArrayCol< Cell_Type, Data_Type >::begin  [inline], [noexcept]
```

Definition at line 68 of file barraycol-meat.hpp.

7.4.3.2 end()

```
template<typename Cell_Type , typename Data_Type >
Col_type< Cell_Type >::iterator BArrayCol< Cell_Type, Data_Type >::end  [inline], [noexcept]
```

Definition at line 73 of file barraycol-meat.hpp.

7.4.3.3 operator Cell_Type()

```
template<typename Cell_Type , typename Data_Type >
BArrayCol< Cell_Type, Data_Type >::operator Cell_Type  [inline]
```

Definition at line 58 of file barraycol-meat.hpp.

7.4.3.4 operator*=()

```
template<typename Cell_Type , typename Data_Type >
void BArrayCol< Cell_Type, Data_Type >::operator*= (
             const Cell_Type & val )  [inline]
```

Definition at line 40 of file barraycol-meat.hpp.

7.4.3.5 operator+=()

```
template<typename Cell_Type , typename Data_Type >
void BArrayCol< Cell_Type, Data_Type >::operator+= (
             const Cell_Type & val )  [inline]
```

Definition at line 18 of file barraycol-meat.hpp.

7.4.3.6 operator-=()

```
template<typename Cell_Type , typename Data_Type >
void BArrayCol< Cell_Type, Data_Type >::operator-= (
             const Cell_Type & val )  [inline]
```

Definition at line 29 of file barraycol-meat.hpp.

7.4.3.7 operator/=()

```
template<typename Cell_Type , typename Data_Type >
void BArrayCol< Cell_Type, Data_Type >::operator/= (
             const Cell_Type & val )  [inline]
```

Definition at line 49 of file barraycol-meat.hpp.

7.4.3.8 operator=()

```
template<typename Cell_Type , typename Data_Type >
void BArrayCol< Cell_Type, Data_Type >::operator= (
             const Cell_Type & val )  [inline]
```

Definition at line 7 of file barraycol-meat.hpp.

7.4.3.9 operator==()

```
template<typename Cell_Type , typename Data_Type >
bool BArrayCol< Cell_Type, Data_Type >::operator== (
             const Cell_Type & val ) const  [inline]
```

Definition at line 63 of file barraycol-meat.hpp.

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

- include/barry/barraycol-bones.hpp
- include/barry/barraycol-meat.hpp

7.5 BArrayCol_const< Cell_Type, Data_Type > Class Template Reference

```
#include <barraycol-bones.hpp>
```

Public Member Functions

- BArrayCol (const BArray< Cell_Type, Data_Type > *Array_, uint i_, bool check_bounds=true)
- ~BArrayCol_const ()
- bool operator== (const Cell_Type &val) const
- bool operator!= (const Cell_Type &val) const
- bool operator< (const Cell_Type &val) const
- bool operator> (const Cell_Type &val) const
- bool operator<= (const Cell_Type &val) const
- bool operator>= (const Cell_Type &val) const

7.5.1 Detailed Description

template<typename Cell_Type = bool, typename Data_Type = bool>
class BArrayCol_const< Cell_Type, Data_Type >

Definition at line 42 of file barraycol-bones.hpp.

7.5.2 Constructor & Destructor Documentation

7.5.2.1 ~BArrayCol_const()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
BArrayCol_const< Cell_Type, Data_Type >::~BArrayCol_const ( )  [inline]
```

Definition at line 60 of file barraycol-bones.hpp.

7.5.3 Member Function Documentation

7.5.3.1 BArrayCol()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
BArrayCol_const< Cell_Type, Data_Type >::BArrayCol (
            const BArray< Cell_Type, Data_Type > * Array_,
            uint i_,
            bool check_bounds = true )  [inline]
```

Definition at line 50 of file barraycol-bones.hpp.

7.5.3.2 operator"!=()

```
template<typename Cell_Type , typename Data_Type >
bool BArrayCol_const< Cell_Type, Data_Type >::operator!= (
            const Cell_Type & val ) const  [inline]
```

Definition at line 92 of file barraycol-meat.hpp.

7.5.3.3 operator<()

```
template<typename Cell_Type , typename Data_Type >
bool BArrayCol_const< Cell_Type, Data_Type >::operator< (
            const Cell_Type & val ) const  [inline]
```

Definition at line 97 of file barraycol-meat.hpp.

7.5.3.4 operator<=()

```
template<typename Cell_Type , typename Data_Type >
bool BArrayCol_const< Cell_Type, Data_Type >::operator<= (
            const Cell_Type & val ) const  [inline]
```

Definition at line 107 of file barraycol-meat.hpp.

7.5.3.5 operator==()

```
template<typename Cell_Type , typename Data_Type >
bool BArrayCol_const< Cell_Type, Data_Type >::operator== (
            const Cell_Type & val ) const  [inline]
```

Definition at line 87 of file barraycol-meat.hpp.

7.5.3.6 operator>()

```
template<typename Cell_Type , typename Data_Type >
bool BArrayCol_const< Cell_Type, Data_Type >::operator> (
            const Cell_Type & val ) const  [inline]
```

Definition at line 102 of file barraycol-meat.hpp.

7.5.3.7 operator>=()

```
template<typename Cell_Type , typename Data_Type >
bool BArrayCol_const< Cell_Type, Data_Type >::operator>= (
            const Cell_Type & val ) const  [inline]
```

Definition at line 112 of file barraycol-meat.hpp.

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

- include/barry/barraycol-bones.hpp
- include/barry/barraycol-meat.hpp

7.6 Cell< Cell_Type > Class Template Reference

Entries in BArray. For now, it only has two members:

```
#include <cell-bones.hpp>
```

Public Member Functions

- Cell ()
- Cell (Cell_Type value_, bool visited_=false)
- ~Cell ()
- Cell (const Cell< Cell_Type > &arg)
- Cell< Cell_Type > & operator= (Cell< Cell_Type > &other)
- Cell (Cell< Cell_Type > &&arg) noexcept
- Cell< Cell_Type > & operator= (Cell< Cell_Type > &&other) noexcept
- void add (Cell_Type x)
- operator Cell_Type () const
- void add (double x)
- void add (unsigned int x)
- void add (int x)
- Cell ()
- Cell ()
- Cell ()

Public Attributes

- Cell_Type value
- bool visited

7.6.1 Detailed Description

template<class Cell_Type>
class Cell< Cell_Type >

Entries in BArray. For now, it only has two members:

- value: the content
- visited: boolean (just a convenient)

Definition at line 13 of file cell-bones.hpp.

7.6.2 Constructor & Destructor Documentation

7.6.2.1 Cell() [1/7]

```
template<class Cell_Type >
Cell< Cell_Type >::Cell ( )
```

7.6.2.2 Cell() [2/7]

```
template<class Cell_Type >
Cell< Cell_Type >::Cell (
            Cell_Type value_,
            bool visited_ = false )  [inline]
```

Definition at line 18 of file cell-bones.hpp.

7.6.2.3 ~Cell()

```
template<class Cell_Type >
Cell< Cell_Type >::~Cell ( )  [inline]
```

Definition at line 20 of file cell-bones.hpp.

7.6.2.4 Cell() [3/7]

```
template<class Cell_Type >
Cell< Cell_Type >::Cell (
            const Cell< Cell_Type > & arg )  [inline]
```

Definition at line 24 of file cell-bones.hpp.

7.6.2.5 Cell() [4/7]

```
template<class Cell_Type >
Cell< Cell_Type >::Cell (
            Cell< Cell_Type > && arg )  [inline], [noexcept]
```

Definition at line 30 of file cell-bones.hpp.

7.6.2.6 Cell() [5/7]

```
Cell< double >::Cell ( )  [inline]
```

Definition at line 44 of file cell-meat.hpp.

7.6.2.7 Cell() [6/7]

```
Cell< uint >::Cell ( )  [inline]
```

Definition at line 45 of file cell-meat.hpp.

7.6.2.8 Cell() [7/7]

```
Cell< int >::Cell ( )  [inline]
```

Definition at line 46 of file cell-meat.hpp.

7.6.3 Member Function Documentation

7.6.3.1 add() [1/4]

```
template<class Cell_Type >
void Cell< Cell_Type >::add (
            Cell_Type x )
```

7.6.3.2 add() [2/4]

```
void Cell< double >::add (
            double x )  [inline]
```

Definition at line 24 of file cell-meat.hpp.

7.6.3.3 add() [3/4]

```
void Cell< int >::add (
            int x )  [inline]
```

Definition at line 34 of file cell-meat.hpp.

7.6.3.4 add() [4/4]

```
void Cell< unsigned int >::add (
            unsigned int x )  [inline]
```

Definition at line 29 of file cell-meat.hpp.

7.6.3.5 operator Cell_Type()

```
template<class Cell_Type >
Cell< Cell_Type >::operator Cell_Type ( ) const  [inline]
```

Definition at line 41 of file cell-bones.hpp.

7.6.3.6 operator=() [1/2]

```
template<typename Cell_Type >
Cell< Cell_Type > & Cell< Cell_Type >::operator= (
            Cell< Cell_Type > && other )  [noexcept]
```

Definition at line 14 of file cell-meat.hpp.

7.6.3.7 operator=() [2/2]

```
template<typename Cell_Type >
Cell< Cell_Type > & Cell< Cell_Type >::operator= (
            Cell< Cell_Type > & other )
```

Definition at line 7 of file cell-meat.hpp.

7.6.4 Member Data Documentation

7.6.4.1 value

```
template<class Cell_Type >
Cell_Type Cell< Cell_Type >::value
```

Definition at line 15 of file cell-bones.hpp.

7.6.4.2 visited

```
template<class Cell_Type >
bool Cell< Cell_Type >::visited
```

Definition at line 16 of file cell-bones.hpp.

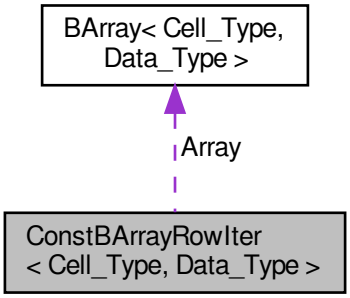
The documentation for this class was generated from the following files:

- include/barry/cell-bones.hpp
- include/barry/cell-meat.hpp

7.7 ConstBArrayRowIter< Cell_Type, Data_Type > Class Template Reference

```
#include <barray-iterator.hpp>
```

Collaboration diagram for ConstBArrayRowIter< Cell_Type, Data_Type >:



Public Member Functions

- ConstBArrayRowIter (const BArray< Cell_Type, Data_Type > *Array_)
- ~ConstBArrayRowIter ()

Public Attributes

- uint current_row
- uint current_col
- Row_type< Cell_Type >::const_iterator iter
- const BArray< Cell_Type, Data_Type > * Array

7.7.1 Detailed Description

template<typename Cell_Type, typename Data_Type>
class ConstBArrayRowIter< Cell_Type, Data_Type >

Definition at line 10 of file barray-iterator.hpp.

7.7.2 Constructor & Destructor Documentation

7.7.2.1 ConstBArrayRowIter()

```
template<typename Cell_Type , typename Data_Type >
ConstBArrayRowIter< Cell_Type, Data_Type >::ConstBArrayRowIter (
            const BArray< Cell_Type, Data_Type > * Array_ )  [inline]
```

Definition at line 17 of file barray-iterator.hpp.

7.7.2.2 ~ConstBArrayRowIter()

```
template<typename Cell_Type , typename Data_Type >
ConstBArrayRowIter< Cell_Type, Data_Type >::~ConstBArrayRowIter ( )  [inline]
```

Definition at line 29 of file barray-iterator.hpp.

7.7.3 Member Data Documentation

7.7.3.1 Array

```
template<typename Cell_Type , typename Data_Type >
const BArray<Cell_Type,Data_Type>* ConstBArrayRowIter< Cell_Type, Data_Type >::Array
```

Definition at line 15 of file barray-iterator.hpp.

7.7.3.2 current_col

```
template<typename Cell_Type , typename Data_Type >
uint ConstBArrayRowIter< Cell_Type, Data_Type >::current_col
```

Definition at line 13 of file barray-iterator.hpp.

7.7.3.3 current_row

```
template<typename Cell_Type , typename Data_Type >
uint ConstBArrayRowIter< Cell_Type, Data_Type >::current_row
```

Definition at line 13 of file barray-iterator.hpp.

7.7.3.4 iter

```
template<typename Cell_Type , typename Data_Type >
Row_type<Cell_Type>::const_iterator ConstBArrayRowIter< Cell_Type, Data_Type >::iter
```

Definition at line 14 of file barray-iterator.hpp.

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

- include/barry/barray-iterator.hpp

7.8 Counter< Array_Type, Data_Type > Class Template Reference

A counter function based on change statistics.

```
#include <counters-bones.hpp>
```

Public Member Functions

- Counter< Array_Type, Data_Type > operator= (const Counter< Array_Type, Data_Type > &counter_)
- ~Counter ()
- double count (Array_Type &Array, uint i, uint j)
- double init (Array_Type &Array, uint i, uint j)

Creator passing a counter and an initializer

Parameters

count_fun_	*The main counter function.*
init_fun_	*The initializer function can also be used to check if the BArray as the needed variables (see BArray::data).*
data_	*Data to be used with the counter.*
delete_data_	*When true, the destructor will delete the pointer in the main data.*

- Counter ()
- Counter (Counter_fun_type< Array_Type, Data_Type > count_fun_, Counter_fun_type< Array_Type,

Data_Type > init_fun_=nullptr, Data_Type *data_=nullptr, bool delete_data_=false, std::string name_↩
=””, std::string desc_=””)
- Counter (const Counter< Array_Type, Data_Type > &counter_)

Public Attributes

- Counter_fun_type< Array_Type, Data_Type > count_fun
- Counter_fun_type< Array_Type, Data_Type > init_fun
- Data_Type * data = nullptr
- bool delete_data = false
- std::string name = ””
- std::string desc = ””

7.8.1 Detailed Description

template<typename Array_Type = BArray<>, typename Data_Type = bool>
class Counter< Array_Type, Data_Type >

A counter function based on change statistics.

This class is used by `CountStats` and `StatsCounter` as a way to count statistics using change statistics.

Definition at line 38 of file counters-bones.hpp.

7.8.2 Constructor & Destructor Documentation

7.8.2.1 Counter() [1/3]

```
template<typename Array_Type = BArray<>, typename Data_Type = bool>
Counter< Array_Type, Data_Type >::Counter ( )  [inline]
```

Definition at line 59 of file counters-bones.hpp.

7.8.2.2 Counter() [2/3]

```
template<typename Array_Type = BArray<>, typename Data_Type = bool>
Counter< Array_Type, Data_Type >::Counter (
            Counter_fun_type< Array_Type, Data_Type > count_fun_,
            Counter_fun_type< Array_Type, Data_Type > init_fun_ = nullptr,
            Data_Type * data_ = nullptr,
            bool delete_data_ = false,
            std::string name_ = "",
            std::string desc_ = "" )  [inline]
```

Definition at line 61 of file counters-bones.hpp.

7.8.2.3 Counter() [3/3]

```
template<typename Array_Type , typename Data_Type >
Counter< Array_Type, Data_Type >::Counter (
            const Counter< Array_Type, Data_Type > & counter_ )  [inline]
```

Definition at line 7 of file counters-meat.hpp.

7.8.2.4 ~Counter()

```
template<typename Array_Type = BArray<>, typename Data_Type = bool>
Counter< Array_Type, Data_Type >::~Counter ( )  [inline]
```

Definition at line 76 of file counters-bones.hpp.

7.8.3 Member Function Documentation

7.8.3.1 count()

```
template<typename Array_Type , typename Data_Type >
double Counter< Array_Type, Data_Type >::count (
            Array_Type & Array,
            uint i,
            uint j )  [inline]
```

Definition at line 124 of file counters-meat.hpp.

7.8.3.2 init()

```
template<typename Array_Type , typename Data_Type >
double Counter< Array_Type, Data_Type >::init (
            Array_Type & Array,
            uint i,
            uint j )  [inline]
```

Definition at line 136 of file counters-meat.hpp.

7.8.3.3 operator=()

```
template<typename Array_Type , typename Data_Type >
Counter< Array_Type, Data_Type > Counter< Array_Type, Data_Type >::operator= (
            const Counter< Array_Type, Data_Type > & counter_ )
```

Definition at line 34 of file counters-meat.hpp.

7.8.4 Member Data Documentation

7.8.4.1 count_fun

```
template<typename Array_Type = BArray<>, typename Data_Type = bool>
Counter_fun_type<Array_Type,Data_Type> Counter< Array_Type, Data_Type >::count_fun
```

Definition at line 41 of file counters-bones.hpp.

7.8.4.2 data

```
template<typename Array_Type = BArray<>, typename Data_Type = bool>
Data_Type* Counter< Array_Type, Data_Type >::data = nullptr
```

Definition at line 43 of file counters-bones.hpp.

7.8.4.3 delete_data

```
template<typename Array_Type = BArray<>, typename Data_Type = bool>
bool Counter< Array_Type, Data_Type >::delete_data = false
```

Definition at line 44 of file counters-bones.hpp.

7.8.4.4 desc

```
template<typename Array_Type = BArray<>, typename Data_Type = bool>
std::string Counter< Array_Type, Data_Type >::desc = ""
```

Definition at line 46 of file counters-bones.hpp.

7.8.4.5 init_fun

```
template<typename Array_Type = BArray<>, typename Data_Type = bool>
Counter_fun_type<Array_Type,Data_Type> Counter< Array_Type, Data_Type >::init_fun
```

Definition at line 42 of file counters-bones.hpp.

7.8.4.6 name

```
template<typename Array_Type = BArray<>, typename Data_Type = bool>
std::string Counter< Array_Type, Data_Type >::name = ""
```

Definition at line 45 of file counters-bones.hpp.

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

- include/barry/counters-bones.hpp
- include/barry/counters-meat.hpp

7.9 Counters< Array_Type, Data_Type > Class Template Reference

Vector of counters.

```
#include <counters-bones.hpp>
```

Public Member Functions

- Counters ()
- ~Counters ()
- Counters (const Counters< Array_Type, Data_Type > &counter_)
- Counters< Array_Type, Data_Type > operator= (const Counters< Array_Type, Data_Type > &counter_)
- Counter< Array_Type, Data_Type > & operator[ ] (uint idx)

  Returns a pointer to a particular counter.
- std::size_t size () const noexcept

  Number of counters in the set.
- void add_counter (Counter< Array_Type, Data_Type > &counter)
- void add_counter (Counter< Array_Type, Data_Type > *counter)
- void add_counter (Counter_fun_type< Array_Type, Data_Type > count_fun_, Counter_fun_type< Array_Type, Data_Type > init_fun_=nullptr, Data_Type *data_=nullptr, bool delete_data_=false, std::string name_="", std::string desc_="")
- void clear ()

7.9.1 Detailed Description

template<typename Array_Type = BArray<>, typename Data_Type = bool>
class Counters< Array_Type, Data_Type >

Vector of counters.

Various functions hold more than one counter, so this class is a helper class that allows managing multiple counters efficiently. The main data is a vector to pointers of counters.

Definition at line 98 of file counters-bones.hpp.

7.9.2 Constructor & Destructor Documentation

7.9.2.1 Counters() [1/2]

```
template<typename Array_Type = BArray<>, typename Data_Type = bool>
Counters< Array_Type, Data_Type >::Counters ( )  [inline]
```

Definition at line 107 of file counters-bones.hpp.

7.9.2.2 ~Counters()

```
template<typename Array_Type = BArray<>, typename Data_Type = bool>
Counters< Array_Type, Data_Type >::~Counters ( )  [inline]
```

Definition at line 110 of file counters-bones.hpp.

7.9.2.3 Counters() [2/2]

```
template<typename Array_Type , typename Data_Type >
Counters< Array_Type, Data_Type >::Counters (
            const Counters< Array_Type, Data_Type > & counter_ )  [inline]
```

Definition at line 67 of file counters-meat.hpp.

7.9.3 Member Function Documentation

7.9.3.1 add_counter() [1/3]

```
template<typename Array_Type , typename Data_Type >
void Counters< Array_Type, Data_Type >::add_counter (
            Counter< Array_Type, Data_Type > & counter )  [inline]
```

Definition at line 157 of file counters-meat.hpp.

7.9.3.2 add_counter() [2/3]

```
template<typename Array_Type , typename Data_Type >
void Counters< Array_Type, Data_Type >::add_counter (
            Counter< Array_Type, Data_Type > * counter )  [inline]
```

Definition at line 169 of file counters-meat.hpp.

7.9.3.3 add_counter() [3/3]

```
template<typename Array_Type , typename Data_Type >
void Counters< Array_Type, Data_Type >::add_counter (
            Counter_fun_type< Array_Type, Data_Type > count_fun_,
            Counter_fun_type< Array_Type, Data_Type > init_fun_ = nullptr,
            Data_Type * data_ = nullptr,
            bool delete_data_ = false,
            std::string name_ = "",
            std::string desc_ = "" )  [inline]
```

Definition at line 180 of file counters-meat.hpp.

7.9.3.4 clear()

```
template<typename Array_Type , typename Data_Type >
void Counters< Array_Type, Data_Type >::clear  [inline]
```

Definition at line 209 of file counters-meat.hpp.

7.9.3.5 operator=()

```
template<typename Array_Type , typename Data_Type >
Counters< Array_Type, Data_Type > Counters< Array_Type, Data_Type >::operator= (
            const Counters< Array_Type, Data_Type > & counter_ )
```

Definition at line 94 of file counters-meat.hpp.

7.9.3.6 operator[]()

```
template<typename Array_Type , typename Data_Type >
Counter< Array_Type, Data_Type > & Counters< Array_Type, Data_Type >::operator[] (
            uint idx )  [inline]
```

Returns a pointer to a particular counter.

Parameters

idx	Id of the counter

Returns

Counter<Array_Type,Data_Type>*

Definition at line 150 of file counters-meat.hpp.

7.9.3.7 size()

```
template<typename Array_Type = BArray<>, typename Data_Type = bool>
std::size_t Counters< Array_Type, Data_Type >::size ( ) const  [inline], [noexcept]
```

Number of counters in the set.

Returns

uint

Definition at line 130 of file counters-bones.hpp.

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

- include/barry/counters-bones.hpp
- include/barry/counters-meat.hpp

7.10 Entries< Cell_Type > Class Template Reference

A wrapper class to store `source`, `target`, `val` from a `BArray` object.

`#include <typedefs.hpp>`

Public Member Functions

- Entries ()
- Entries (uint n)
- ~Entries ()
- void resize (uint n)

Public Attributes

- std::vector< uint > source
- std::vector< uint > target
- std::vector< Cell_Type > val

7.10.1 Detailed Description

template<typename Cell_Type>
class Entries< Cell_Type >

A wrapper class to store `source`, `target`, `val` from a `BArray` object.

Template Parameters

Cell_Type	Any type

Definition at line 69 of file typedefs.hpp.

7.10.2 Constructor & Destructor Documentation

7.10.2.1 Entries() [1/2]

```
template<typename Cell_Type >
Entries< Cell_Type >::Entries ( )  [inline]
```

Definition at line 75 of file typedefs.hpp.

7.10.2.2 Entries() [2/2]

```
template<typename Cell_Type >
Entries< Cell_Type >::Entries (
            uint n )  [inline]
```

Definition at line 76 of file typedefs.hpp.

7.10.2.3 ~Entries()

```
template<typename Cell_Type >
Entries< Cell_Type >::~Entries ( )  [inline]
```

Definition at line 83 of file typedefs.hpp.

7.10.3 Member Function Documentation

7.10.3.1 resize()

```
template<typename Cell_Type >
void Entries< Cell_Type >::resize (
            uint n )  [inline]
```

Definition at line 85 of file typedefs.hpp.

7.10.4 Member Data Documentation

7.10.4.1 source

```
template<typename Cell_Type >
std::vector< uint > Entries< Cell_Type >::source
```

Definition at line 71 of file typedefs.hpp.

7.10.4.2 target

```
template<typename Cell_Type >
std::vector< uint > Entries< Cell_Type >::target
```

Definition at line 72 of file typedefs.hpp.

7.10.4.3 val

```
template<typename Cell_Type >
std::vector< Cell_Type > Entries< Cell_Type >::val
```

Definition at line 73 of file typedefs.hpp.

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

- include/barry/typedefs.hpp

7.11 Flock Class Reference

A Flock is a group of Geese.

```
#include <flock-bones.hpp>
```

Public Member Functions

- Flock ()
- ~Flock ()
- unsigned int add_data (std::vector< std::vector< unsigned int > > &annotations, std::vector< unsigned int > &geneid, std::vector< int > &parent, std::vector< bool > &duplication)

  Add a tree to the flock.
- void set_seed (const unsigned int &s)

  Set the seed of the model.
- void init ()
- phylocounters::PhyloCounters * counters_ptr ()
- double likelihood_joint (const std::vector< double > &par, bool as_log=false, bool use_reduced_sequence=true)

  Returns the joint likelihood of the model.
- Geese * operator() (unsigned int i, bool check_bounds=true)

  Access the i-th geese element.

Information about the model

- unsigned int nfuns () const noexcept
- unsigned int ntrees () const noexcept
- std::vector< unsigned int > nnodes () const noexcept
- std::vector< unsigned int > nleafs () const noexcept
- unsigned int nterms () const

Public Attributes

- std::vector< Geese > dat
- unsigned int nfunctions = 0u
- bool initialized = false
- std::mt19937 rengine
- phylocounters::PhyloModel support = phylocounters::PhyloModel()

7.11.1 Detailed Description

A Flock is a group of Geese.

This object buils a model with multiple trees (Geese objects), with all of these using the same PhyloModel object. Available counters (terms) can be found in counter-phylo.

Definition at line 14 of file flock-bones.hpp.

7.11.2 Constructor & Destructor Documentation

7.11.2.1 Flock()

```
Flock::Flock ( )  [inline]
```

Definition at line 25 of file flock-bones.hpp.

7.11.2.2 ~Flock()

```
Flock::~Flock ( )  [inline]
```

Definition at line 26 of file flock-bones.hpp.

7.11.3 Member Function Documentation

7.11.3.1 add_data()

```
unsigned int Flock::add_data (
            std::vector< std::vector< unsigned int > > & annotations,
            std::vector< unsigned int > & geneid,
            std::vector< int > & parent,
            std::vector< bool > & duplication )  [inline]
```

Add a tree to the flock.

Parameters

annotations	see Geese::Geese.
geneid	see Geese.
parent	see Geese.
duplication	see Geese.

Returns

unsigned int The number of tree in the model (starting from zero).

Definition at line 6 of file flock-meet.hpp.

7.11.3.2 counters_ptr()

```
phylocounters::PhyloCounters * Flock::counters_ptr ( )  [inline]
```

Definition at line 75 of file flock-meet.hpp.

7.11.3.3 init()

```
void Flock::init ( )  [inline]
```

Definition at line 41 of file flock-meet.hpp.

7.11.3.4 likelihood_joint()

```
double Flock::likelihood_joint (
            const std::vector< double > & par,
            bool as_log = false,
            bool use_reduced_sequence = true )  [inline]
```

Returns the joint likelihood of the model.

Parameters

par	Vector of model parameters.
as_log	When `true` it will return the value as log.
use_reduced_sequence	When `true` (default) will compute the likelihood using the reduced sequence, which is faster.

Returns

double

Definition at line 84 of file flock-meet.hpp.

7.11.3.5 nfuns()

```
unsigned int Flock::nfuns ( ) const  [inline], [noexcept]
```

Definition at line 109 of file flock-meet.hpp.

7.11.3.6 nleafs()

```
std::vector< unsigned int > Flock::nleafs ( ) const  [inline], [noexcept]
```

Definition at line 132 of file flock-meet.hpp.

7.11.3.7 nnodes()

```
std::vector< unsigned int > Flock::nnodes ( ) const  [inline], [noexcept]
```

Definition at line 121 of file flock-meet.hpp.

7.11.3.8 nterms()

```
unsigned int Flock::nterms ( ) const  [inline]
```

Definition at line 144 of file flock-meet.hpp.

7.11.3.9 ntrees()

```
unsigned int Flock::ntrees ( ) const  [inline], [noexcept]
```

Definition at line 115 of file flock-meet.hpp.

7.11.3.10 operator()()

```
Geese * Flock::operator() (
            unsigned int i,
            bool check_bounds = true )  [inline]
```

Access the i-th geese element.

Parameters

i	Element to access
check_bounds	When true, it will check bounds.

Returns

Geese*

Definition at line 151 of file flock-meet.hpp.

7.11.3.11 set_seed()

```
void Flock::set_seed (
            const unsigned int & s )  [inline]
```

Set the seed of the model.

Parameters

s	Passed to the `rengine.seed()` member object.

Definition at line 37 of file flock-meet.hpp.

7.11.4 Member Data Documentation

7.11.4.1 dat

```
std::vector< Geese > Flock::dat
```

Definition at line 17 of file flock-bones.hpp.

7.11.4.2 initialized

```
bool Flock::initialized = false
```

Definition at line 19 of file flock-bones.hpp.

7.11.4.3 nfunctions

```
unsigned int Flock::nfunctions = 0u
```

Definition at line 18 of file flock-bones.hpp.

7.11.4.4 rengine

```
std::mt19937 Flock::rengine
```

Definition at line 22 of file flock-bones.hpp.

7.11.4.5 support

```
phylocounters::PhyloModel Flock::support = phylocounters::PhyloModel()
```

Definition at line 23 of file flock-bones.hpp.

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

- include/barry/models/geese/flock-bones.hpp
- include/barry/models/geese/flock-meet.hpp

7.12 FreqTable< T > Class Template Reference

Database of statistics.

`#include <statsdb.hpp>`

Public Member Functions

- FreqTable ()
- ~FreqTable ()
- void add (const std::vector< T > &x)
- Counts_type as_vector () const
- MapVec_type< T, uint > get_data () const
- const MapVec_type< T, uint > * get_data_ptr () const
- void clear ()
- void reserve (unsigned int n)
- void print () const
- size_t size () const noexcept

7.12.1 Detailed Description

template<typename T = double>
class FreqTable< T >

Database of statistics.

This is mostly used in `Support`.

Definition at line 16 of file statsdb.hpp.

7.12.2 Constructor & Destructor Documentation

7.12.2.1 FreqTable()

```
template<typename T = double>
FreqTable< T >::FreqTable ( )  [inline]
```

Definition at line 28 of file statsdb.hpp.

7.12.2.2 ~FreqTable()

```
template<typename T = double>
FreqTable< T >::~FreqTable ( )  [inline]
```

Definition at line 29 of file statsdb.hpp.

7.12.3 Member Function Documentation

7.12.3.1 add()

```
template<typename T >
void FreqTable< T >::add (
            const std::vector< T > & x )  [inline]
```

Definition at line 47 of file statsdb.hpp.

7.12.3.2 as_vector()

```
template<typename T >
Counts_type FreqTable< T >::as_vector  [inline]
```

Definition at line 61 of file statsdb.hpp.

7.12.3.3 clear()

```
template<typename T >
void FreqTable< T >::clear  [inline]
```

Definition at line 83 of file statsdb.hpp.

7.12.3.4 get_data()

```
template<typename T >
MapVec_type< T, uint > FreqTable< T >::get_data  [inline]
```

Definition at line 73 of file statsdb.hpp.

7.12.3.5 get_data_ptr()

```
template<typename T >
const MapVec_type< T, uint > * FreqTable< T >::get_data_ptr  [inline]
```

Definition at line 78 of file statsdb.hpp.

7.12.3.6 print()

```
template<typename T >
void FreqTable< T >::print  [inline]
```

Definition at line 102 of file statsdb.hpp.

7.12.3.7 reserve()

```
template<typename T >
void FreqTable< T >::reserve (
            unsigned int n )  [inline]
```

Definition at line 89 of file statsdb.hpp.

7.12.3.8 size()

```
template<typename T >
size_t FreqTable< T >::size  [inline], [noexcept]
```

Definition at line 126 of file statsdb.hpp.

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

- include/barry/statsdb.hpp

7.13 Geese Class Reference

Annotated Phylo Model.

`#include <geese-bones.hpp>`

Public Member Functions

- ~Geese ()
- void init ()
- void inherit_support (const Geese &model_, bool delete_support_=false)
- void calc_sequence (Node *n=nullptr)
- void calc_reduced_sequence ()
- double likelihood (const std::vector< double > &par, bool as_log=false, bool use_reduced_sequence=true)
- double likelihood_exhaust (const std::vector< double > &par)
- std::vector< double > get_probabilities () const
- void set_seed (const unsigned int &s)
- std::vector< std::vector< unsigned int > > simulate (const std::vector< double > &par)
- std::vector< std::vector< double > > observed_counts ()
- void print_observed_counts ()
- void init_node (Node &n)
- void update_annotations (unsigned int nodeid, std::vector< unsigned int > newann)

Construct a new Geese object

The model includes a total of `N + 1` nodes, the `+ 1` beign the root node.

Parameters

annotations	*A vector of vectors with annotations. It should be of length* `k` *(number of functions). Each vector should be of length* `N` *(equal to the number of nodes, including interior). Possible values are 0, 1, and 9.*
geneid	*Id of the gene. It should be of length* `N`*.*
parent	*Id of the parent gene. Also of length* `N`

- Geese ()
- Geese (std::vector< std::vector< unsigned int > > &annotations, std::vector< unsigned int > &geneid, std::vector< int > &parent, std::vector< bool > &duplication)
- Geese (const Geese &model_, bool copy_data=true)
- Geese (Geese &&x) noexcept
- Geese & operator= (const Geese &model_)=delete
- Geese & operator= (Geese &&model_) noexcept=delete

Information about the model

- unsigned int nfuns () const noexcept
- unsigned int nnodes () const noexcept
- unsigned int nleafs () const noexcept
- unsigned int nterms () const

Geese prediction

Calculate the conditional probability

Parameters

par	*Vector of parameters (terms + root).*
res_prob	*Vector indicating each nodes' state probability.*
leave_one_out	*When* `true`*, it will compute the predictions using leave-one-out, thus the prediction will be repeated nleaf times.*
only_annotated	*When* `true`*, it will make the predictions only on the induced sub-tree with annotated leafs.*
use_reduced_sequence	*Passed to the* `likelihood` *method.*
preorder	*For the tree traversal.*

When `res_prob` *is specified, the function will attach the member vector* `probabilities` *from the* `Node`*s objects. This contains the probability that the ith node has either of the possible states.*

Returns

std::vector< double > Returns the posterior probability

- std::vector< std::vector< double > > predict (const std::vector< double > &par, std::vector< std::vector< double > > *res_prob=nullptr, bool leave_one_out=false, bool only_annotated=false, bool use_reduced↩ _sequence=true)
- std::vector< std::vector< double > > predict_backend (const std::vector< double > &par, bool use_↩ reduced_sequence, const std::vector< uint > &preorder)

Public Attributes

- unsigned int nfunctions
- std::map< unsigned int, Node > nodes

- barry::MapVec_type< unsigned int > map_to_nodes
- std::vector< unsigned int > sequence
- std::vector< unsigned int > reduced_sequence
- bool initialized = false
- bool delete_rengine = false
- bool delete_counters = false
- bool delete_support = false

Shared objects within a <tt>Geese</tt>

Since users may start adding counters before initializing the PhyloModel object, the object `counter` *is initialized first.*

While the member `support` *has an* `rengine`*, since* `Geese` *can sample trees, we have the option to keep it separate.*

- std::mt19937 * rengine = nullptr
- phylocounters::PhyloCounters * counters = nullptr
- phylocounters::PhyloModel * support = nullptr
- std::vector< std::vector< bool > > states

7.13.1 Detailed Description

Annotated Phylo Model.

A list of available terms for this model can be found in the Phylo counters section.

Definition at line 70 of file geese-bones.hpp.

7.13.2 Constructor & Destructor Documentation

7.13.2.1 Geese() [1/4]

```
Geese::Geese ( )  [inline]
```

Definition at line 6 of file geese-meat-constructors.hpp.

7.13.2.2 Geese() [2/4]

```
Geese::Geese (
            std::vector< std::vector< unsigned int > > & annotations,
            std::vector< unsigned int > & geneid,
            std::vector< int > & parent,
            std::vector< bool > & duplication )  [inline]
```

Definition at line 17 of file geese-meat-constructors.hpp.

7.13.2.3 Geese() [3/4]

```
Geese::Geese (
            const Geese & model_,
            bool copy_data = true )  [inline]
```

Definition at line 157 of file geese-meat-constructors.hpp.

7.13.2.4 Geese() [4/4]

```
Geese::Geese (
            Geese && x )  [inline], [noexcept]
```

Definition at line 232 of file geese-meat-constructors.hpp.

7.13.2.5 ~Geese()

```
Geese::~Geese ( )  [inline]
```

Definition at line 71 of file geese-meat.hpp.

7.13.3 Member Function Documentation

7.13.3.1 calc_reduced_sequence()

```
void Geese::calc_reduced_sequence ( )  [inline]
```

Definition at line 234 of file geese-meat.hpp.

7.13.3.2 calc_sequence()

```
void Geese::calc_sequence (
            Node * n = nullptr )  [inline]
```

Definition at line 191 of file geese-meat.hpp.

7.13.3.3 get_probabilities()

```
std::vector< double > Geese::get_probabilities ( ) const  [inline]
```

Definition at line 277 of file geese-meat.hpp.

7.13.3.4 inherit_support()

```
void Geese::inherit_support (
            const Geese & model_,
            bool delete_support_ = false )  [inline]
```

Definition at line 140 of file geese-meat.hpp.

7.13.3.5 init()

```
void Geese::init ( )  [inline]
```

Definition at line 83 of file geese-meat.hpp.

7.13.3.6 init_node()

```
void Geese::init_node (
            Node & n )  [inline]
```

Definition at line 6 of file geese-meat.hpp.

7.13.3.7 likelihood()

```
double Geese::likelihood (
            const std::vector< double > & par,
            bool as_log = false,
            bool use_reduced_sequence = true )  [inline]
```

Definition at line 6 of file geese-meat-likelihood.hpp.

7.13.3.8 likelihood_exhaust()

```
double Geese::likelihood_exhaust (
            const std::vector< double > & par )  [inline]
```

Definition at line 7 of file geese-meat-likelihood_exhaust.hpp.

7.13.3.9 nfuns()

```
unsigned int Geese::nfuns ( ) const  [inline], [noexcept]
```

Definition at line 293 of file geese-meat.hpp.

7.13.3.10 nleafs()

```
unsigned int Geese::nleafs ( ) const  [inline], [noexcept]
```

Definition at line 301 of file geese-meat.hpp.

7.13.3.11 nnodes()

```
unsigned int Geese::nnodes ( ) const  [inline], [noexcept]
```

Definition at line 297 of file geese-meat.hpp.

7.13.3.12 nterms()

```
unsigned int Geese::nterms ( ) const  [inline]
```

Definition at line 311 of file geese-meat.hpp.

7.13.3.13 observed_counts()

```
std::vector< std::vector< double > > Geese::observed_counts ( )  [inline]
```

Definition at line 318 of file geese-meat.hpp.

7.13.3.14 operator=() [1/2]

```
Geese& Geese::operator= (
            const Geese & model_ )  [delete]
```

7.13.3.15 operator=() [2/2]

```
Geese& Geese::operator= (
            Geese && model_ )  [delete], [noexcept]
```

7.13.3.16 predict()

```
std::vector< std::vector< double > > Geese::predict (
            const std::vector< double > & par,
            std::vector< std::vector< double > > * res_prob = nullptr,
            bool leave_one_out = false,
            bool only_annotated = false,
            bool use_reduced_sequence = true )  [inline]
```

Definition at line 166 of file geese-meat-predict.hpp.

7.13.3.17 predict_backend()

```
std::vector< std::vector< double > > Geese::predict_backend (
            const std::vector< double > & par,
            bool use_reduced_sequence,
            const std::vector< uint > & preorder )  [inline]
```

Definition at line 6 of file geese-meat-predict.hpp.

7.13.3.18 print_observed_counts()

```
void Geese::print_observed_counts ( )  [inline]
```

Definition at line 366 of file geese-meat.hpp.

7.13.3.19 set_seed()

```
void Geese::set_seed (
            const unsigned int & s )  [inline]
```

Definition at line 8 of file geese-meat-simulate.hpp.

7.13.3.20 simulate()

```
std::vector< std::vector< unsigned int > > Geese::simulate (
            const std::vector< double > & par )  [inline]
```

Definition at line 12 of file geese-meat-simulate.hpp.

7.13.3.21 update_annotations()

```
void Geese::update_annotations (
            unsigned int nodeid,
            std::vector< unsigned int > newann )  [inline]
```

Definition at line 168 of file geese-meat.hpp.

7.13.4 Member Data Documentation

7.13.4.1 counters

```
phylocounters::PhyloCounters* Geese::counters = nullptr
```

Definition at line 85 of file geese-bones.hpp.

7.13.4.2 delete_counters

```
bool Geese::delete_counters = false
```

Definition at line 102 of file geese-bones.hpp.

7.13.4.3 delete_rengine

```
bool Geese::delete_rengine = false
```

Definition at line 101 of file geese-bones.hpp.

7.13.4.4 delete_support

```
bool Geese::delete_support = false
```

Definition at line 103 of file geese-bones.hpp.

7.13.4.5 initialized

```
bool Geese::initialized = false
```

Definition at line 100 of file geese-bones.hpp.

7.13.4.6 map_to_nodes

```
barry::MapVec_type< unsigned int > Geese::map_to_nodes
```

Definition at line 93 of file geese-bones.hpp.

7.13.4.7 nfunctions

```
unsigned int Geese::nfunctions
```

Definition at line 91 of file geese-bones.hpp.

7.13.4.8 nodes

```
std::map< unsigned int, Node > Geese::nodes
```

Definition at line 92 of file geese-bones.hpp.

7.13.4.9 reduced_sequence

```
std::vector< unsigned int > Geese::reduced_sequence
```

Definition at line 97 of file geese-bones.hpp.

7.13.4.10 rengine

```
std::mt19937* Geese::rengine = nullptr
```

Definition at line 84 of file geese-bones.hpp.

7.13.4.11 sequence

```
std::vector< unsigned int > Geese::sequence
```

Definition at line 96 of file geese-bones.hpp.

7.13.4.12 states

```
std::vector< std::vector< bool > > Geese::states
```

Definition at line 87 of file geese-bones.hpp.

7.13.4.13 support

```
phylocounters::PhyloModel* Geese::support = nullptr
```

Definition at line 86 of file geese-bones.hpp.

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

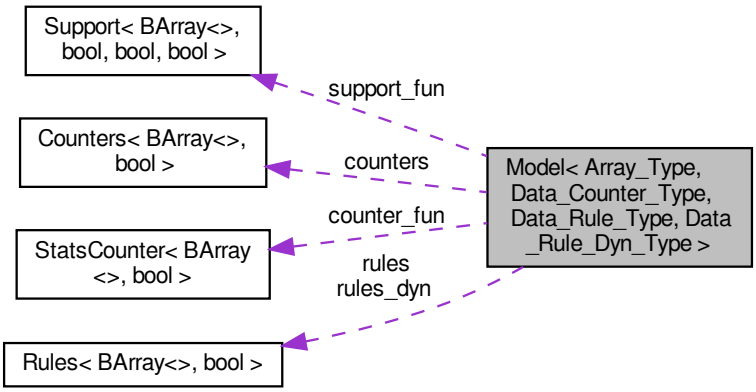
- include/barry/models/geese/geese-bones.hpp
- include/barry/models/geese/geese-meat-constructors.hpp
- include/barry/models/geese/geese-meat-likelihood.hpp
- include/barry/models/geese/geese-meat-likelihood_exhaust.hpp
- include/barry/models/geese/geese-meat-predict.hpp
- include/barry/models/geese/geese-meat-simulate.hpp
- include/barry/models/geese/geese-meat.hpp

7.14 Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type > Class Template Reference

General framework for discrete exponential models. This class allows generating discrete exponential models in the form of a linear exponential model:

`#include <model-bones.hpp>`

Collaboration diagram for Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >:



Public Member Functions

- Model ()
- Model (uint size_)
- Model (const Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type > &Model↩ _)
- Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type > & operator= (const Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type > &Model_)
- ~Model ()
- void store_psets () noexcept
- void set_keygen (std::function< std::vector< double >(const Array_Type &)> keygen_)
- uint add_array (const Array_Type &Array_, bool force_new=false)

  Adds an array to the support of not already included.
- void print_stats (uint i) const
- Array_Type sample (const Array_Type &Array_, const std::vector< double > &params={})
- Array_Type sample (const uint &i, const std::vector< double > &params)

Wrappers for the <tt>Counters</tt> member.

These will add counters to the model, which are shared by the support and the actual counter function.

- void add_counter (Counter< Array_Type, Data_Counter_Type > &counter)
- void add_counter (Counter< Array_Type, Data_Counter_Type > *counter)
- void add_counter (Counter_fun_type< Array_Type, Data_Counter_Type > count_fun_, Counter_fun_type< Array_Type, Data_Counter_Type > init_fun_=nullptr, Data_Counter_Type *data_=nullptr, bool delete_↩ data_=false)

- void set_counters (Counters< Array_Type, Data_Counter_Type > *counters_)

Wrappers for the <tt>Rules</tt> member.

These will add rules to the model, which are shared by the support and the actual counter function.

- void add_rule (Rule< Array_Type, Data_Rule_Type > &rule)
- void add_rule (Rule< Array_Type, Data_Rule_Type > *rule)
- void add_rule (Rule_fun_type< Array_Type, Data_Rule_Type > count_fun_, Data_Rule_Type *data_=nullptr, bool delete_data_=false)
- void set_rules (Rules< Array_Type, Data_Rule_Type > *rules_)
- void add_rule_dyn (Rule< Array_Type, Data_Rule_Dyn_Type > &rule)
- void add_rule_dyn (Rule< Array_Type, Data_Rule_Dyn_Type > *rule)
- void add_rule_dyn (Rule_fun_type< Array_Type, Data_Rule_Dyn_Type > count_fun_, Data_Rule_Dyn_Type *data_=nullptr, bool delete_data_=false)
- void set_rules_dyn (Rules< Array_Type, Data_Rule_Dyn_Type > *rules_)

Likelihood functions.

Calculation of likelihood functions is done reusing normalizing constants. Before recalculating the normalizing constant, the function checks whether `params` matches the last set vector of parameters used to compute it.

Parameters

params	*Vector of parameters*
as_log	*When `true`, the function returns the log-likelihood.*

- double likelihood (const std::vector< double > &params, const uint &i, bool as_log=false)
- double likelihood (const std::vector< double > &params, const Array_Type &Array_, int i=-1, bool as_log=false)
- double likelihood (const std::vector< double > &params, const std::vector< double > &target_, const uint &i, bool as_log=false)
- double likelihood_total (const std::vector< double > &params, bool as_log=false)

Extract elements by index

Parameters

i	*Index relative to the array in the model.*
params	*A new vector of model parameters to compute the normalizing constant.*
as_log	*When `true` returns the logged version of the normalizing constant.*

- double get_norm_const (const std::vector< double > &params, const uint &i, bool as_log=false)
- const std::vector< Array_Type > * get_pset (const uint &i)
- const std::vector< std::vector< double > > * get_stats (const uint &i)

Size of the model

Number of different supports included in the model

This will return the size of `stats`.

Returns

`size()` returns the number of arrays in the model.

`size_unique()` returns the number of unique arrays (according to the hasher) in the model.

`nterms()` returns the number of terms in the model.

- unsigned int size () const noexcept
- unsigned int size_unique () const noexcept
- unsigned int nterms () const noexcept

Public Attributes

- std::vector< Counts_type > stats
- std::vector< uint > n_arrays_per_stats
- MapVec_type< double, uint > keys2support

  Map of types of arrays to support sets.
- std::vector< std::vector< double > > params_last

  Vector of the previously used parameters.
- std::vector< double > normalizing_constants
- std::vector< bool > first_calc_done
- std::function< std::vector< double >const Array_Type &)> keygen = nullptr

  Function to extract features of the array to be hash.

Container space for the powerset (and its sufficient stats)

This is useful in the case of using simulations or evaluating functions that need to account for the full set of states.

- bool with_pset = false
- std::vector< std::vector< Array_Type > > pset_arrays
- std::vector< std::vector< std::vector< double > > > pset_stats
- std::vector< std::vector< double > > pset_probs

Information about the arrays used in the model

`target_stats` *holds the observed sufficient statistics for each array in the dataset.* `array_frequency` *contains the frequency with which each of the target stats (arrays) shows in the support.* `array2support` *maps array indices (0, 1, ...) to the corresponding support.*

- std::vector< std::vector< double > > target_stats
- std::vector< uint > array_frequency
- std::vector< uint > arrays2support

Functions to compute statistics

Arguments are recycled to save memory and computation.

- Counters< Array_Type, Data_Counter_Type > counters
- Rules< Array_Type, Data_Rule_Type > rules
- Rules< Array_Type, Data_Rule_Dyn_Type > rules_dyn
- Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type > support_fun
- StatsCounter< Array_Type, Data_Counter_Type > counter_fun

Random number generation

Random number generation

- std::mt19937 * rengine = nullptr
- bool delete_rengine = false
- void set_rengine (std::mt19937 *rengine_, bool delete_=false)
- void set_seed (unsigned int s)

7.14.1 Detailed Description

template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
class Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >

General framework for discrete exponential models. This class allows generating discrete exponential models in the form of a linear exponential model:

$$\frac{\exp\left(\theta^{\mathsf{t}} c(A)\right)}{\sum_{A' \in \mathcal{A}} \exp\left(\theta^{\mathsf{t}} c(A')\right)}$$

This implementation aims to reduce the number of times that the support needs to be computed. Models included here use more than a single array, and thus allow the function to recycle support sets as needed. For example, if we are looking at directed graphs all of the same size and without vertex level features, i.e. a model that only counts edges, triangles, etc. then the support needs to be fully computed only once.

Template Parameters

Array_Type	Class of `BArray` object.
Data_Counter_Type	Any type.
Data_Rule_Type	Any type.

Definition at line 104 of file model-bones.hpp.

7.14.2 Constructor & Destructor Documentation

7.14.2.1 Model() [1/3]

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::Model  [inline]
```

Definition at line 7 of file model-meat.hpp.

7.14.2.2 Model() [2/3]

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::Model (
            uint size_ )  [inline]
```

Definition at line 27 of file model-meat.hpp.

7.14.2.3 Model() [3/3]

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::Model (
            const Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >
& Model_ )  [inline]
```

Definition at line 50 of file model-meat.hpp.

7.14.2.4 ~Model()

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::~Model ( )  [inline]
```

Definition at line 200 of file model-bones.hpp.

7.14.3 Member Function Documentation

7.14.3.1 add_array()

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
uint Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::add_array (
            const Array_Type & Array_,
            bool force_new = false )  [inline]
```

Adds an array to the support of not already included.

Parameters

*Array_*	array to be added
force_new	If `false`, it will use `keygen` to obtain a double vector and create a hash of it. If the hash has been computed earlier, the support is recycled.

Returns

The number of the array.

When computing with the powerset, we need to grow the corresponding vectors on the fly

Definition at line 286 of file model-meat.hpp.

7.14.3.2 add_counter() [1/3]

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
void Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::add_counter (
            Counter< Array_Type, Data_Counter_Type > & counter )  [inline]
```

Definition at line 134 of file model-meat.hpp.

7.14.3.3 add_counter() [2/3]

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
void Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::add_counter (
            Counter< Array_Type, Data_Counter_Type > * counter )  [inline]
```

Definition at line 143 of file model-meat.hpp.

7.14.3.4 add_counter() [3/3]

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
void Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::add_counter (
            Counter_fun_type< Array_Type, Data_Counter_Type > count_fun_,
            Counter_fun_type< Array_Type, Data_Counter_Type > init_fun_ = nullptr,
            Data_Counter_Type * data_ = nullptr,
            bool delete_data_ = false )  [inline]
```

Definition at line 153 of file model-meat.hpp.

7.14.3.5 add_rule() [1/3]

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
void Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::add_rule (
            Rule< Array_Type, Data_Rule_Type > & rule )  [inline]
```

Definition at line 187 of file model-meat.hpp.

7.14.3.6 add_rule() [2/3]

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
void Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::add_rule (
            Rule< Array_Type, Data_Rule_Type > * rule )  [inline]
```

Definition at line 196 of file model-meat.hpp.

7.14.3.7 add_rule() [3/3]

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
void Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::add_rule (
            Rule_fun_type< Array_Type, Data_Rule_Type > count_fun_,
            Data_Rule_Type * data_ = nullptr,
            bool delete_data_ = false )  [inline]
```

Definition at line 206 of file model-meat.hpp.

7.14.3.8 add_rule_dyn() [1/3]

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
void Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::add_rule_dyn
(
            Rule< Array_Type, Data_Rule_Dyn_Type > & rule )  [inline]
```

Definition at line 236 of file model-meat.hpp.

7.14.3.9 add_rule_dyn() [2/3]

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
void Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::add_rule_dyn
(
            Rule< Array_Type, Data_Rule_Dyn_Type > * rule )  [inline]
```

Definition at line 245 of file model-meat.hpp.

7.14.3.10 add_rule_dyn() [3/3]

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
void Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::add_rule_dyn
(
            Rule_fun_type< Array_Type, Data_Rule_Dyn_Type > count_fun_,
            Data_Rule_Dyn_Type * data_ = nullptr,
            bool delete_data_ = false )  [inline]
```

Definition at line 255 of file model-meat.hpp.

7.14.3.11 get_norm_const()

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
double Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::get_norm_↩
const (
            const std::vector< double > & params,
            const uint & i,
            bool as_log = false )  [inline]
```

Definition at line 517 of file model-meat.hpp.

7.14.3.12 get_pset()

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
const std::vector< Array_Type > * Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data↩
_Rule_Dyn_Type >::get_pset (
            const uint & i )  [inline]
```

Definition at line 549 of file model-meat.hpp.

7.14.3.13 get_stats()

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
const std::vector< std::vector< double > > * Model< Array_Type, Data_Counter_Type, Data_↩
Rule_Type, Data_Rule_Dyn_Type >::get_stats (
            const uint & i )  [inline]
```

Definition at line 562 of file model-meat.hpp.

7.14.3.14 likelihood() [1/3]

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
double Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::likelihood
(
            const std::vector< double > & params,
            const Array_Type & Array_,
            int i = -1,
            bool as_log = false )  [inline]
```

Definition at line 403 of file model-meat.hpp.

7.14.3.15 likelihood() [2/3]

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
double Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::likelihood
(
            const std::vector< double > & params,
            const std::vector< double > & target_,
            const uint & i,
            bool as_log = false )  [inline]
```

Definition at line 443 of file model-meat.hpp.

7.14.3.16 likelihood() [3/3]

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
double Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::likelihood
(
            const std::vector< double > & params,
            const uint & i,
            bool as_log = false )  [inline]
```

Definition at line 370 of file model-meat.hpp.

7.14.3.17 likelihood_total()

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
double Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::likelihood↩
_total (
            const std::vector< double > & params,
            bool as_log = false )  [inline]
```

Definition at line 477 of file model-meat.hpp.

7.14.3.18 nterms()

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
uint Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::nterms  [inline],
[noexcept]
```

Definition at line 603 of file model-meat.hpp.

7.14.3.19 operator=()

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type > & Model< Array_↩
Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::operator= (
            const Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >
& Model_ )  [inline]
```

Definition at line 82 of file model-meat.hpp.

7.14.3.20 print_stats()

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
void Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::print_stats (
            uint i ) const  [inline]
```

Definition at line 574 of file model-meat.hpp.

7.14.3.21 sample() [1/2]

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
Array_Type Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::sample
(
            const Array_Type & Array_,
            const std::vector< double > & params = {} )
```

7.14.3.22 sample() [2/2]

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
Array_Type Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::sample
(
            const uint & i,
            const std::vector< double > & params )  [inline]
```

Definition at line 610 of file model-meat.hpp.

7.14.3.23 set_counters()

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
void Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::set_counters
(
            Counters< Array_Type, Data_Counter_Type > * counters_ )  [inline]
```

Definition at line 172 of file model-meat.hpp.

7.14.3.24 set_keygen()

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
void Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::set_keygen (
            std::function< std::vector< double >(const Array_Type &)> keygen_ )  [inline]
```

Definition at line 126 of file model-meat.hpp.

7.14.3.25 set_rengine()

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
void Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::set_rengine (
            std::mt19937 * rengine_,
            bool delete_ = false )  [inline]
```

Definition at line 115 of file model-bones.hpp.

7.14.3.26 set_rules()

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
void Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::set_rules (
            Rules< Array_Type, Data_Rule_Type > * rules_ )  [inline]
```

Definition at line 223 of file model-meat.hpp.

7.14.3.27 set_rules_dyn()

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
void Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::set_rules_dyn
(
            Rules< Array_Type, Data_Rule_Dyn_Type > * rules_ )  [inline]
```

Definition at line 272 of file model-meat.hpp.

7.14.3.28 set_seed()

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
void Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::set_seed (
            unsigned int s )  [inline]
```

Definition at line 125 of file model-bones.hpp.

7.14.3.29 size()

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
uint Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::size  [inline],
[noexcept]
```

Definition at line 593 of file model-meat.hpp.

7.14.3.30 size_unique()

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
uint Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::size_unique
[inline], [noexcept]
```

Definition at line 598 of file model-meat.hpp.

7.14.3.31 store_psets()

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
void Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::store_psets
[inline], [noexcept]
```

Definition at line 118 of file model-meat.hpp.

7.14.4 Member Data Documentation

7.14.4.1 array_frequency

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
std::vector< uint > Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type
>::array_frequency
```

Definition at line 163 of file model-bones.hpp.

7.14.4.2 arrays2support

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
std::vector< uint > Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type
>::arrays2support
```

Definition at line 164 of file model-bones.hpp.

7.14.4.3 counter_fun

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
StatsCounter<Array_Type,Data_Counter_Type> Model< Array_Type, Data_Counter_Type, Data_Rule_↩
Type, Data_Rule_Dyn_Type >::counter_fun
```

Definition at line 182 of file model-bones.hpp.

7.14.4.4 counters

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
Counters<Array_Type,Data_Counter_Type> Model< Array_Type, Data_Counter_Type, Data_Rule_Type,
Data_Rule_Dyn_Type >::counters
```

Definition at line 178 of file model-bones.hpp.

7.14.4.5 delete_rengine

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
bool Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::delete_↩
rengine = false
```

Definition at line 114 of file model-bones.hpp.

7.14.4.6 first_calc_done

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
std::vector< bool > Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type
>::first_calc_done
```

Definition at line 188 of file model-bones.hpp.

7.14.4.7 keygen

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
std::function<std::vector<double>const Array_Type &)> Model< Array_Type, Data_Counter_Type,
Data_Rule_Type, Data_Rule_Dyn_Type >::keygen = nullptr
```

Function to extract features of the array to be hash.

Definition at line 192 of file model-bones.hpp.

7.14.4.8 keys2support

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
MapVec_type< double, uint > Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule↩
_Dyn_Type >::keys2support
```

Map of types of arrays to support sets.

This is of the same length as the vector `stats`.

Definition at line 171 of file model-bones.hpp.

7.14.4.9 n_arrays_per_stats

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
std::vector< uint > Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type
>::n_arrays_per_stats
```

Definition at line 140 of file model-bones.hpp.

7.14.4.10 normalizing_constants

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
std::vector< double > Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_↩
Type >::normalizing_constants
```

Definition at line 187 of file model-bones.hpp.

7.14.4.11 params_last

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
std::vector< std::vector<double> > Model< Array_Type, Data_Counter_Type, Data_Rule_Type,
Data_Rule_Dyn_Type >::params_last
```

Vector of the previously used parameters.

Definition at line 186 of file model-bones.hpp.

7.14.4.12 pset_arrays

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
std::vector< std::vector< Array_Type > > Model< Array_Type, Data_Counter_Type, Data_Rule_↩
Type, Data_Rule_Dyn_Type >::pset_arrays
```

Definition at line 149 of file model-bones.hpp.

7.14.4.13 pset_probs

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
std::vector< std::vector<double> > Model< Array_Type, Data_Counter_Type, Data_Rule_Type,
Data_Rule_Dyn_Type >::pset_probs
```

Definition at line 151 of file model-bones.hpp.

7.14.4.14 pset_stats

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
std::vector< std::vector< std::vector<double> > > Model< Array_Type, Data_Counter_Type,
Data_Rule_Type, Data_Rule_Dyn_Type >::pset_stats
```

Definition at line 150 of file model-bones.hpp.

7.14.4.15 rengine

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
std::mt19937* Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >↩
::rengine = nullptr
```

Definition at line 113 of file model-bones.hpp.

7.14.4.16 rules

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
Rules<Array_Type,Data_Rule_Type> Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data↩
_Rule_Dyn_Type >::rules
```

Definition at line 179 of file model-bones.hpp.

7.14.4.17 rules_dyn

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
Rules<Array_Type,Data_Rule_Dyn_Type> Model< Array_Type, Data_Counter_Type, Data_Rule_Type,
Data_Rule_Dyn_Type >::rules_dyn
```

Definition at line 180 of file model-bones.hpp.

7.14.4.18 stats

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
std::vector< Counts_type > Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_↩
Dyn_Type >::stats
```

Definition at line 139 of file model-bones.hpp.

7.14.4.19 support_fun

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
Support<Array_Type,Data_Counter_Type,Data_Rule_Type,Data_Rule_Dyn_Type> Model< Array_Type,
Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::support_fun
```

Definition at line 181 of file model-bones.hpp.

7.14.4.20 target_stats

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
std::vector< std::vector< double > > Model< Array_Type, Data_Counter_Type, Data_Rule_Type,
Data_Rule_Dyn_Type >::target_stats
```

Definition at line 162 of file model-bones.hpp.

7.14.4.21 with_pset

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
bool Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::with_pset =
false
```

Definition at line 148 of file model-bones.hpp.

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

- include/barry/model-bones.hpp
- include/barry/model-meat.hpp

7.15 NetCounterData Class Reference

Data class used to store arbitrary uint or double vectors.

`#include <network.hpp>`

Public Member Functions

- NetCounterData ()
- NetCounterData (const std::vector< uint > indices_, const std::vector< double > numbers_)
- ~NetCounterData ()

Public Attributes

- std::vector< uint > indices
- std::vector< double > numbers

7.15.1 Detailed Description

Data class used to store arbitrary uint or double vectors.

Definition at line 61 of file network.hpp.

7.15.2 Constructor & Destructor Documentation

7.15.2.1 NetCounterData() [1/2]

```
NetCounterData::NetCounterData ( )  [inline]
```

Definition at line 67 of file network.hpp.

7.15.2.2 NetCounterData() [2/2]

```
NetCounterData::NetCounterData (
            const std::vector< uint > indices_,
            const std::vector< double > numbers_ )  [inline]
```

Definition at line 68 of file network.hpp.

7.15.2.3 ~NetCounterData()

```
NetCounterData::~NetCounterData ( )  [inline]
```

Definition at line 73 of file network.hpp.

7.15.3 Member Data Documentation

7.15.3.1 indices

```
std::vector< uint > NetCounterData::indices
```

Definition at line 64 of file network.hpp.

7.15.3.2 numbers

```
std::vector< double > NetCounterData::numbers
```

Definition at line 65 of file network.hpp.

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

- include/barry/counters/network.hpp

7.16 NetworkData Class Reference

Data class for Networks.

```
#include <network.hpp>
```

Public Member Functions

- NetworkData ()
- NetworkData (std::vector< double > vertex_attr_, bool directed_=true)

  Constructor using a single attribute.
- NetworkData (std::vector< std::vector< double > > vertex_attr_, bool directed_=true)

  Constructor using multiple attributes.
- ~NetworkData ()

Public Attributes

- bool directed = true
- std::vector< std::vector< double > > vertex_attr

7.16.1 Detailed Description

Data class for Networks.

Details on the available counters for `NetworkData` can be found in the Network counters section.

This holds information about whether the graph is directed or not, and, if defined, vectors of node (vertex) attributes (`vertex_attr`).

Definition at line 24 of file network.hpp.

7.16.2 Constructor & Destructor Documentation

7.16.2.1 NetworkData() [1/3]

```
NetworkData::NetworkData ( )  [inline]
```

Definition at line 30 of file network.hpp.

7.16.2.2 NetworkData() [2/3]

```
NetworkData::NetworkData (
            std::vector< double > vertex_attr_,
            bool directed_ = true )  [inline]
```

Constructor using a single attribute.

Parameters

*vertex_attr_*	Double vector of length equal to the number of vertices in the data.
*directed_*	When `true` the graph as treated as directed.

Definition at line 38 of file network.hpp.

7.16.2.3 NetworkData() [3/3]

```
NetworkData::NetworkData (
            std::vector< std::vector< double > > vertex_attr_,
            bool directed_ = true )  [inline]
```

Constructor using multiple attributes.

Parameters

*vertex_↩ attr_*	Vector of double vectors. The size equals to the number of attributes to be created. Each individual vector should be of length equal to the number of vertices.
*directed_*	When `true` the graph as treated as directed.

Definition at line 50 of file network.hpp.

7.16.2.4 ~NetworkData()

```
NetworkData::~NetworkData ( )  [inline]
```

Definition at line 56 of file network.hpp.

7.16.3 Member Data Documentation

7.16.3.1 directed

```
bool NetworkData::directed = true
```

Definition at line 27 of file network.hpp.

7.16.3.2 vertex_attr

```
std::vector< std::vector< double > > NetworkData::vertex_attr
```

Definition at line 28 of file network.hpp.

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

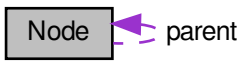
- include/barry/counters/network.hpp

7.17 Node Class Reference

A single node for the model.

`#include <geese-node-bones.hpp>`

Collaboration diagram for Node:



Public Member Functions

- ~Node ()
- int get_parent () const
- bool is_leaf () const noexcept

Construct a new Node object

- Node ()
- Node (unsigned int id_, unsigned int ord_, bool duplication_)
- Node (unsigned int id_, unsigned int ord_, std::vector< unsigned int > annotations_, bool duplication_)
- Node (Node &&x) noexcept
- Node (const Node &x)

Public Attributes

- unsigned int id

  Id of the node (as specified in the input)
- unsigned int ord

  Order in which the node was created.
- phylocounters::PhyloArray array
- std::vector< unsigned int > annotations

  Observed annotations (only defined for Geese)
- bool duplication
- std::vector< phylocounters::PhyloArray > arrays = {}

  Arrays given all possible states.
- Node * parent = nullptr

  Parent node.
- std::vector< Node * > offspring = {}

  Offspring nodes.
- std::vector< unsigned int > narray = {}

  ID of the array in the model.
- bool visited = false
- std::vector< double > subtree_prob

  Induced subtree probabilities.
- std::vector< double > probability

  The probability of observing each state.

7.17.1 Detailed Description

A single node for the model.

Each node contains all the information to compute the conditional probability of the pruning algorithm at that node.

Definition at line 11 of file geese-node-bones.hpp.

7.17.2 Constructor & Destructor Documentation

7.17.2.1 Node() [1/5]

```
Node::Node ( )  [inline]
```

Definition at line 36 of file geese-node-bones.hpp.

7.17.2.2 Node() [2/5]

```
Node::Node (
            unsigned int id_,
            unsigned int ord_,
            bool duplication_ )  [inline]
```

Definition at line 55 of file geese-node-bones.hpp.

7.17.2.3 Node() [3/5]

```
Node::Node (
            unsigned int id_,
            unsigned int ord_,
            std::vector< unsigned int > annotations_,
            bool duplication_ )  [inline]
```

Definition at line 61 of file geese-node-bones.hpp.

7.17.2.4 Node() [4/5]

```
Node::Node (
            Node && x )  [inline], [noexcept]
```

Definition at line 68 of file geese-node-bones.hpp.

7.17.2.5 Node() [5/5]

```
Node::Node (
            const Node & x )  [inline]
```

Definition at line 82 of file geese-node-bones.hpp.

7.17.2.6 ~Node()

```
Node::~Node ( )  [inline]
```

Definition at line 47 of file geese-node-bones.hpp.

7.17.3 Member Function Documentation

7.17.3.1 get_parent()

```
int Node::get_parent ( ) const  [inline]
```

Definition at line 96 of file geese-node-bones.hpp.

7.17.3.2 is_leaf()

```
bool Node::is_leaf ( ) const  [inline], [noexcept]
```

Definition at line 103 of file geese-node-bones.hpp.

7.17.4 Member Data Documentation

7.17.4.1 annotations

```
std::vector< unsigned int > Node::annotations
```

Observed annotations (only defined for Geese)

Definition at line 18 of file geese-node-bones.hpp.

7.17.4.2 array

```
phylocounters::PhyloArray Node::array
```

Definition at line 17 of file geese-node-bones.hpp.

7.17.4.3 arrays

```
std::vector< phylocounters::PhyloArray > Node::arrays = {}
```

Arrays given all possible states.

Definition at line 21 of file geese-node-bones.hpp.

7.17.4.4 duplication

```
bool Node::duplication
```

Definition at line 19 of file geese-node-bones.hpp.

7.17.4.5 id

```
unsigned int Node::id
```

Id of the node (as specified in the input)

Definition at line 14 of file geese-node-bones.hpp.

7.17.4.6 narray

```
std::vector< unsigned int > Node::narray = {}
```

ID of the array in the model.

Definition at line 24 of file geese-node-bones.hpp.

7.17.4.7 offspring

```
std::vector< Node* > Node::offspring = {}
```

Offspring nodes.

Definition at line 23 of file geese-node-bones.hpp.

7.17.4.8 ord

```
unsigned int Node::ord
```

Order in which the node was created.

Definition at line 15 of file geese-node-bones.hpp.

7.17.4.9 parent

```
Node* Node::parent = nullptr
```

Parent node.

Definition at line 22 of file geese-node-bones.hpp.

7.17.4.10 probability

```
std::vector< double > Node::probability
```

The probability of observing each state.

Definition at line 28 of file geese-node-bones.hpp.

7.17.4.11 subtree_prob

```
std::vector< double > Node::subtree_prob
```

Induced subtree probabilities.

Definition at line 27 of file geese-node-bones.hpp.

7.17.4.12 visited

```
bool Node::visited = false
```

Definition at line 25 of file geese-node-bones.hpp.

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

- include/barry/models/geese/geese-node-bones.hpp

7.18 NodeData Class Reference

Data definition for the `PhyloArray` class.

`#include <phylo.hpp>`

Public Member Functions

- NodeData ()
- NodeData (const std::vector< double > &blengths_, const std::vector< bool > &states_, bool duplication_=true)
- ~NodeData ()

Public Attributes

- std::vector< double > blengths
- std::vector< bool > states
- bool duplication = true

7.18.1 Detailed Description

Data definition for the `PhyloArray` class.

Details about the available counters for `PhyloArray` objects can be found in the Phylo counters section.

This holds basic information about a given node.

Definition at line 23 of file phylo.hpp.

7.18.2 Constructor & Destructor Documentation

7.18.2.1 NodeData() [1/2]

```
NodeData::NodeData ( )  [inline]
```

Definition at line 41 of file phylo.hpp.

7.18.2.2 NodeData() [2/2]

```
NodeData::NodeData (
            const std::vector< double > & blengths_,
            const std::vector< bool > & states_,
            bool duplication_ = true )  [inline]
```

Definition at line 43 of file phylo.hpp.

7.18.2.3 ~NodeData()

```
NodeData::~NodeData ( )  [inline]
```

Definition at line 49 of file phylo.hpp.

7.18.3 Member Data Documentation

7.18.3.1 blengths

```
std::vector< double > NodeData::blengths
```

Branch length.

Definition at line 29 of file phylo.hpp.

7.18.3.2 duplication

```
bool NodeData::duplication = true
```

Definition at line 39 of file phylo.hpp.

7.18.3.3 states

```
std::vector< bool > NodeData::states
```

State of the parent node.

Definition at line 34 of file phylo.hpp.

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

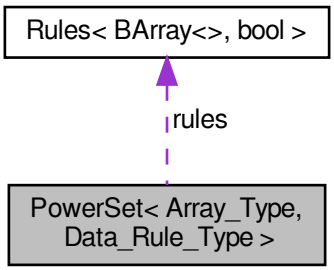
- include/barry/counters/phylo.hpp

7.19 PowerSet< Array_Type, Data_Rule_Type > Class Template Reference

Powerset of a binary array.

`#include <powerset-bones.hpp>`

Collaboration diagram for PowerSet< Array_Type, Data_Rule_Type >:



Public Member Functions

- void init_support ()
- void calc ()
- void reset (uint N_, uint M_)

Construct and destroy a PowerSet object

- PowerSet ()
- PowerSet (uint N_, uint M_)
- PowerSet (const Array_Type &array)
- ~PowerSet ()

Wrappers for the <tt>Rules</tt> member.

These will add rules to the model, which are shared by the support and the actual counter function.

- void add_rule (Rule< Array_Type, Data_Rule_Type > &rule)
- void add_rule (Rule< Array_Type, Data_Rule_Type > *rule)
- void add_rule (Rule_fun_type< Array_Type, Data_Rule_Type > count_fun_, Data_Rule_Type *data_=nullptr, bool delete_data_=false)

Getter functions

- const std::vector< Array_Type > * get_data_ptr () const
- std::vector< Array_Type > get_data () const
- std::vector< Array_Type >::iterator begin ()
- std::vector< Array_Type >::iterator end ()
- std::size_t size () const noexcept
- const Array_Type & operator[] (const unsigned int &i) const

Public Attributes

- Array_Type EmptyArray
- std::vector< Array_Type > data
- Rules< Array_Type, Data_Rule_Type > * rules
- uint N
- uint M
- bool rules_deleted = false
- std::vector< std::pair< uint, uint > > coordinates_free
- std::vector< std::pair< uint, uint > > coordinates_locked

7.19.1 Detailed Description

template<typename Array_Type = BArray<>, typename Data_Rule_Type = bool>
class PowerSet< Array_Type, Data_Rule_Type >

Powerset of a binary array.

Template Parameters

Array_Type	
Data_Rule_Type	

Definition at line 17 of file powerset-bones.hpp.

7.19.2 Constructor & Destructor Documentation

7.19.2.1 PowerSet() [1/3]

```
template<typename Array_Type = BArray<>, typename Data_Rule_Type = bool>
PowerSet< Array_Type, Data_Rule_Type >::PowerSet ( )  [inline]
```

Definition at line 39 of file powerset-bones.hpp.

7.19.2.2 PowerSet() [2/3]

```
template<typename Array_Type = BArray<>, typename Data_Rule_Type = bool>
PowerSet< Array_Type, Data_Rule_Type >::PowerSet (
            uint N_,
            uint M_ )  [inline]
```

Definition at line 41 of file powerset-bones.hpp.

7.19.2.3 PowerSet() [3/3]

```
template<typename Array_Type , typename Data_Rule_Type >
PowerSet< Array_Type, Data_Rule_Type >::PowerSet (
            const Array_Type & array )  [inline]
```

Definition at line 7 of file powerset-meat.hpp.

7.19.2.4 ~PowerSet()

```
template<typename Array_Type , typename Data_Rule_Type >
PowerSet< Array_Type, Data_Rule_Type >::~PowerSet  [inline]
```

Definition at line 15 of file powerset-meat.hpp.

7.19.3 Member Function Documentation

7.19.3.1 add_rule() [1/3]

```
template<typename Array_Type , typename Data_Rule_Type >
void PowerSet< Array_Type, Data_Rule_Type >::add_rule (
            Rule< Array_Type, Data_Rule_Type > & rule )  [inline]
```

Definition at line 113 of file powerset-meat.hpp.

7.19.3.2 add_rule() [2/3]

```
template<typename Array_Type , typename Data_Rule_Type >
void PowerSet< Array_Type, Data_Rule_Type >::add_rule (
            Rule< Array_Type, Data_Rule_Type > * rule )  [inline]
```

Definition at line 122 of file powerset-meat.hpp.

7.19.3.3 add_rule() [3/3]

```
template<typename Array_Type , typename Data_Rule_Type >
void PowerSet< Array_Type, Data_Rule_Type >::add_rule (
            Rule_fun_type< Array_Type, Data_Rule_Type > count_fun_,
            Data_Rule_Type * data_ = nullptr,
            bool delete_data_ = false )  [inline]
```

Definition at line 132 of file powerset-meat.hpp.

7.19.3.4 begin()

```
template<typename Array_Type = BArray<>, typename Data_Rule_Type = bool>
std::vector< Array_Type >::iterator PowerSet< Array_Type, Data_Rule_Type >::begin ( )  [inline]
```

Definition at line 73 of file powerset-bones.hpp.

7.19.3.5 calc()

```
template<typename Array_Type , typename Data_Rule_Type >
void PowerSet< Array_Type, Data_Rule_Type >::calc  [inline]
```

Definition at line 88 of file powerset-meat.hpp.

7.19.3.6 end()

```
template<typename Array_Type = BArray<>, typename Data_Rule_Type = bool>
std::vector< Array_Type >::iterator PowerSet< Array_Type, Data_Rule_Type >::end ( )  [inline]
```

Definition at line 74 of file powerset-bones.hpp.

7.19.3.7 get_data()

```
template<typename Array_Type = BArray<>, typename Data_Rule_Type = bool>
std::vector< Array_Type > PowerSet< Array_Type, Data_Rule_Type >::get_data ( ) const  [inline]
```

Definition at line 72 of file powerset-bones.hpp.

7.19.3.8 get_data_ptr()

```
template<typename Array_Type = BArray<>, typename Data_Rule_Type = bool>
const std::vector< Array_Type >* PowerSet< Array_Type, Data_Rule_Type >::get_data_ptr ( )
const  [inline]
```

Definition at line 71 of file powerset-bones.hpp.

7.19.3.9 init_support()

```
template<typename Array_Type , typename Data_Rule_Type >
void PowerSet< Array_Type, Data_Rule_Type >::init_support  [inline]
```

Definition at line 21 of file powerset-meat.hpp.

7.19.3.10 operator[]()

```
template<typename Array_Type = BArray<>, typename Data_Rule_Type = bool>
const Array_Type& PowerSet< Array_Type, Data_Rule_Type >::operator[] (
            const unsigned int & i ) const  [inline]
```

Definition at line 76 of file powerset-bones.hpp.

7.19.3.11 reset()

```
template<typename Array_Type , typename Data_Rule_Type >
void PowerSet< Array_Type, Data_Rule_Type >::reset (
            uint N_,
            uint M_ )  [inline]
```

Definition at line 101 of file powerset-meat.hpp.

7.19.3.12 size()

```
template<typename Array_Type = BArray<>, typename Data_Rule_Type = bool>
std::size_t PowerSet< Array_Type, Data_Rule_Type >::size ( ) const  [inline], [noexcept]
```

Definition at line 75 of file powerset-bones.hpp.

7.19.4 Member Data Documentation

7.19.4.1 coordinates_free

```
template<typename Array_Type = BArray<>, typename Data_Rule_Type = bool>
std::vector< std::pair<uint,uint> > PowerSet< Array_Type, Data_Rule_Type >::coordinates_free
```

Definition at line 31 of file powerset-bones.hpp.

7.19.4.2 coordinates_locked

```
template<typename Array_Type = BArray<>, typename Data_Rule_Type = bool>
std::vector< std::pair<uint,uint> > PowerSet< Array_Type, Data_Rule_Type >::coordinates_↩
locked
```

Definition at line 32 of file powerset-bones.hpp.

7.19.4.3 data

```
template<typename Array_Type = BArray<>, typename Data_Rule_Type = bool>
std::vector< Array_Type > PowerSet< Array_Type, Data_Rule_Type >::data
```

Definition at line 24 of file powerset-bones.hpp.

7.19.4.4 EmptyArray

```
template<typename Array_Type = BArray<>, typename Data_Rule_Type = bool>
Array_Type PowerSet< Array_Type, Data_Rule_Type >::EmptyArray
```

Definition at line 23 of file powerset-bones.hpp.

7.19.4.5 M

```
template<typename Array_Type = BArray<>, typename Data_Rule_Type = bool>
uint PowerSet< Array_Type, Data_Rule_Type >::M
```

Definition at line 27 of file powerset-bones.hpp.

7.19.4.6 N

```
template<typename Array_Type = BArray<>, typename Data_Rule_Type = bool>
uint PowerSet< Array_Type, Data_Rule_Type >::N
```

Definition at line 27 of file powerset-bones.hpp.

7.19.4.7 rules

```
template<typename Array_Type = BArray<>, typename Data_Rule_Type = bool>
Rules<Array_Type,Data_Rule_Type>* PowerSet< Array_Type, Data_Rule_Type >::rules
```

Definition at line 25 of file powerset-bones.hpp.

7.19.4.8 rules_deleted

```
template<typename Array_Type = BArray<>, typename Data_Rule_Type = bool>
bool PowerSet< Array_Type, Data_Rule_Type >::rules_deleted = false
```

Definition at line 28 of file powerset-bones.hpp.

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

- include/barry/powerset-bones.hpp
- include/barry/powerset-meat.hpp

7.20 Rule< Array_Type, Data_Type > Class Template Reference

Rule for determining if a cell should be included in a sequence.

```
#include <rules-bones.hpp>
```

Public Member Functions

- ~Rule ()
- Data_Type * D ()

  Read/Write access to the data.
- bool operator() (const Array_Type &a, uint i, uint j)

Construct a new Rule object

Construct a new Rule object

Parameters

fun_	*A function of type `Rule_fun_type`.*
dat_	*Data pointer to be passed to `fun_`*
delete_↩ dat_	*When `true`, the `Rule` destructor will delete the pointer, if defined.*

- Rule ()
- Rule (Rule_fun_type< Array_Type, Data_Type > fun_, Data_Type *dat_=nullptr, bool delete_dat_=false)

7.20.1 Detailed Description

template<typename Array_Type = BArray<>, typename Data_Type = bool>
class Rule< Array_Type, Data_Type >

Rule for determining if a cell should be included in a sequence.

Rules can be used together with `Support` and `PowerSet` to determine which cells should be included when enumerating all possible realizations of a binary array.

Template Parameters

Array_Type	An object of class `BArray`.
Data_Type	Any type.

Definition at line 23 of file rules-bones.hpp.

7.20.2 Constructor & Destructor Documentation

7.20.2.1 Rule() [1/2]

```
template<typename Array_Type = BArray<>, typename Data_Type = bool>
Rule< Array_Type, Data_Type >::Rule ( )  [inline]
```

Definition at line 42 of file rules-bones.hpp.

7.20.2.2 Rule() [2/2]

```
template<typename Array_Type = BArray<>, typename Data_Type = bool>
Rule< Array_Type, Data_Type >::Rule (
            Rule_fun_type< Array_Type, Data_Type > fun_,
            Data_Type * dat_ = nullptr,
            bool delete_dat_ = false )  [inline]
```

Definition at line 43 of file rules-bones.hpp.

7.20.2.3 ~Rule()

```
template<typename Array_Type = BArray<>, typename Data_Type = bool>
Rule< Array_Type, Data_Type >::~Rule ( )  [inline]
```

Definition at line 50 of file rules-bones.hpp.

7.20.3 Member Function Documentation

7.20.3.1 D()

```
template<typename Array_Type = BArray<>, typename Data_Type = bool>
Data_Type* Rule< Array_Type, Data_Type >::D ( )
```

Read/Write access to the data.

7.20.3.2 operator()()

```
template<typename Array_Type , typename Data_Type >
bool Rule< Array_Type, Data_Type >::operator() (
            const Array_Type & a,
            uint i,
            uint j )  [inline]
```

Definition at line 63 of file rules-meat.hpp.

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

- include/barry/rules-bones.hpp
- include/barry/rules-meat.hpp

7.21 Rules< Array_Type, Data_Type > Class Template Reference

Vector of objects of class Rule.

```
#include <rules-bones.hpp>
```

Public Member Functions

- Rules ()
- Rules (const Rules< Array_Type, Data_Type > &rules_)
- Rules< Array_Type, Data_Type > operator= (const Rules< Array_Type, Data_Type > &rules_)
- ~Rules ()
- uint size () const noexcept
- bool operator() (const Array_Type &a, uint i, uint j)

  Check whether a given cell is free or locked.
- void clear ()
- void get_seq (const Array_Type &a, std::vector< std::pair< uint, uint > > *free, std::vector< std::pair< uint, uint > > *locked=nullptr)

  Computes the sequence of free and locked cells in an BArray.

Rule adding

Parameters

rule	

- void add_rule (Rule< Array_Type, Data_Type > &rule)
- void add_rule (Rule< Array_Type, Data_Type > *rule)
- void add_rule (Rule_fun_type< Array_Type, Data_Type > rule_, Data_Type *data_=nullptr, bool delete_data_=false)

7.21.1 Detailed Description

template<typename Array_Type, typename Data_Type>
class Rules< Array_Type, Data_Type >

Vector of objects of class Rule.

Template Parameters

Array_Type	An object of class `BArray`
Data_Type	Any type.

Definition at line 69 of file rules-bones.hpp.

7.21.2 Constructor & Destructor Documentation

7.21.2.1 Rules() [1/2]

```
template<typename Array_Type , typename Data_Type >
Rules< Array_Type, Data_Type >::Rules ( )  [inline]
```

Definition at line 76 of file rules-bones.hpp.

7.21.2.2 Rules() [2/2]

```
template<typename Array_Type , typename Data_Type >
Rules< Array_Type, Data_Type >::Rules (
            const Rules< Array_Type, Data_Type > & rules_ )  [inline]
```

Definition at line 10 of file rules-meat.hpp.

7.21.2.3 ~Rules()

```
template<typename Array_Type , typename Data_Type >
Rules< Array_Type, Data_Type >::~Rules ( )  [inline]
```

Definition at line 81 of file rules-bones.hpp.

7.21.3 Member Function Documentation

7.21.3.1 add_rule() [1/3]

```
template<typename Array_Type , typename Data_Type >
void Rules< Array_Type, Data_Type >::add_rule (
            Rule< Array_Type, Data_Type > & rule )  [inline]
```

Definition at line 68 of file rules-meat.hpp.

7.21.3.2 add_rule() [2/3]

```
template<typename Array_Type , typename Data_Type >
void Rules< Array_Type, Data_Type >::add_rule (
            Rule< Array_Type, Data_Type > * rule )  [inline]
```

Definition at line 79 of file rules-meat.hpp.

7.21.3.3 add_rule() [3/3]

```
template<typename Array_Type , typename Data_Type >
void Rules< Array_Type, Data_Type >::add_rule (
            Rule_fun_type< Array_Type, Data_Type > rule_,
            Data_Type * data_ = nullptr,
            bool delete_data_ = false )  [inline]
```

Definition at line 89 of file rules-meat.hpp.

7.21.3.4 clear()

```
template<typename Array_Type , typename Data_Type >
void Rules< Array_Type, Data_Type >::clear  [inline]
```

Definition at line 127 of file rules-meat.hpp.

7.21.3.5 get_seq()

```
template<typename Array_Type , typename Data_Type >
void Rules< Array_Type, Data_Type >::get_seq (
            const Array_Type & a,
            std::vector< std::pair< uint, uint > > * free,
            std::vector< std::pair< uint, uint > > * locked = nullptr )  [inline]
```

Computes the sequence of free and locked cells in an BArray.

Parameters

a	An object of class BArray.
free	Pointer to a vector of pairs (i, j) listing the free cells.
locked	(optional) Pointer to a vector of pairs (i, j) listing the locked cells.

Returns

Nothing.

Definition at line 139 of file rules-meat.hpp.

7.21.3.6 operator()()

```
template<typename Array_Type , typename Data_Type >
bool Rules< Array_Type, Data_Type >::operator() (
            const Array_Type & a,
            uint i,
            uint j )  [inline]
```

Check whether a given cell is free or locked.

Parameters

a	A BArray object
i	row position
j	col position

Returns

true If the cell is locked

false If the cell is free

Definition at line 111 of file rules-meat.hpp.

7.21.3.7 operator=()

```
template<typename Array_Type , typename Data_Type >
Rules< Array_Type, Data_Type > Rules< Array_Type, Data_Type >::operator= (
            const Rules< Array_Type, Data_Type > & rules_ )
```

Definition at line 35 of file rules-meat.hpp.

7.21.3.8 size()

```
template<typename Array_Type , typename Data_Type >
uint Rules< Array_Type, Data_Type >::size ( ) const  [inline], [noexcept]
```

Definition at line 86 of file rules-bones.hpp.

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

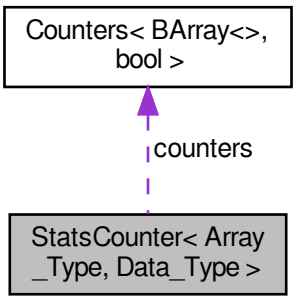
- include/barry/rules-bones.hpp
- include/barry/rules-meat.hpp

7.22 StatsCounter< Array_Type, Data_Type > Class Template Reference

Count stats for a single Array.

`#include <statscounter-bones.hpp>`

Collaboration diagram for StatsCounter< Array_Type, Data_Type >:



Public Member Functions

- StatsCounter (const Array_Type *Array_)
  Creator of a `StatsCounter`
- StatsCounter ()
  Can be created without setting the array.
- ~StatsCounter ()
- void reset_array (const Array_Type *Array_)
  Changes the reference array for the counting.
- void add_counter (Counter< Array_Type, Data_Type > *f_)
- void add_counter (Counter< Array_Type, Data_Type > f_)
- void set_counters (Counters< Array_Type, Data_Type > *counters_)
- void count_init (uint i, uint j)
  Counter functions This function recurses through the entries of `Array` and at each step of adding a new cell it uses the functions to list the statistics.
- void count_current (uint i, uint j)
- std::vector< double > count_all ()

Public Attributes

- const Array_Type * Array
- Array_Type EmptyArray
- std::vector< double > current_stats
- Counters< Array_Type, Data_Type > * counters
- bool counter_deleted = false

7.22.1 Detailed Description

template<typename Array_Type = BArray<>, typename Data_Type = bool>
class StatsCounter< Array_Type, Data_Type >

Count stats for a single Array.

Users can a list of functions that can be used with this. The baseline set of arguments is a pointer to a binary array and a dataset to add the counts to.

Definition at line 16 of file statscounter-bones.hpp.

7.22.2 Constructor & Destructor Documentation

7.22.2.1 StatsCounter() [1/2]

```
template<typename Array_Type = BArray<>, typename Data_Type = bool>
StatsCounter< Array_Type, Data_Type >::StatsCounter (
            const Array_Type * Array_ )  [inline]
```

Creator of a StatsCounter

Parameters

*Array_*	A const pointer to a BArray.

Definition at line 34 of file statscounter-bones.hpp.

7.22.2.2 StatsCounter() [2/2]

```
template<typename Array_Type = BArray<>, typename Data_Type = bool>
StatsCounter< Array_Type, Data_Type >::StatsCounter ( )  [inline]
```

Can be created without setting the array.

Definition at line 49 of file statscounter-bones.hpp.

7.22.2.3 ~StatsCounter()

```
template<typename Array_Type , typename Data_Type >
StatsCounter< Array_Type, Data_Type >::~StatsCounter  [inline]
```

Definition at line 7 of file statscounter-meat.hpp.

7.22.3 Member Function Documentation

7.22.3.1 add_counter() [1/2]

```
template<typename Array_Type , typename Data_Type >
void StatsCounter< Array_Type, Data_Type >::add_counter (
            Counter< Array_Type, Data_Type > * f_ )  [inline]
```

Definition at line 25 of file statscounter-meat.hpp.

7.22.3.2 add_counter() [2/2]

```
template<typename Array_Type , typename Data_Type >
void StatsCounter< Array_Type, Data_Type >::add_counter (
            Counter< Array_Type, Data_Type > f_ )  [inline]
```

Definition at line 35 of file statscounter-meat.hpp.

7.22.3.3 count_all()

```
template<typename Array_Type , typename Data_Type >
std::vector< double > StatsCounter< Array_Type, Data_Type >::count_all  [inline]
```

Definition at line 99 of file statscounter-meat.hpp.

7.22.3.4 count_current()

```
template<typename Array_Type , typename Data_Type >
void StatsCounter< Array_Type, Data_Type >::count_current (
            uint i,
            uint j )  [inline]
```

Definition at line 81 of file statscounter-meat.hpp.

7.22.3.5 count_init()

```
template<typename Array_Type , typename Data_Type >
void StatsCounter< Array_Type, Data_Type >::count_init (
            uint i,
            uint j )  [inline]
```

Counter functions This function recurses through the entries of `Array` and at each step of adding a new cell it uses the functions to list the statistics.

Definition at line 61 of file statscounter-meat.hpp.

7.22.3.6 reset_array()

```
template<typename Array_Type , typename Data_Type >
void StatsCounter< Array_Type, Data_Type >::reset_array (
            const Array_Type * Array_ )  [inline]
```

Changes the reference array for the counting.

Parameters

*Array_*	A pointer to an array of class `Array_Type`.

Definition at line 14 of file statscounter-meat.hpp.

7.22.3.7 set_counters()

```
template<typename Array_Type , typename Data_Type >
void StatsCounter< Array_Type, Data_Type >::set_counters (
            Counters< Array_Type, Data_Type > * counters_ )  [inline]
```

Definition at line 46 of file statscounter-meat.hpp.

7.22.4 Member Data Documentation

7.22.4.1 Array

```
template<typename Array_Type = BArray<>, typename Data_Type = bool>
const Array_Type* StatsCounter< Array_Type, Data_Type >::Array
```

Definition at line 21 of file statscounter-bones.hpp.

7.22.4.2 counter_deleted

```
template<typename Array_Type = BArray<>, typename Data_Type = bool>
bool StatsCounter< Array_Type, Data_Type >::counter_deleted = false
```

Definition at line 27 of file statscounter-bones.hpp.

7.22.4.3 counters

```
template<typename Array_Type = BArray<>, typename Data_Type = bool>
Counters<Array_Type,Data_Type>* StatsCounter< Array_Type, Data_Type >::counters
```

Definition at line 26 of file statscounter-bones.hpp.

7.22.4.4 current_stats

```
template<typename Array_Type = BArray<>, typename Data_Type = bool>
std::vector< double > StatsCounter< Array_Type, Data_Type >::current_stats
```

Definition at line 23 of file statscounter-bones.hpp.

7.22.4.5 EmptyArray

```
template<typename Array_Type = BArray<>, typename Data_Type = bool>
Array_Type StatsCounter< Array_Type, Data_Type >::EmptyArray
```

Definition at line 22 of file statscounter-bones.hpp.

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

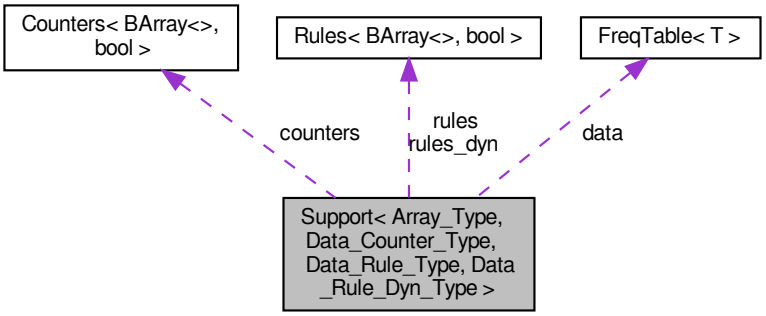
- include/barry/statscounter-bones.hpp
- include/barry/statscounter-meat.hpp

7.23 Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type > Class Template Reference

Compute the support of sufficient statistics.

`#include <support-bones.hpp>`

Collaboration diagram for Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >:



Public Member Functions

- Support (const Array_Type &Array_)

  Constructor passing a reference Array.
- Support (uint N_, uint M_)

  Constructor specifying the dimensions of the array (empty).
- Support ()
- ~Support ()
- void init_support (std::vector< Array_Type > *array_bank=nullptr, std::vector< std::vector< double > > *stats_bank=nullptr)
- void calc (std::vector< Array_Type > *array_bank=nullptr, std::vector< std::vector< double > > *stats_↩ bank=nullptr, unsigned int max_num_elements_=0u)

  Computes the entire support.
- Counts_type get_counts () const
- const MapVec_type * get_counts_ptr () const
- const std::vector< double > & get_current_stats () const

  List current statistics.
- void print () const

Resets the support calculator

If needed, the counters of a support object can be reused.

Parameters

Array↩ _	*New array over which the support will be computed.*

- void reset_array ()
- void reset_array (const Array_Type &Array_)

Manage counters

Parameters

f_	*A counter to be added.*
counters↩ _	*A vector of counters to be added.*

- void add_counter (Counter< Array_Type, Data_Counter_Type > *f_)
- void add_counter (Counter< Array_Type, Data_Counter_Type > f_)
- void set_counters (Counters< Array_Type, Data_Counter_Type > *counters_)

Manage rules

Parameters

f_	*A rule to be added.*
counters↩ _	*A vector of rules to be added.*

- void add_rule (Rule< Array_Type, Data_Rule_Type > *f_)
- void add_rule (Rule< Array_Type, Data_Rule_Type > f_)
- void set_rules (Rules< Array_Type, Data_Rule_Type > *rules_)
- void add_rule_dyn (Rule< Array_Type, Data_Rule_Dyn_Type > *f_)
- void add_rule_dyn (Rule< Array_Type, Data_Rule_Dyn_Type > f_)
- void set_rules_dyn (Rules< Array_Type, Data_Rule_Dyn_Type > *rules_)

Public Attributes

- Array_Type EmptyArray

  Reference array to generate the support.
- FreqTable data

  Table with the support.
- Counters< Array_Type, Data_Counter_Type > * counters

  Vector of couter functions.
- Rules< Array_Type, Data_Rule_Type > * rules

  Vector of static rules (cells to iterate).
- Rules< Array_Type, Data_Rule_Dyn_Type > * rules_dyn

  Vector of dynamic rules (to include/exclude a realizaton).
- uint N
- uint M
- bool delete_counters = true
- bool delete_rules = true
- bool delete_rules_dyn = true
- uint max_num_elements = BARRY_MAX_NUM_ELEMENTS
- std::vector< double > current_stats
- std::vector< std::pair< uint, uint > > coordinates_free
- std::vector< std::pair< uint, uint > > coordinates_locked
- std::vector< std::vector< double > > change_stats

7.23.1 Detailed Description

template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
class Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >

Compute the support of sufficient statistics.

Given an array and a set of counters, this object iterates throughout the support set of the Array while at the same time computing the support of the sufficient statitics.

The members `rule` and `rule_dyn` allow constraining the support. The first will establish which cells of the array will be used to iterate, for example, in the case of social networks, self-loops are not allowed, so the entire diagonal would be fixed to zero, reducing the size of the support.

In the case of `rule_dyn`, the function will stablish dynamically whether the current state will be included in the counts or not. For example, this set of rules can be used to constrain the support to networks that have a prescribed degree sequence.

Definition at line 35 of file support-bones.hpp.

7.23.2 Constructor & Destructor Documentation

7.23.2.1 Support() [1/3]

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::Support (
            const Array_Type & Array_ )  [inline]
```

Constructor passing a reference Array.

Definition at line 69 of file support-bones.hpp.

7.23.2.2 Support() [2/3]

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::Support (
            uint N_,
            uint M_ )  [inline]
```

Constructor specifying the dimensions of the array (empty).

Definition at line 78 of file support-bones.hpp.

7.23.2.3 Support() [3/3]

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::Support ( )
[inline]
```

Definition at line 85 of file support-bones.hpp.

7.23.2.4 ~Support()

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::~Support ( )
[inline]
```

Definition at line 92 of file support-bones.hpp.

7.23.3 Member Function Documentation

7.23.3.1 add_counter() [1/2]

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
void Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::add_counter
(
            Counter< Array_Type, Data_Counter_Type > * f_ )  [inline]
```

Definition at line 204 of file support-meat.hpp.

7.23.3.2 add_counter() [2/2]

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
void Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::add_counter
(
            Counter< Array_Type, Data_Counter_Type > f_ )  [inline]
```

Definition at line 214 of file support-meat.hpp.

7.23.3.3 add_rule() [1/2]

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
void Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::add_rule (
            Rule< Array_Type, Data_Rule_Type > * f_ )  [inline]
```

Definition at line 241 of file support-meat.hpp.

7.23.3.4 add_rule() [2/2]

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
void Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::add_rule (
            Rule< Array_Type, Data_Rule_Type > f_ )  [inline]
```

Definition at line 251 of file support-meat.hpp.

7.23.3.5 add_rule_dyn() [1/2]

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
void Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::add_rule_↩
dyn (
            Rule< Array_Type, Data_Rule_Dyn_Type > * f_ )  [inline]
```

Definition at line 276 of file support-meat.hpp.

7.23.3.6 add_rule_dyn() [2/2]

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
void Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::add_rule_↩
dyn (
            Rule< Array_Type, Data_Rule_Dyn_Type > f_ )  [inline]
```

Definition at line 286 of file support-meat.hpp.

7.23.3.7 calc()

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
void Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::calc (
            std::vector< Array_Type > * array_bank = nullptr,
            std::vector< std::vector< double > > * stats_bank = nullptr,
            unsigned int max_num_elements_ = 0u )  [inline]
```

Computes the entire support.

Not to be used by the user. Sets the starting point in the array (column-major).

Parameters

array_bank	If specified, the counter will add to the vector each possible state of the array, as it counts.
stats_bank	If specified, the counter will add to the vector each possible set of statistics, as it counts.

Definition at line 178 of file support-meat.hpp.

7.23.3.8 get_counts()

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
Counts_type Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >↩
::get_counts  [inline]
```

Definition at line 313 of file support-meat.hpp.

7.23.3.9 get_counts_ptr()

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
const MapVec_type * Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type
>::get_counts_ptr  [inline]
```

Definition at line 320 of file support-meat.hpp.

7.23.3.10 get_current_stats()

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
const std::vector< double > & Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_↩
Rule_Dyn_Type >::get_current_stats  [inline]
```

List current statistics.

Definition at line 327 of file support-meat.hpp.

7.23.3.11 init_support()

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
void Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::init_↩
support (
            std::vector< Array_Type > * array_bank = nullptr,
            std::vector< std::vector< double > > * stats_bank = nullptr )  [inline]
```

Definition at line 7 of file support-meat.hpp.

7.23.3.12 print()

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
void Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::print
[inline]
```

Definition at line 332 of file support-meat.hpp.

7.23.3.13 reset_array() [1/2]

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
void Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::reset_array
[inline]
```

Definition at line 73 of file support-meat.hpp.

7.23.3.14 reset_array() [2/2]

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
void Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::reset_array
(
            const Array_Type & Array_ )  [inline]
```

Definition at line 80 of file support-meat.hpp.

7.23.3.15 set_counters()

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
void Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::set_↩
counters (
            Counters< Array_Type, Data_Counter_Type > * counters_ )  [inline]
```

Definition at line 224 of file support-meat.hpp.

7.23.3.16 set_rules()

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
void Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::set_rules (
            Rules< Array_Type, Data_Rule_Type > * rules_ )  [inline]
```

Definition at line 261 of file support-meat.hpp.

7.23.3.17 set_rules_dyn()

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
void Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::set_rules↩
_dyn (
            Rules< Array_Type, Data_Rule_Dyn_Type > * rules_ )  [inline]
```

Definition at line 296 of file support-meat.hpp.

7.23.4 Member Data Documentation

7.23.4.1 change_stats

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
std::vector< std::vector< double > > Support< Array_Type, Data_Counter_Type, Data_Rule_Type,
Data_Rule_Dyn_Type >::change_stats
```

Definition at line 65 of file support-bones.hpp.

7.23.4.2 coordinates_free

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
std::vector< std::pair<uint,uint> > Support< Array_Type, Data_Counter_Type, Data_Rule_Type,
Data_Rule_Dyn_Type >::coordinates_free
```

Definition at line 63 of file support-bones.hpp.

7.23.4.3 coordinates_locked

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
std::vector< std::pair<uint,uint> > Support< Array_Type, Data_Counter_Type, Data_Rule_Type,
Data_Rule_Dyn_Type >::coordinates_locked
```

Definition at line 64 of file support-bones.hpp.

7.23.4.4 counters

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
Counters<Array_Type,Data_Counter_Type>* Support< Array_Type, Data_Counter_Type, Data_Rule_↩
Type, Data_Rule_Dyn_Type >::counters
```

Vector of couter functions.

Definition at line 51 of file support-bones.hpp.

7.23.4.5 current_stats

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
std::vector< double > Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn↩
_Type >::current_stats
```

Definition at line 62 of file support-bones.hpp.

7.23.4.6 data

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
FreqTable Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::data
```

Table with the support.

Definition at line 50 of file support-bones.hpp.

7.23.4.7 delete_counters

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
bool Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::delete_↩
counters = true
```

Definition at line 56 of file support-bones.hpp.

7.23.4.8 delete_rules

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
bool Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::delete_↩
rules = true
```

Definition at line 57 of file support-bones.hpp.

7.23.4.9 delete_rules_dyn

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
bool Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::delete_↩
rules_dyn = true
```

Definition at line 58 of file support-bones.hpp.

7.23.4.10 EmptyArray

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
Array_Type Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >↩
::EmptyArray
```

Reference array to generate the support.

Temp array used to iterate through the support.

Definition at line 49 of file support-bones.hpp.

7.23.4.11 M

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
uint Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::M
```

Definition at line 55 of file support-bones.hpp.

7.23.4.12 max_num_elements

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
uint Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::max_num_↩
elements = BARRY_MAX_NUM_ELEMENTS
```

Definition at line 59 of file support-bones.hpp.

7.23.4.13 N

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
uint Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::N
```

Definition at line 55 of file support-bones.hpp.

7.23.4.14 rules

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
Rules<Array_Type,Data_Rule_Type>* Support< Array_Type, Data_Counter_Type, Data_Rule_Type,
Data_Rule_Dyn_Type >::rules
```

Vector of static rules (cells to iterate).

Definition at line 52 of file support-bones.hpp.

7.23.4.15 rules_dyn

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
Rules<Array_Type,Data_Rule_Dyn_Type>* Support< Array_Type, Data_Counter_Type, Data_Rule_Type,
Data_Rule_Dyn_Type >::rules_dyn
```

Vector of dynamic rules (to include/exclude a realizaton).

Definition at line 53 of file support-bones.hpp.

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

- include/barry/support-bones.hpp
- include/barry/support-meat.hpp

7.24 vecHasher< T > Struct Template Reference

`#include <typedefs.hpp>`

Public Member Functions

- std::size_t operator() (std::vector< T > const &dat) const noexcept

7.24.1 Detailed Description

template<typename T>
struct vecHasher< T >

Definition at line 96 of file typedefs.hpp.

7.24.2 Member Function Documentation

7.24.2.1 operator()()

```
template<typename T >
std::size_t vecHasher< T >::operator() (
            std::vector< T > const & dat ) const  [inline], [noexcept]
```

Definition at line 97 of file typedefs.hpp.

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

- include/barry/typedefs.hpp

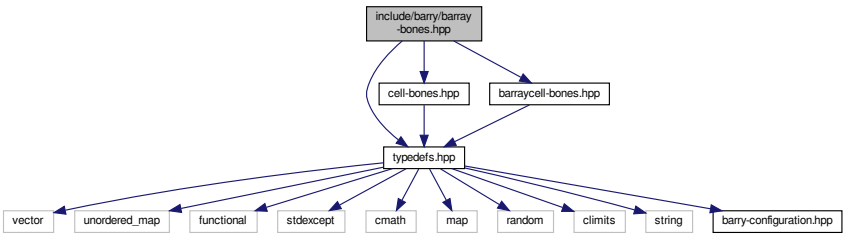
Chapter 8

File Documentation

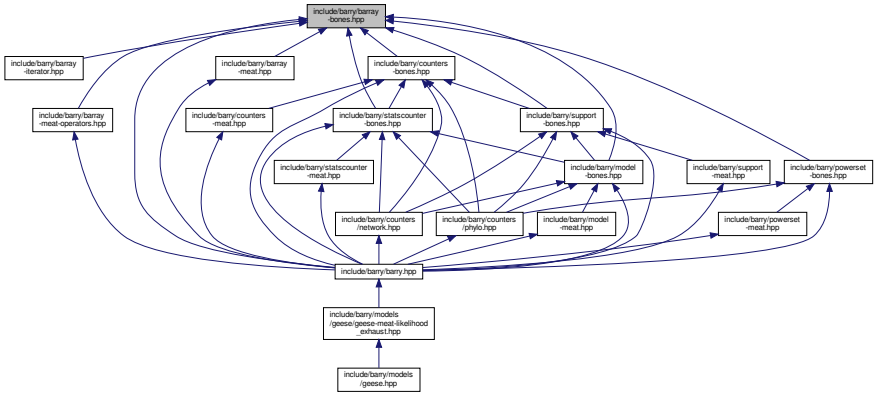
8.1 include/barry/barray-bones.hpp File Reference

```
#include "typedefs.hpp"
#include "cell-bones.hpp"
#include "barraycell-bones.hpp"
```

Include dependency graph for barray-bones.hpp:



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



Classes

- class BArray< Cell_Type, Data_Type >

  Baseline class for binary arrays.

Macros

- #define BARRAY_BONES_HPP 1

8.1.1 Macro Definition Documentation

8.1.1.1 BARRAY_BONES_HPP

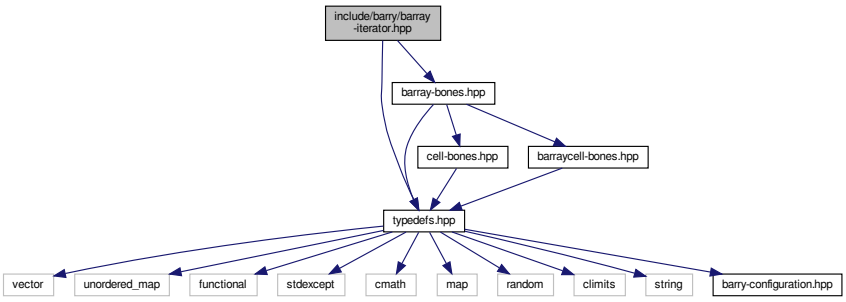
```
#define BARRAY_BONES_HPP 1
```

Definition at line 8 of file barray-bones.hpp.

8.2 include/barry/barray-iterator.hpp File Reference

```
#include "typedefs.hpp"
#include "barray-bones.hpp"
```

Include dependency graph for barray-iterator.hpp:



Classes

- class ConstBArrayRowIter< Cell_Type, Data_Type >

Macros

- #define BARRAY_ITERATOR_HPP 1

8.2.1 Macro Definition Documentation

8.2.1.1 BARRAY_ITERATOR_HPP

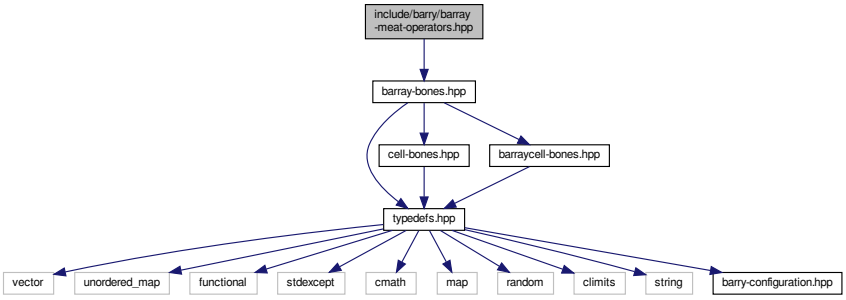
```
#define BARRAY_ITERATOR_HPP 1
```

Definition at line 7 of file barray-iterator.hpp.

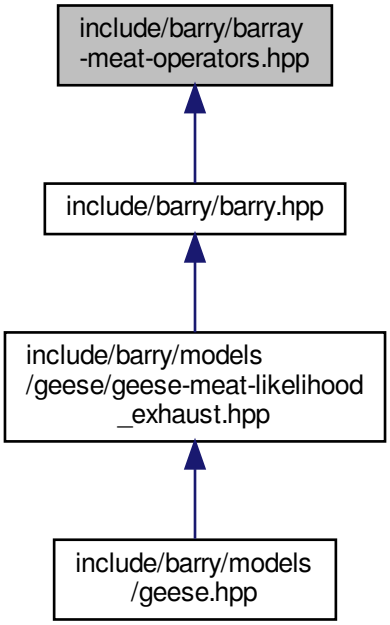
8.3 include/barry/barray-meat-operators.hpp File Reference

```
#include "barray-bones.hpp"
```

Include dependency graph for barray-meat-operators.hpp:



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



Macros

- #define ROW(a) this->el_ij[a]
- #define COL(a) this->el_ji[a]

Functions

- template<typename Cell_Type , typename Data_Type >
  void checkdim_ (const BArray< Cell_Type, Data_Type > &lhs, const BArray< Cell_Type, Data_Type > &rhs)

8.3.1 Macro Definition Documentation

8.3.1.1 COL

```
#define COL(
              a ) this->el_ji[a]
```

Definition at line 8 of file barray-meat-operators.hpp.

8.3.1.2 ROW

```
#define ROW(
              a ) this->el_ij[a]
```

Definition at line 7 of file barray-meat-operators.hpp.

8.3.2 Function Documentation

8.3.2.1 checkdim_()

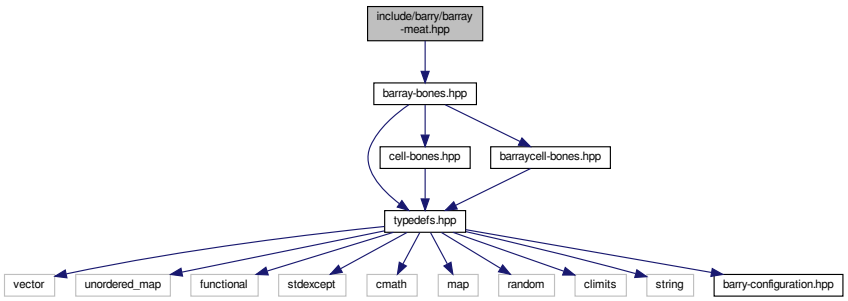
```
template<typename Cell_Type , typename Data_Type >
void checkdim_ (
            const BArray< Cell_Type, Data_Type > & lhs,
            const BArray< Cell_Type, Data_Type > & rhs )  [inline]
```

Definition at line 11 of file barray-meat-operators.hpp.

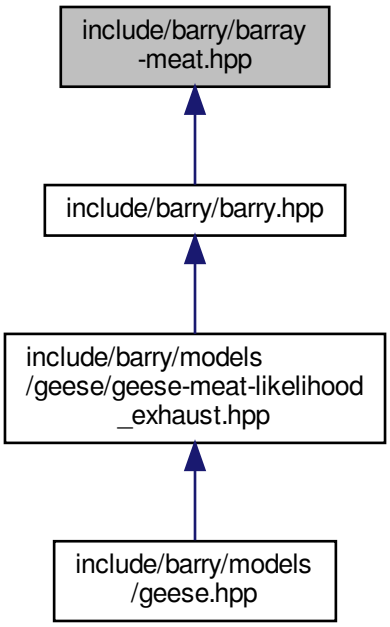
8.4 include/barry/barray-meat.hpp File Reference

```
#include "barray-bones.hpp"
```

Include dependency graph for barray-meat.hpp:



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



Macros

- #define ROW(a) this->el_ij[a]
- #define COL(a) this->el_ji[a]

8.4.1 Macro Definition Documentation

8.4.1.1 COL

```
#define COL(
              a ) this->el_ji[a]
```

Definition at line 8 of file barray-meat.hpp.

8.4.1.2 ROW

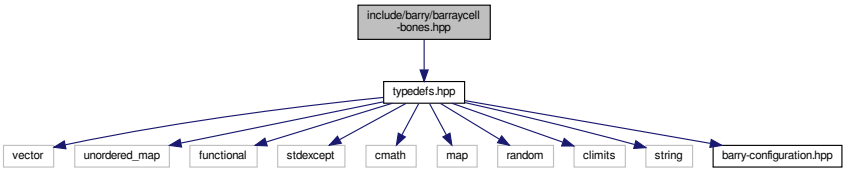
```
#define ROW(
              a ) this->el_ij[a]
```

Definition at line 7 of file barray-meat.hpp.

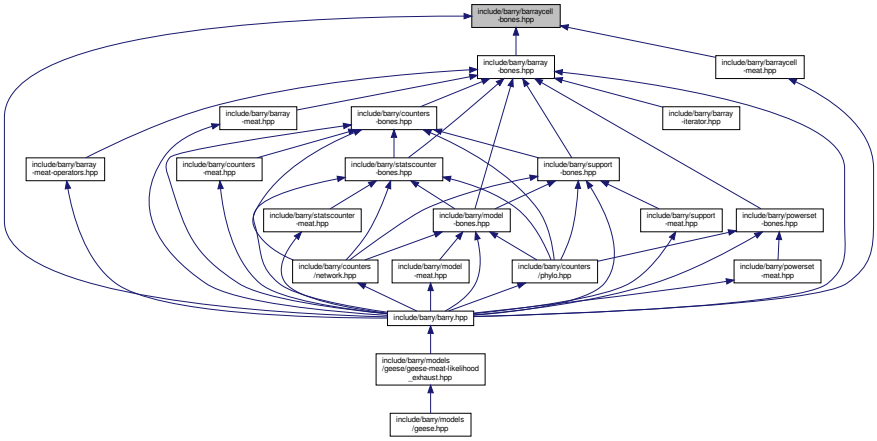
8.5 include/barry/barraycell-bones.hpp File Reference

```
#include "typedefs.hpp"
```

Include dependency graph for barraycell-bones.hpp:



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



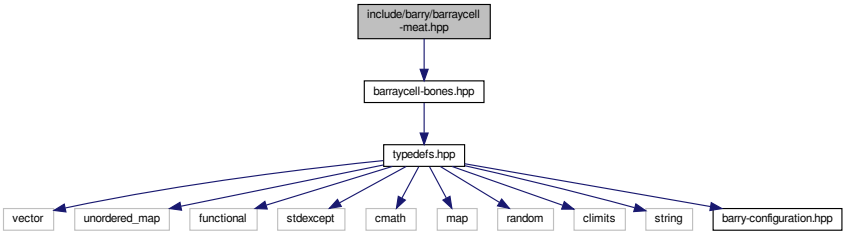
Classes

- class BArrayCell< Cell_Type, Data_Type >
- class BArrayCell_const< Cell_Type, Data_Type >

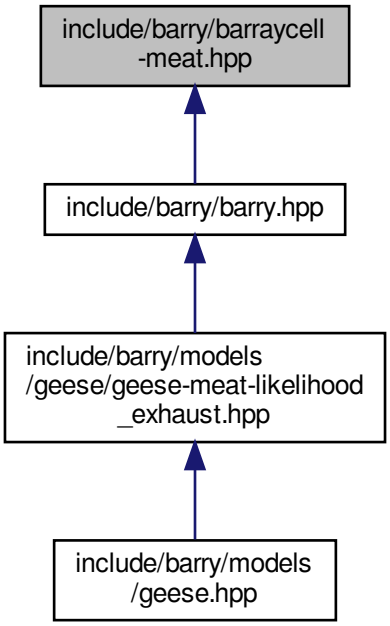
8.6 include/barry/barraycell-meat.hpp File Reference

```
#include "barraycell-bones.hpp"
```

Include dependency graph for barraycell-meat.hpp:

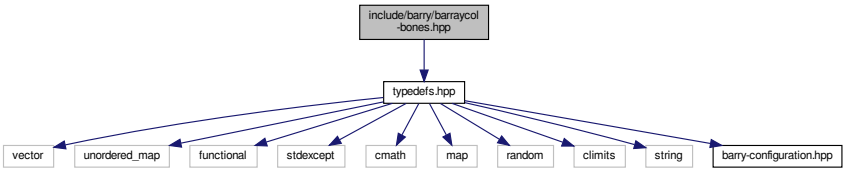


This graph shows which files directly or indirectly include this file:

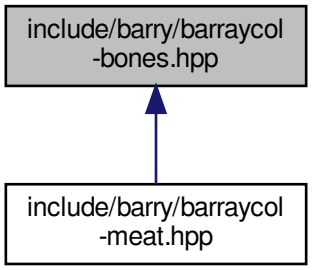


8.7 include/barry/barraycol-bones.hpp File Reference

`#include "typedefs.hpp"`
Include dependency graph for barraycol-bones.hpp:



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

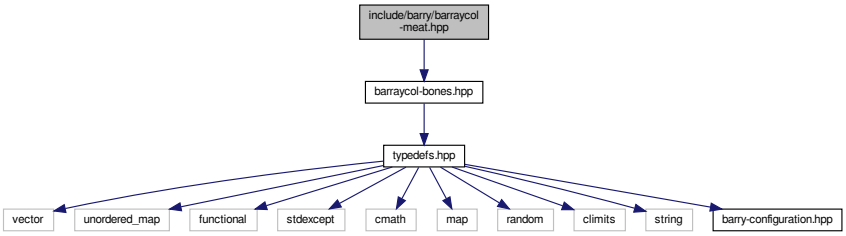


Classes

- class BArrayCol< Cell_Type, Data_Type >
- class BArrayCol_const< Cell_Type, Data_Type >

8.8 include/barry/barraycol-meat.hpp File Reference

`#include "barraycol-bones.hpp"`
Include dependency graph for barraycol-meat.hpp:



Macros

- #define BARRY_BARRAYCOL_MEAT_HPP 1

8.8.1 Macro Definition Documentation

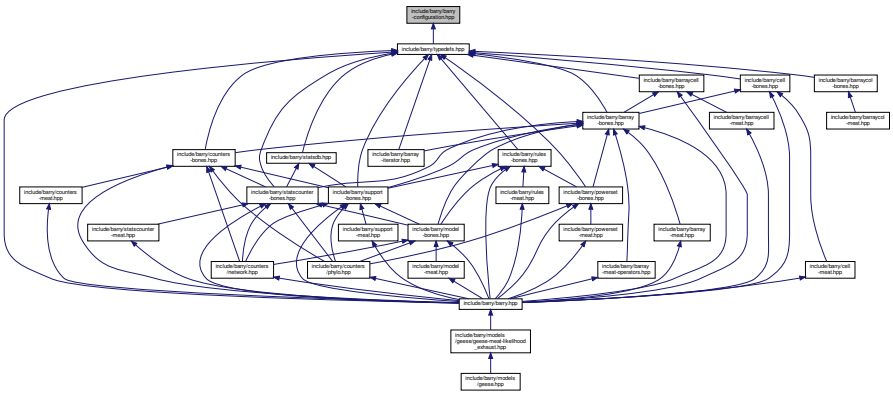
8.8.1.1 BARRY_BARRAYCOL_MEAT_HPP

```
#define BARRY_BARRAYCOL_MEAT_HPP 1
```

Definition at line 4 of file barraycol-meat.hpp.

8.9 include/barry/barry-configuration.hpp File Reference

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



Macros

- #define BARRY_MAX_NUM_ELEMENTS static_cast< unsigned int >(UINT_MAX/2u)

Configuration MACROS

These are mostly related to performance. The definitions follow:

- `BARRY_USE_UNORDERED_MAP` If specified, then barry is compiled using `std::unordered_map`. Otherwise it will use `std::map` for the arrays.
- `BARRY_USE_SAFE_EXP` When specified, it will multiply all likelihoods in `Model` by (1/-100)/(1/-100) so that numerical overflows are avoided.
- `BARRY_CHECK_FINITE` When specified, it will introduce a macro

- #define BARRY_SAFE_EXP -100.0
- #define BARRY_ISFINITE(a)
- #define BARRY_CHECK_SUPPORT(x, maxs)
- template<typename Ta , typename Tb >
  using Map = std::map< Ta, Tb >

8.9.1 Macro Definition Documentation

8.9.1.1 BARRY_CHECK_SUPPORT

```
#define BARRY_CHECK_SUPPORT(
              x,
              maxs )
```

Definition at line 46 of file barry-configuration.hpp.

8.9.1.2 BARRY_ISFINITE

```
#define BARRY_ISFINITE(
              a )
```

Definition at line 39 of file barry-configuration.hpp.

8.9.1.3 BARRY_MAX_NUM_ELEMENTS

```
#define BARRY_MAX_NUM_ELEMENTS static_cast< unsigned int >(UINT_MAX/2u)
```

Definition at line 5 of file barry-configuration.hpp.

8.9.1.4 BARRY_SAFE_EXP

```
#define BARRY_SAFE_EXP -100.0
```

Definition at line 32 of file barry-configuration.hpp.

8.9.2 Typedef Documentation

8.9.2.1 Map

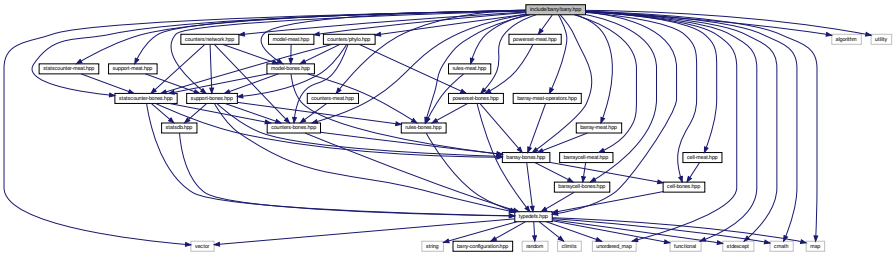
```
template<typename Ta , typename Tb >
using Map = std::map<Ta,Tb>
```

Definition at line 26 of file barry-configuration.hpp.

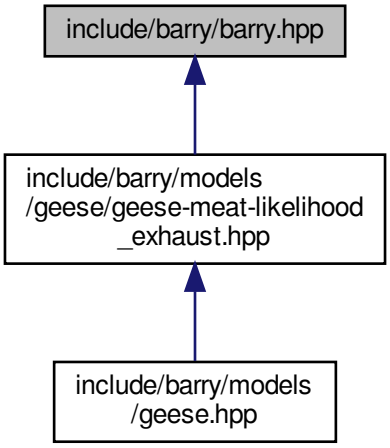
8.10 include/barry/barry.hpp File Reference

```
#include <vector>
#include <unordered_map>
#include <functional>
#include <stdexcept>
#include <cmath>
#include <map>
#include <algorithm>
#include <utility>
#include "typedefs.hpp"
#include "cell-bones.hpp"
#include "cell-meat.hpp"
#include "barray-bones.hpp"
#include "barraycell-bones.hpp"
#include "barray-meat.hpp"
#include "barraycell-meat.hpp"
#include "barray-meat-operators.hpp"
#include "counters-bones.hpp"
#include "counters-meat.hpp"
#include "statscounter-bones.hpp"
#include "statscounter-meat.hpp"
#include "support-bones.hpp"
#include "support-meat.hpp"
#include "powerset-bones.hpp"
#include "powerset-meat.hpp"
#include "model-bones.hpp"
#include "model-meat.hpp"
#include "rules-bones.hpp"
#include "rules-meat.hpp"
#include "counters/network.hpp"
#include "counters/phylo.hpp"
```

Include dependency graph for barry.hpp:



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



Namespaces

- barry

  barry: Your go-to motif accountant
- barry::counters

  Tree class and TreeIterator class.
- barry::counters::network
- barry::counters::phylo

Macros

- #define COUNTER_FUNCTION(a)
- #define COUNTER_LAMBDA(a)
- #define RULE_FUNCTION(a)
- #define RULE_LAMBDA(a)

8.10.1 Macro Definition Documentation

8.10.1.1 COUNTER_FUNCTION

```
#define COUNTER_FUNCTION(
              a )
```

Value:

```
    template <typename Array_Type = barry::BArray<>, typename Data_Type = bool> \
    inline double (a) (const Array_Type & Array, uint i, uint j, Data_Type * data)\
```

Definition at line 64 of file barry.hpp.

8.10.1.2 COUNTER_LAMBDA

```
#define COUNTER_LAMBDA(
              a )
```

Value:

```
template <typename Array_Type = barry::BArray<>, typename Data_Type = bool> \
Counter_fun_type<Array_Type, Data_Type> a = \
[](const Array_Type & Array, uint i, uint j, Data_Type * data)
```

Definition at line 67 of file barry.hpp.

8.10.1.3 RULE_FUNCTION

```
#define RULE_FUNCTION(
              a )
```

Value:

```
template <typename Array_Type = barry::BArray<>, typename Data_Type = bool> \
inline bool (a) (const Array_Type & Array, uint i, uint j, Data_Type * data)\
```

Definition at line 71 of file barry.hpp.

8.10.1.4 RULE_LAMBDA

```
#define RULE_LAMBDA(
              a )
```

Value:

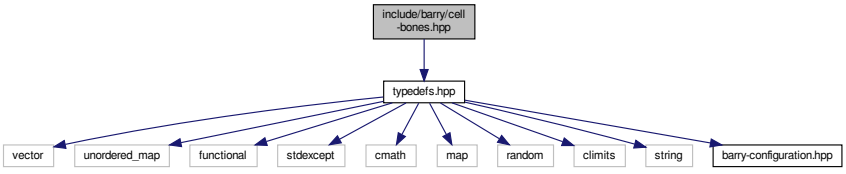
```
template <typename Array_Type = barry::BArray<>, typename Data_Type = bool> \
Rule_fun_type<Array_Type, Data_Type> a = \
[](const Array_Type & Array, uint i, uint j, Data_Type * data)
```

Definition at line 74 of file barry.hpp.

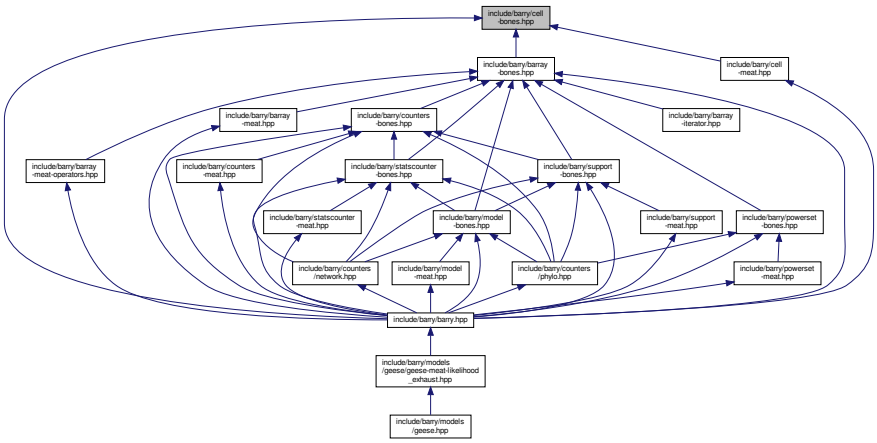
8.11 include/barry/cell-bones.hpp File Reference

```
#include "typedefs.hpp"
```

Include dependency graph for cell-bones.hpp:



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



Classes

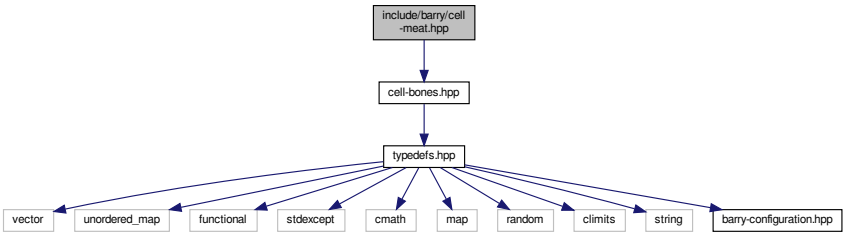
- class Cell< Cell_Type >

  Entries in *BArray*. *For now, it only has two members:*

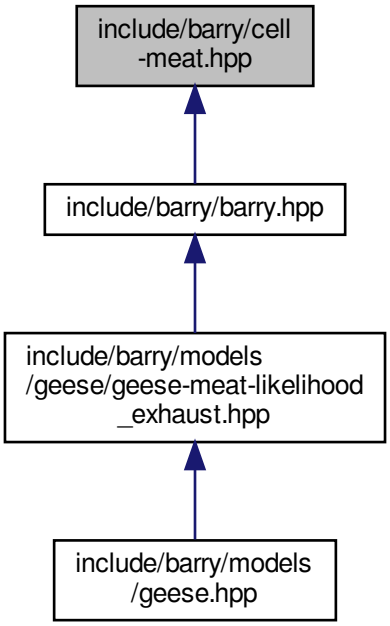
8.12 include/barry/cell-meat.hpp File Reference

```
#include "cell-bones.hpp"
```

Include dependency graph for cell-meat.hpp:



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

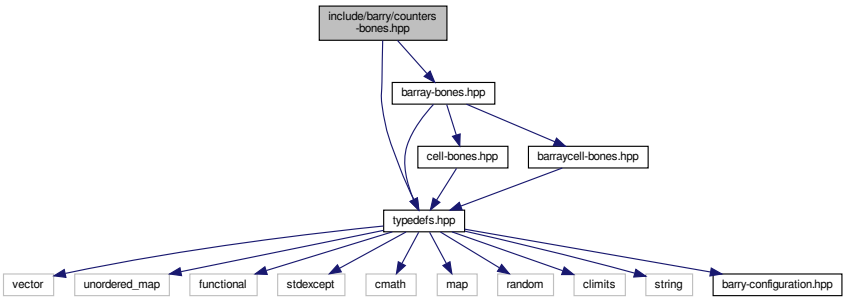


8.13 include/barry/col-bones.hpp File Reference

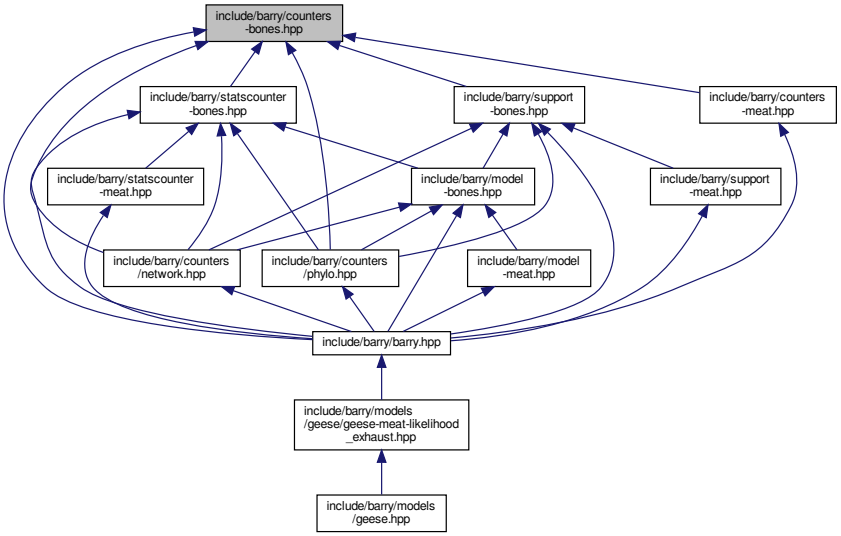
8.14 include/barry/counters-bones.hpp File Reference

```
#include "typedefs.hpp"
#include "barray-bones.hpp"
```

Include dependency graph for counters-bones.hpp:



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



Classes

- class Counter< Array_Type, Data_Type >

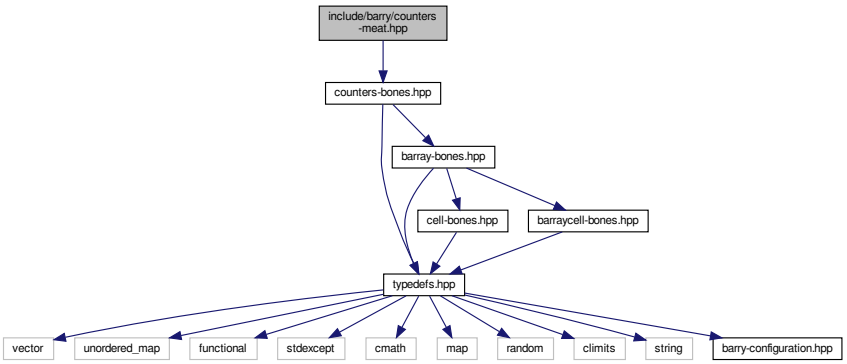
  A counter function based on change statistics.
- class Counters< Array_Type, Data_Type >

  Vector of counters.

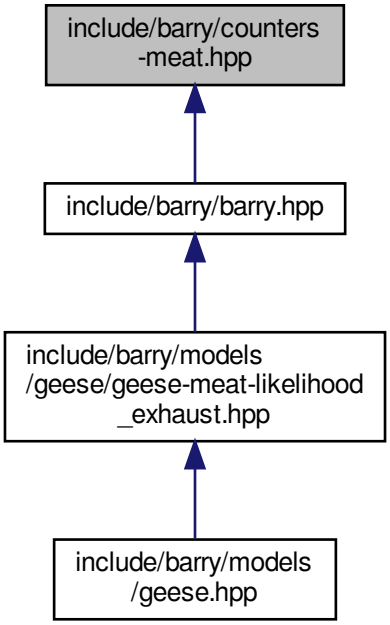
8.15 include/barry/counters-meat.hpp File Reference

```
#include "counters-bones.hpp"
```

Include dependency graph for counters-meat.hpp:



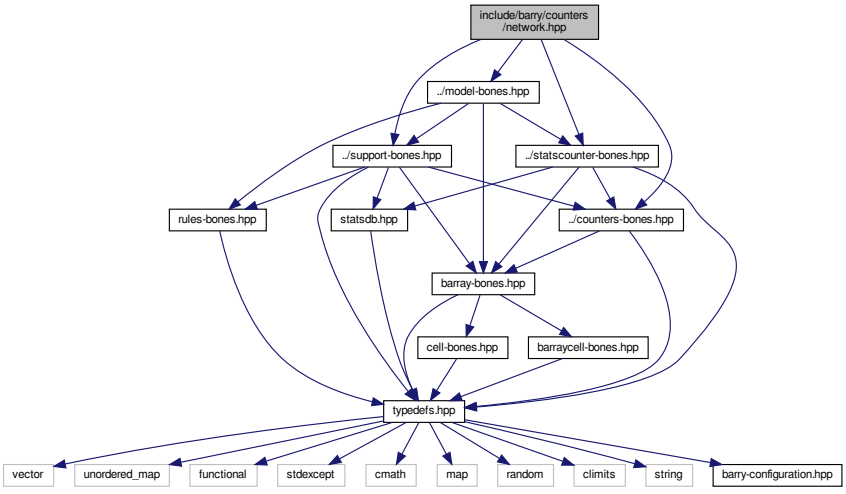
This graph shows which files directly or indirectly include this file:



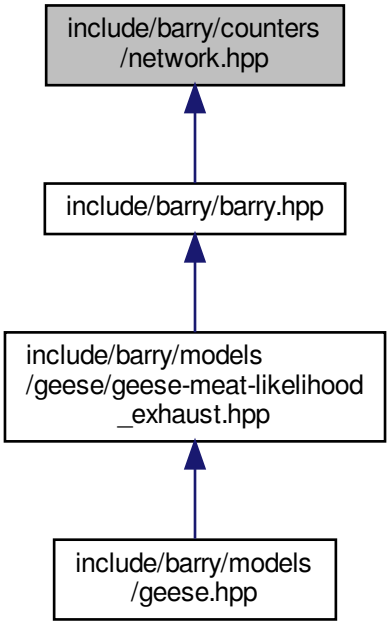
8.16 include/barry/counters/network.hpp File Reference

```
#include "../counters-bones.hpp"
#include "../support-bones.hpp"
#include "../statscounter-bones.hpp"
#include "../model-bones.hpp"
```

Include dependency graph for network.hpp:



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



Classes

- class NetworkData

  Data class for Networks.
- class NetCounterData

  Data class used to store arbitrary uint or double vectors.

Macros

- #define NET_C_DATA_IDX(i) (data->indices[i])
- #define NET_C_DATA_NUM(i) (data->numbers[i])

Macros for defining counters

- #define NETWORK_COUNTER(a)
- #define NETWORK_COUNTER_LAMBDA(a)

Macros for defining rules

- #define NETWORK_RULE(a)
- #define NETWORK_RULE_LAMBDA(a)

Typedefs

Convenient typedefs for network objects.

- typedef BArray< double, NetworkData > Network
- typedef Counter< Network, NetCounterData > NetCounter
- typedef Counters< Network, NetCounterData > NetCounters
- typedef Support< Network, NetCounterData > NetSupport
- typedef StatsCounter< Network, NetCounterData > NetStatsCounter
- typedef Model< Network, NetCounterData > NetModel
- typedef Rule< Network, bool > NetRule
- typedef Rules< Network, bool > NetRules

Functions

- void counter_edges (NetCounters *counters)

  Number of edges.
- void counter_isolates (NetCounters *counters)

  Number of isolated vertices.
- void counter_mutual (NetCounters *counters)

  Number of mutual ties.
- void counter_istar2 (NetCounters *counters)
- void counter_ostar2 (NetCounters *counters)
- void counter_ttriads (NetCounters *counters)
- void counter_ctriads (NetCounters *counters)
- void counter_density (NetCounters *counters)
- void counter_idegree15 (NetCounters *counters)
- void counter_odegree15 (NetCounters *counters)
- void counter_absdiff (NetCounters *counters, uint attr_id, double alpha=1.0)

  Sum of absolute attribute difference between ego and alter.
- void counter_diff (NetCounters *counters, uint attr_id, double alpha=1.0, double tail_head=true)

  Sum of attribute difference between ego and alter to pow(alpha)
- NETWORK_COUNTER (init_single_attr)
- void counter_nodeicov (NetCounters *counters, uint attr_id)
- void counter_nodeocov (NetCounters *counters, uint attr_id)
- void counter_nodecov (NetCounters *counters, uint attr_id)
- void counter_nodematch (NetCounters *counters, uint attr_id)
- void counter_idegree (NetCounters *counters, std::vector< uint > d)

  Counts number of vertices with a given in-degree.
- void counter_odegree (NetCounters *counters, std::vector< uint > d)

  Counts number of vertices with a given out-degree.
- void counter_degree (NetCounters *counters, std::vector< uint > d)

  Counts number of vertices with a given out-degree.

Rules for network models

Parameters

rules	*A pointer to a* `NetRules` *object (*`Rules<Network, bool>`*).*

- void rules_zerodiag (NetRules *rules)

  Number of edges.

8.16.1 Macro Definition Documentation

8.16.1.1 NET_C_DATA_IDX

```
#define NET_C_DATA_IDX(
              i ) (data->indices[i])
```

Definition at line 79 of file network.hpp.

8.16.1.2 NET_C_DATA_NUM

```
#define NET_C_DATA_NUM(
              i ) (data->numbers[i])
```

Definition at line 80 of file network.hpp.

8.16.1.3 NETWORK_COUNTER

```
#define NETWORK_COUNTER(
              a )
```

Value:

```
inline double (a) \
(const Network & Array, uint i, uint j, NetCounterData * data)
```

Function for definition of a network counter function

Definition at line 101 of file network.hpp.

8.16.1.4 NETWORK_COUNTER_LAMBDA

```
#define NETWORK_COUNTER_LAMBDA(
              a )
```

Value:

```
    Counter_fun_type<Network, NetCounterData> a = \
    [](const Network & Array, uint i, uint j, NetCounterData * data)
```

Lambda function for definition of a network counter function

Definition at line 104 of file network.hpp.

8.16.1.5 NETWORK_RULE

```
#define NETWORK_RULE(
                a )
```

Value:
```
inline bool (a) \
(const Network & Array, uint i, uint j, bool * data)
```

Function for definition of a network counter function

Definition at line 113 of file network.hpp.

8.16.1.6 NETWORK_RULE_LAMBDA

```
#define NETWORK_RULE_LAMBDA(
                a )
```

Value:
```
Rule_fun_type<Network, bool> a = \
[](const Network & Array, uint i, uint j, bool * data)
```

Lambda function for definition of a network counter function

Definition at line 116 of file network.hpp.

8.16.2 Typedef Documentation

8.16.2.1 NetCounter

```
typedef Counter<Network, NetCounterData > NetCounter
```

Definition at line 88 of file network.hpp.

8.16.2.2 NetCounters

```
typedef Counters< Network, NetCounterData> NetCounters
```

Definition at line 89 of file network.hpp.

8.16.2.3 NetModel

```
typedef Model<Network, NetCounterData> NetModel
```

Definition at line 92 of file network.hpp.

8.16.2.4 NetRule

```
typedef Rule<Network,bool> NetRule
```

Definition at line 93 of file network.hpp.

8.16.2.5 NetRules

```
typedef Rules<Network,bool> NetRules
```

Definition at line 94 of file network.hpp.

8.16.2.6 NetStatsCounter

```
typedef StatsCounter<Network, NetCounterData> NetStatsCounter
```

Definition at line 91 of file network.hpp.

8.16.2.7 NetSupport

```
typedef Support<Network, NetCounterData > NetSupport
```

Definition at line 90 of file network.hpp.

8.16.2.8 Network

```
typedef BArray<double, NetworkData> Network
```

Definition at line 87 of file network.hpp.

8.16.3 Function Documentation

8.16.3.1 rules_zerodiag()

```
void rules_zerodiag (
            NetRules * rules )  [inline]
```

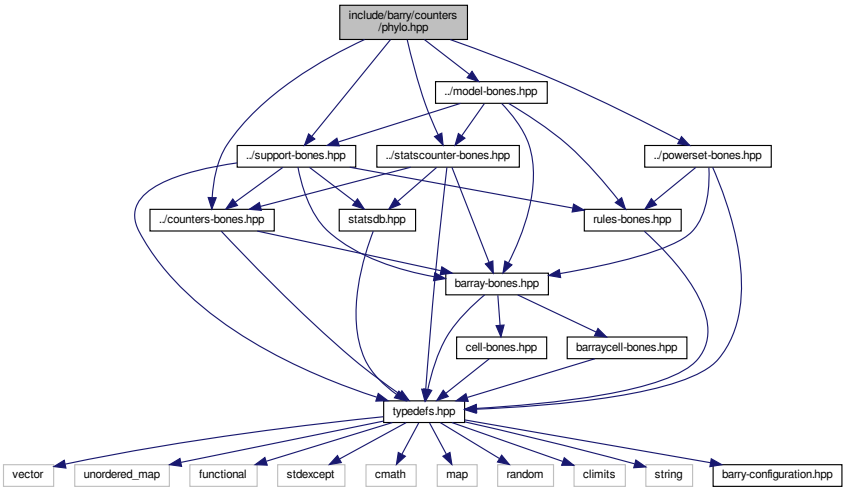
Number of edges.

Definition at line 742 of file network.hpp.

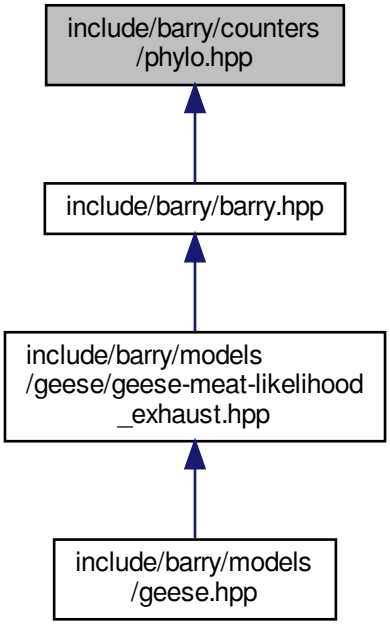
8.17 include/barry/counters/phylo.hpp File Reference

```
#include "../counters-bones.hpp"
#include "../support-bones.hpp"
#include "../statscounter-bones.hpp"
#include "../model-bones.hpp"
#include "../powerset-bones.hpp"
```

Include dependency graph for phylo.hpp:



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



Classes

- class NodeData

  Data definition for the `PhyloArray` *class.*

Macros

- #define PHYLO_COUNTER(a)

  Extension of a simple counter.
- #define PHYLO_COUNTER_LAMBDA(a)
- #define PHYLO_CHECK_MISSING()

Typedefs

Convenient typedefs for Node objects.

- typedef BArray< uint, NodeData > PhyloArray
- typedef Counter< PhyloArray, PhyloCounterData > PhyloCounter
- typedef Counters< PhyloArray, PhyloCounterData > PhyloCounters
- typedef Rule< PhyloArray, PhyloRuleData > PhyloRule
- typedef Rules< PhyloArray, PhyloRuleData > PhyloRules
- typedef Support< PhyloArray, PhyloCounterData, PhyloRuleData, uint > PhyloSupport
- typedef StatsCounter< PhyloArray, PhyloCounterData > PhyloStatsCounter
- typedef Model< PhyloArray, PhyloCounterData, PhyloRuleData, uint > PhyloModel
- typedef PowerSet< PhyloArray, PhyloRuleData > PhyloPowerSet

Functions

- std::string get_last_name (bool d)
- void counter_overall_gains (PhyloCounters *counters, bool duplication=true)

  Overall functional gains.
- void counter_gains (PhyloCounters *counters, std::vector< uint > nfun, bool duplication=true)

  Functional gains for a specific function (`nfun`).
- void counter_gains_k_offspring (PhyloCounters *counters, std::vector< uint > nfun, uint k=1u, bool duplication=true)

  k genes gain function nfun
- void counter_genes_changing (PhyloCounters *counters, bool duplication=true)

  Keeps track of how many genes are changing (either 0, 1, or 2 if dealing with regular trees.)
- void counter_overall_loss (PhyloCounters *counters, bool duplication=true)

  Overall functional loss.
- void counter_maxfuns (PhyloCounters *counters, uint lb, uint ub, bool duplication=true)

  Cap the number of functions per gene.
- void counter_loss (PhyloCounters *counters, std::vector< uint > nfun, bool duplication=true)

  Total count of losses for an specific function.
- void counter_overall_changes (PhyloCounters *counters, bool duplication=true)

  Total number of changes. Use this statistic to account for "preservation".
- void counter_subfun (PhyloCounters *counters, uint nfunA, uint nfunB, bool duplication=true)

  Total count of Sub-functionalization events.
- void counter_cogain (PhyloCounters *counters, uint nfunA, uint nfunB, bool duplication=true)

  Co-evolution (joint gain or loss)
- void counter_longest (PhyloCounters *counters)

  Longest branch mutates (either by gain or by loss)
- void counter_neofun (PhyloCounters *counters, uint nfunA, uint nfunB, bool duplication=true)

  Total number of neofunctionalization events.
- void counter_neofun_a2b (PhyloCounters *counters, uint nfunA, uint nfunB, bool duplication=true)

  Total number of neofunctionalization events.
- void counter_co_opt (PhyloCounters *counters, uint nfunA, uint nfunB, bool duplication=true)

  Function co-opting.

- #define PHYLO_C_DATA_IDX(i) (data.operator[](i))
- typedef std::vector< uint > PhyloCounterData
- typedef std::vector< std::pair< uint, uint > > PhyloRuleData

8.17.1 Macro Definition Documentation

8.17.1.1 PHYLO_C_DATA_IDX

```
#define PHYLO_C_DATA_IDX(
              i ) (data.operator[](i))
```

Definition at line 56 of file phylo.hpp.

8.17.1.2 PHYLO_CHECK_MISSING

```
#define PHYLO_CHECK_MISSING( )
```

Value:

```
    if (Array.D() == nullptr) \
    throw std::logic_error("The array data is nullptr."); \
    if (data == nullptr) \
    throw std::logic_error("The counter data is nullptr.")
```

Definition at line 91 of file phylo.hpp.

8.17.1.3 PHYLO_COUNTER

```
#define PHYLO_COUNTER(
                a )
```

Value:

```
    inline double (a) (const PhyloArray & Array, uint i, \
    uint j, PhyloCounterData * data)
```

Extension of a simple counter.

It allows specifying extra arguments, in particular, the corresponding sets of rows to which this statistic may be relevant. This could be important in the case of, for example, counting correlation type statistics between function 1 and 2, and between function 1 and 3.

Definition at line 85 of file phylo.hpp.

8.17.1.4 PHYLO_COUNTER_LAMBDA

```
#define PHYLO_COUNTER_LAMBDA(
                a )
```

Value:

```
    Counter_fun_type<PhyloArray, PhyloCounterData> a = \
    [](const PhyloArray & Array, uint i, uint j, PhyloCounterData * data)
```

Definition at line 88 of file phylo.hpp.

8.17.2 Typedef Documentation

8.17.2.1 PhyloArray

```
typedef BArray<uint, NodeData> PhyloArray
```

Definition at line 63 of file phylo.hpp.

8.17.2.2 PhyloCounter

```
typedef Counter<PhyloArray, PhyloCounterData > PhyloCounter
```

Definition at line 64 of file phylo.hpp.

8.17.2.3 PhyloCounterData

```
typedef std::vector< uint > PhyloCounterData
```

Definition at line 53 of file phylo.hpp.

8.17.2.4 PhyloCounters

```
typedef Counters< PhyloArray, PhyloCounterData> PhyloCounters
```

Definition at line 65 of file phylo.hpp.

8.17.2.5 PhyloModel

```
typedef Model<PhyloArray, PhyloCounterData, PhyloRuleData, uint > PhyloModel
```

Definition at line 70 of file phylo.hpp.

8.17.2.6 PhyloPowerSet

```
typedef PowerSet<PhyloArray, PhyloRuleData> PhyloPowerSet
```

Definition at line 71 of file phylo.hpp.

8.17.2.7 PhyloRule

```
typedef Rule<PhyloArray,PhyloRuleData> PhyloRule
```

Definition at line 66 of file phylo.hpp.

8.17.2.8 PhyloRuleData

```
typedef std::vector< std::pair< uint, uint > > PhyloRuleData
```

Definition at line 54 of file phylo.hpp.

8.17.2.9 PhyloRules

```
typedef Rules<PhyloArray,PhyloRuleData> PhyloRules
```

Definition at line 67 of file phylo.hpp.

8.17.2.10 PhyloStatsCounter

```
typedef StatsCounter<PhyloArray, PhyloCounterData> PhyloStatsCounter
```

Definition at line 69 of file phylo.hpp.

8.17.2.11 PhyloSupport

```
typedef Support<PhyloArray, PhyloCounterData, PhyloRuleData, uint > PhyloSupport
```

Definition at line 68 of file phylo.hpp.

8.17.3 Function Documentation

8.17.3.1 get_last_name()

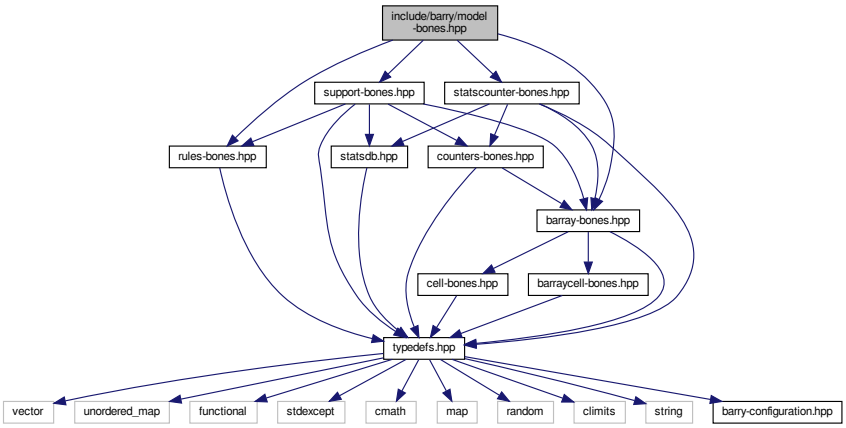
```
std::string get_last_name (
            bool d )  [inline]
```

Definition at line 96 of file phylo.hpp.

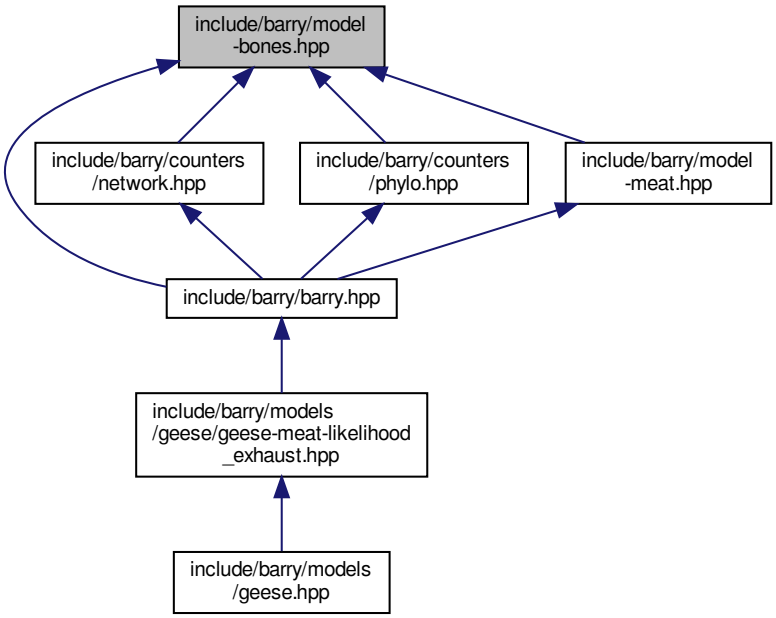
8.18 include/barry/model-bones.hpp File Reference

```
#include "barray-bones.hpp"
#include "support-bones.hpp"
#include "statscounter-bones.hpp"
#include "rules-bones.hpp"
```

Include dependency graph for model-bones.hpp:



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



Classes

- class Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >

General framework for discrete exponential models. This class allows generating discrete exponential models in the form of a linear exponential model:

Functions

- double update_normalizing_constant (const std::vector< double > &params, const Counts_type &support)
- double likelihood_ (const std::vector< double > &target_stats, const std::vector< double > &params, const double normalizing_constant, bool log_=false)
- template<typename Array_Type >
  std::vector< double > keygen_default (const Array_Type &Array_)

  Array Hasher class (used for computing support)

8.18.1 Function Documentation

8.18.1.1 keygen_default()

```
template<typename Array_Type >
std::vector< double > keygen_default (
            const Array_Type & Array_ )  [inline]
```

Array Hasher class (used for computing support)

Definition at line 69 of file model-bones.hpp.

8.18.1.2 likelihood_()

```
double likelihood_ (
            const std::vector< double > & target_stats,
            const std::vector< double > & params,
            const double normalizing_constant,
            bool log_ = false )  [inline]
```

Definition at line 40 of file model-bones.hpp.

8.18.1.3 update_normalizing_constant()

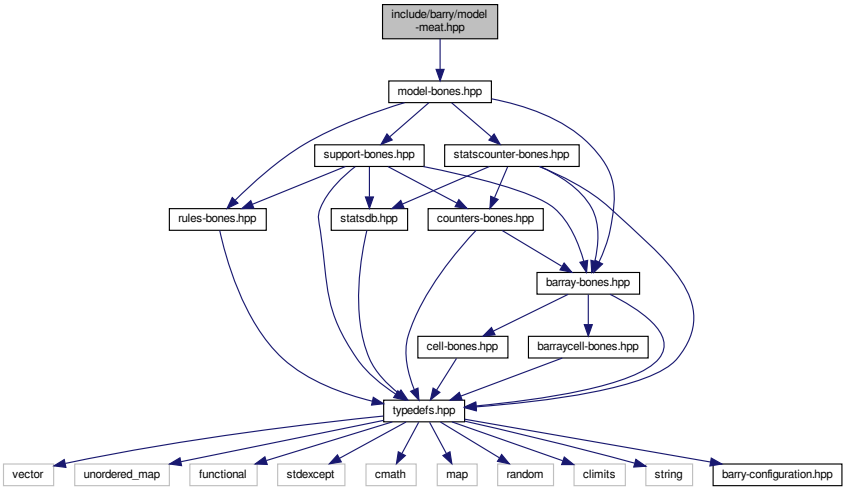
```
double update_normalizing_constant (
            const std::vector< double > & params,
            const Counts_type & support )  [inline]
```

Definition at line 16 of file model-bones.hpp.

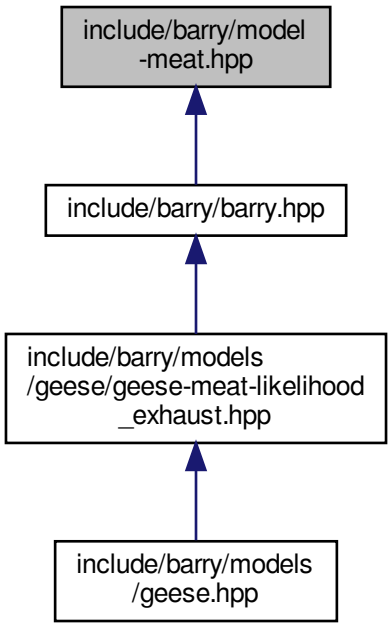
8.19 include/barry/model-meat.hpp File Reference

```
#include "model-bones.hpp"
```

Include dependency graph for model-meat.hpp:



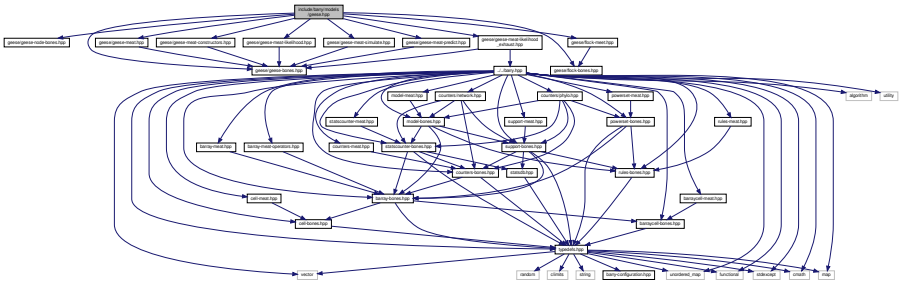
This graph shows which files directly or indirectly include this file:



8.20 include/barry/models/geese.hpp File Reference

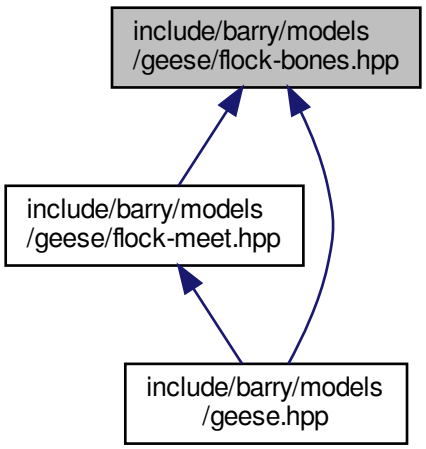
```
#include "geese/geese-node-bones.hpp"
#include "geese/geese-bones.hpp"
#include "geese/geese-meat.hpp"
#include "geese/geese-meat-constructors.hpp"
#include "geese/geese-meat-likelihood.hpp"
#include "geese/geese-meat-likelihood_exhaust.hpp"
#include "geese/geese-meat-simulate.hpp"
#include "geese/geese-meat-predict.hpp"
#include "geese/flock-bones.hpp"
#include "geese/flock-meet.hpp"
```

Include dependency graph for geese.hpp:



8.21 include/barry/models/geese/flock-bones.hpp File Reference

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



Classes

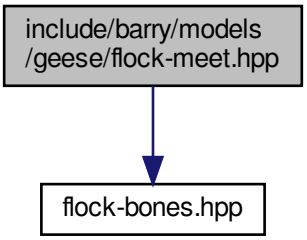
- class Flock

  A Flock is a group of Geese.

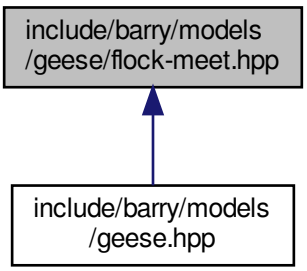
8.22 include/barry/models/geese/flock-meet.hpp File Reference

```
#include "flock-bones.hpp"
```

Include dependency graph for flock-meet.hpp:

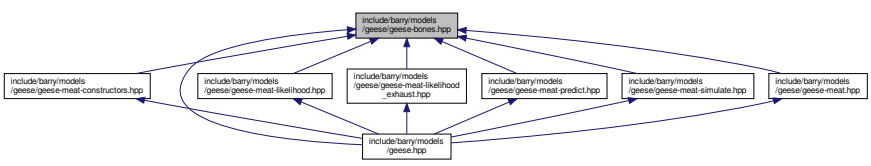


This graph shows which files directly or indirectly include this file:



8.23 include/barry/models/geese/geese-bones.hpp File Reference

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



Classes

- class Geese

  Annotated Phylo Model.

Macros

- #define INITIALIZED()

Functions

- template<typename Ta , typename Tb >
  std::vector< Ta > vector_caster (const std::vector< Tb > &x)
- RULE_FUNCTION (rule_empty_free)
- std::vector< double > keygen_full (const phylocounters::PhyloArray &array)
- bool vec_diff (const std::vector< unsigned int > &s, const std::vector< unsigned int > &a)

8.23.1 Macro Definition Documentation

8.23.1.1 INITIALIZED

```
#define INITIALIZED( )
```

Value:

```
    if (!this->initialized) \
    throw std::logic_error("The model has not been initialized yet.");
```

Definition at line 18 of file geese-bones.hpp.

8.23.2 Function Documentation

8.23.2.1 keygen_full()

```
std::vector< double > keygen_full (
            const phylocounters::PhyloArray & array )  [inline]
```

Definition at line 31 of file geese-bones.hpp.

8.23.2.2 RULE_FUNCTION()

```
RULE_FUNCTION (
             rule_empty_free  )
```

Definition at line 22 of file geese-bones.hpp.

8.23.2.3 vec_diff()

```
bool vec_diff (
             const std::vector< unsigned int > & s,
             const std::vector< unsigned int > & a )  [inline]
```

Definition at line 51 of file geese-bones.hpp.

8.23.2.4 vector_caster()

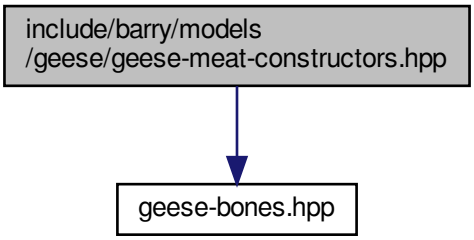
```
template<typename Ta , typename Tb >
std::vector< Ta > vector_caster (
             const std::vector< Tb > & x )  [inline]
```

Definition at line 10 of file geese-bones.hpp.

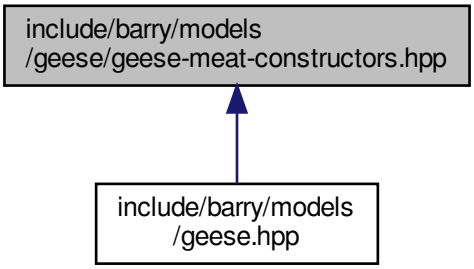
8.24 include/barry/models/geese/geese-meat-constructors.hpp File Reference

```
#include "geese-bones.hpp"
```

Include dependency graph for geese-meat-constructors.hpp:



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



Macros

- #define GEESE_MEAT_CONSTRUCTORS_HPP 1

8.24.1 Macro Definition Documentation

8.24.1.1 GEESE_MEAT_CONSTRUCTORS_HPP

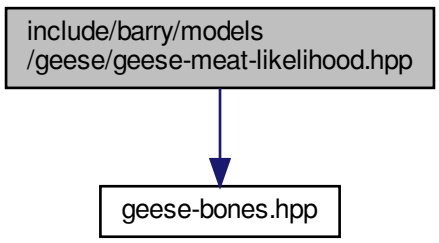
```
#define GEESE_MEAT_CONSTRUCTORS_HPP 1
```

Definition at line 4 of file geese-meat-constructors.hpp.

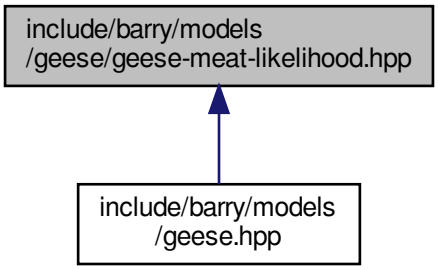
8.25 include/barry/models/geese/geese-meat-likelihood.hpp File Reference

```
#include "geese-bones.hpp"
```

Include dependency graph for geese-meat-likelihood.hpp:



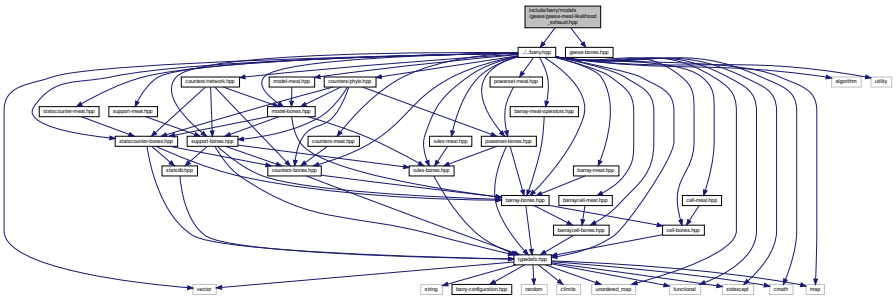
This graph shows which files directly or indirectly include this file:



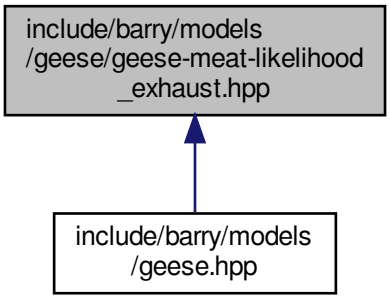
8.26 include/barry/models/geese/geese-meat-likelihood_exhaust.hpp File Reference

```
#include "../../barry.hpp"
#include "geese-bones.hpp"
```

Include dependency graph for geese-meat-likelihood_exhaust.hpp:



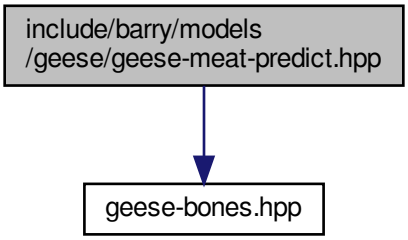
This graph shows which files directly or indirectly include this file:



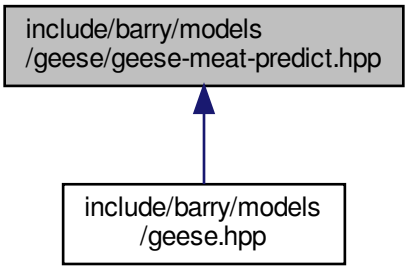
8.27 include/barry/models/geese/geese-meat-predict.hpp File Reference

```
#include "geese-bones.hpp"
```

Include dependency graph for geese-meat-predict.hpp:



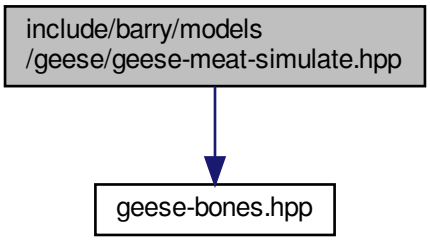
This graph shows which files directly or indirectly include this file:



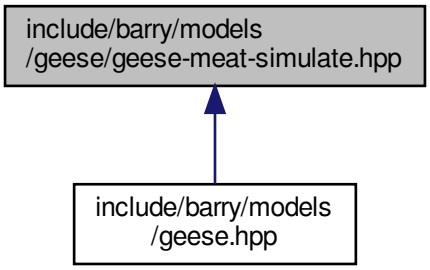
8.28 include/barry/models/geese/geese-meat-simulate.hpp File Reference

```
#include "geese-bones.hpp"
```

Include dependency graph for geese-meat-simulate.hpp:



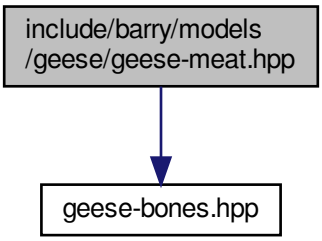
This graph shows which files directly or indirectly include this file:



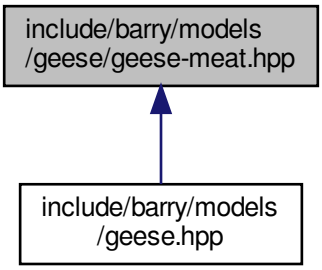
8.29 include/barry/models/geese/geese-meat.hpp File Reference

```
#include "geese-bones.hpp"
```

Include dependency graph for geese-meat.hpp:

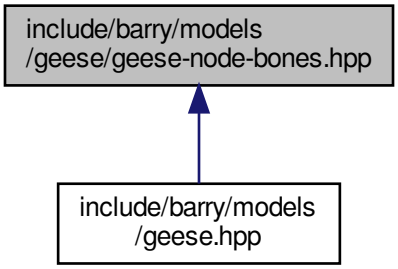


This graph shows which files directly or indirectly include this file:



8.30 include/barry/models/geese/geese-node-bones.hpp File Reference

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



Classes

- class Node

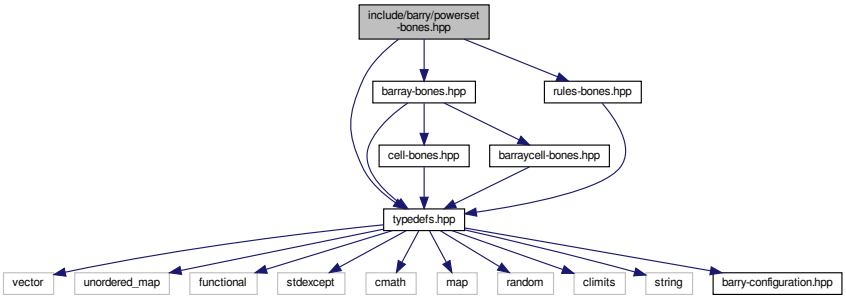
  A single node for the model.

8.31 include/barry/powerset-bones.hpp File Reference

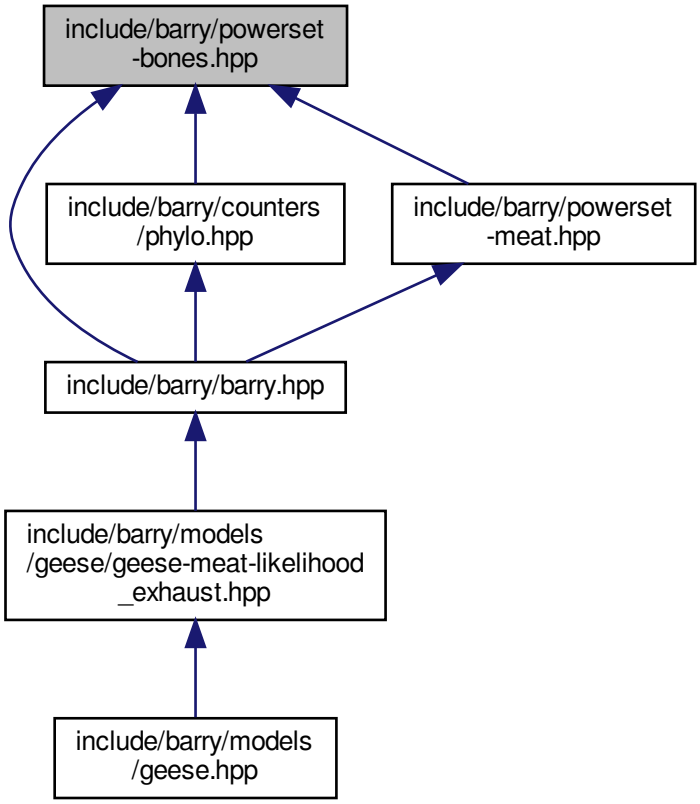
```
#include "typedefs.hpp"
#include "barray-bones.hpp"
```

```
#include "rules-bones.hpp"
```

Include dependency graph for powerset-bones.hpp:



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



Classes

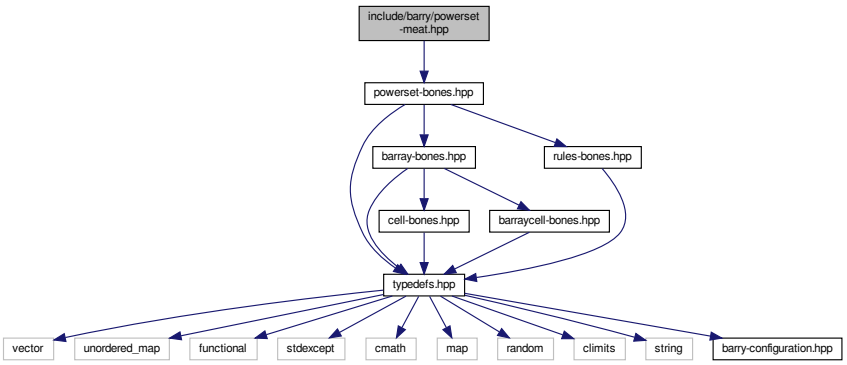
- class PowerSet< Array_Type, Data_Rule_Type >

  Powerset of a binary array.

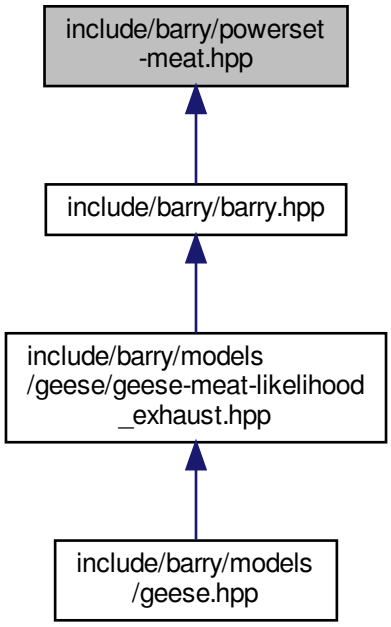
8.32 include/barry/powerset-meat.hpp File Reference

```
#include "powerset-bones.hpp"
```

Include dependency graph for powerset-meat.hpp:



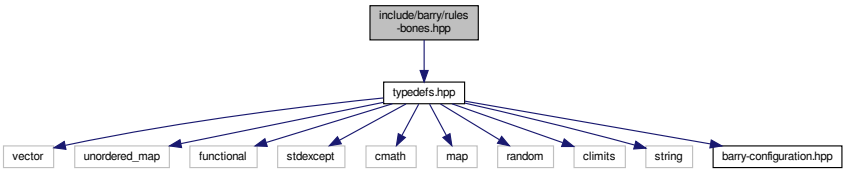
This graph shows which files directly or indirectly include this file:



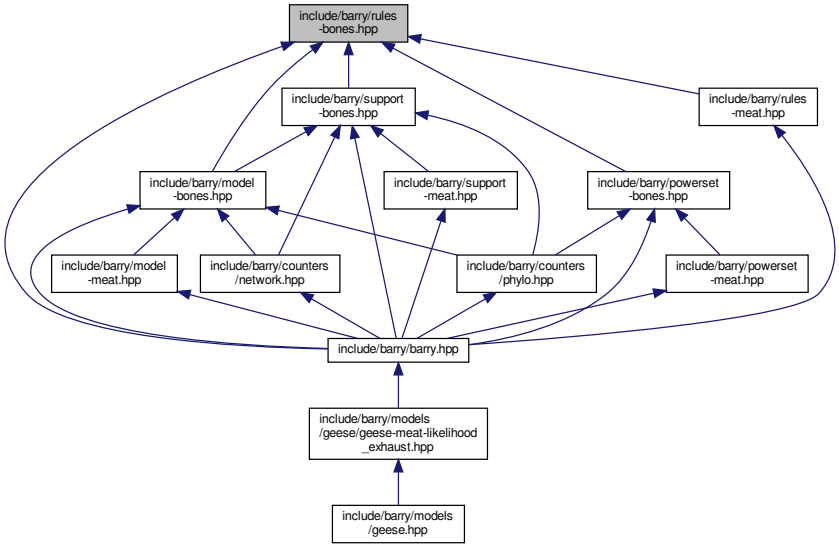
8.33 include/barry/rules-bones.hpp File Reference

```
#include "typedefs.hpp"
```

Include dependency graph for rules-bones.hpp:



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



Classes

- class Rule< Array_Type, Data_Type >

  Rule for determining if a cell should be included in a sequence.
- class Rules< Array_Type, Data_Type >

  Vector of objects of class Rule.

Functions

- template<typename Array_Type , typename Data_Type >
  bool rule_fun_default (const Array_Type *array, uint i, uint j, Data_Type *dat)

8.33.1 Function Documentation

8.33.1.1 rule_fun_default()

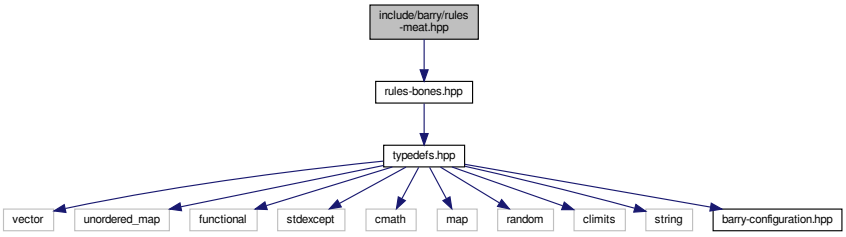
```
template<typename Array_Type , typename Data_Type >
bool rule_fun_default (
            const Array_Type * array,
            uint i,
            uint j,
            Data_Type * dat )
```

Definition at line 10 of file rules-bones.hpp.

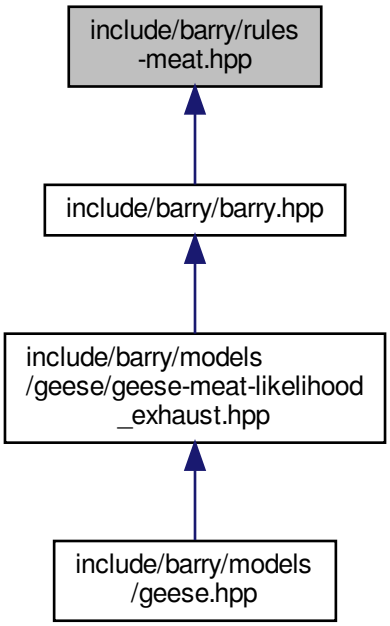
8.34 include/barry/rules-meat.hpp File Reference

```
#include "rules-bones.hpp"
```

Include dependency graph for rules-meat.hpp:



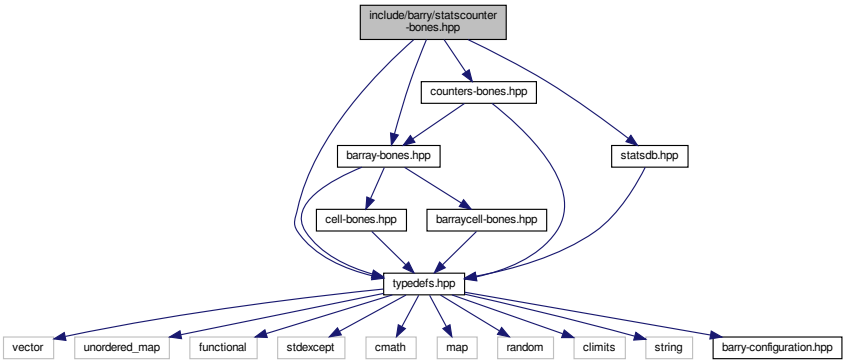
This graph shows which files directly or indirectly include this file:



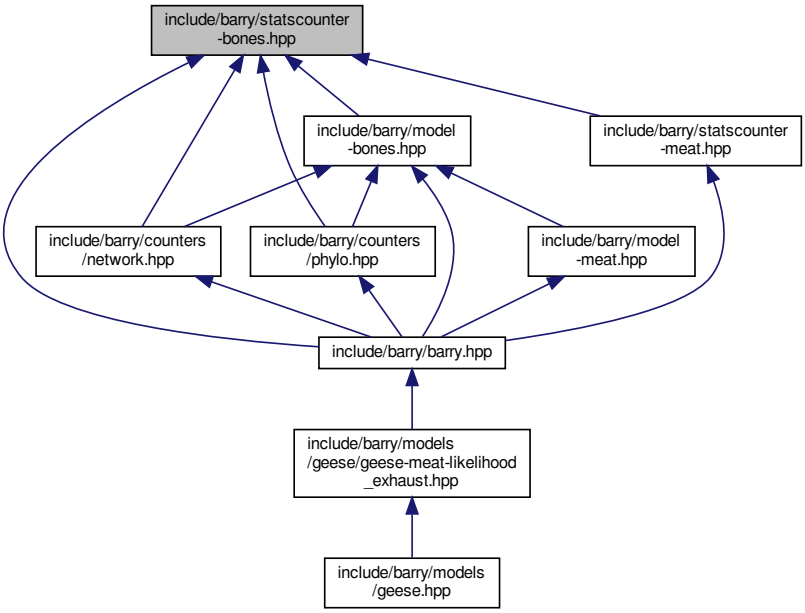
8.35 include/barry/statscounter-bones.hpp File Reference

```
#include "typedefs.hpp"
#include "barray-bones.hpp"
#include "statsdb.hpp"
#include "counters-bones.hpp"
```

Include dependency graph for statscounter-bones.hpp:



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



Classes

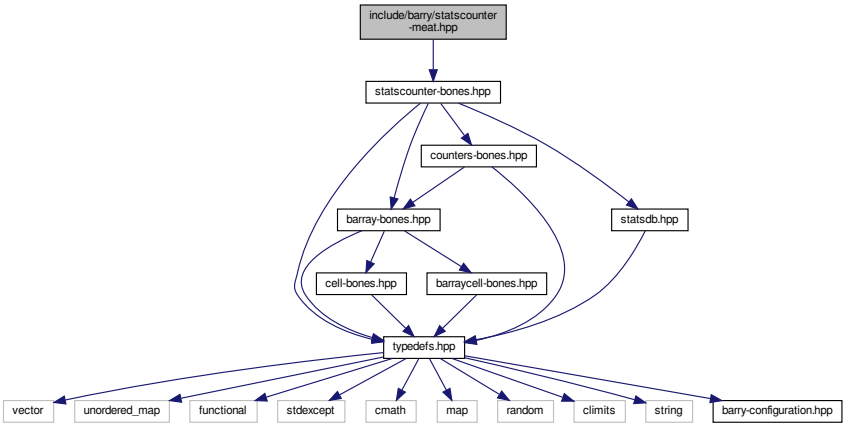
- class StatsCounter< Array_Type, Data_Type >

  Count stats for a single Array.

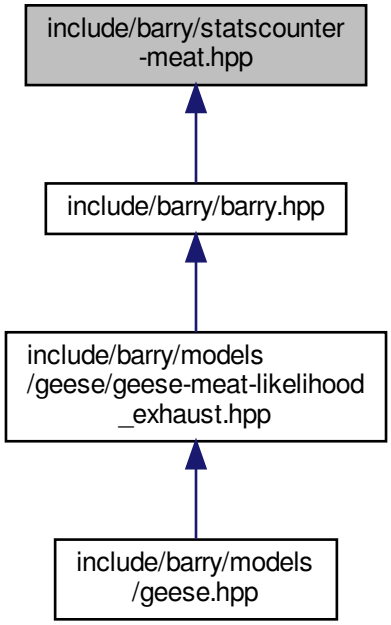
8.36 include/barry/statscounter-meat.hpp File Reference

```
#include "statscounter-bones.hpp"
```

Include dependency graph for statscounter-meat.hpp:



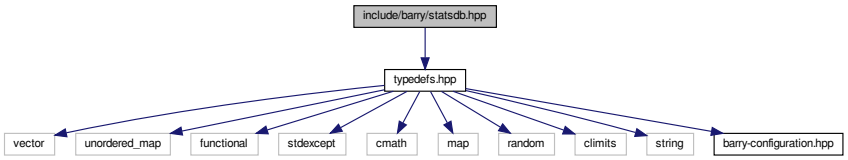
This graph shows which files directly or indirectly include this file:



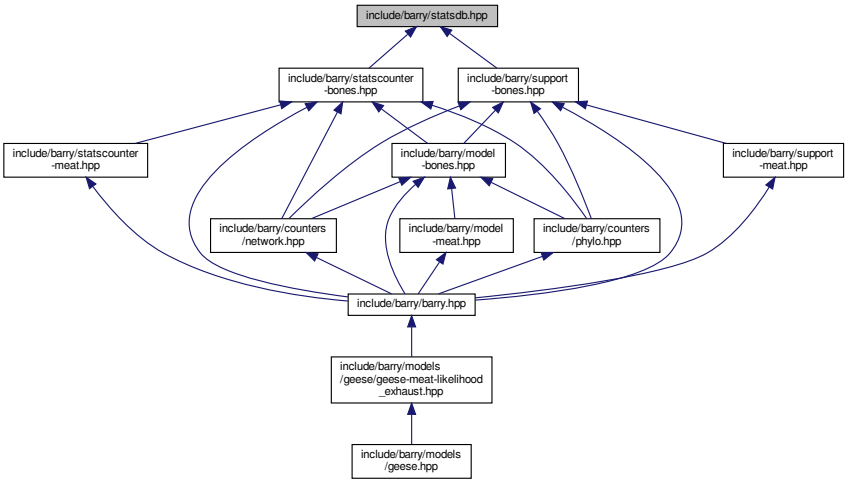
8.37 include/barry/statsdb.hpp File Reference

```
#include "typedefs.hpp"
```

Include dependency graph for statsdb.hpp:



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



Classes

- class FreqTable< T >

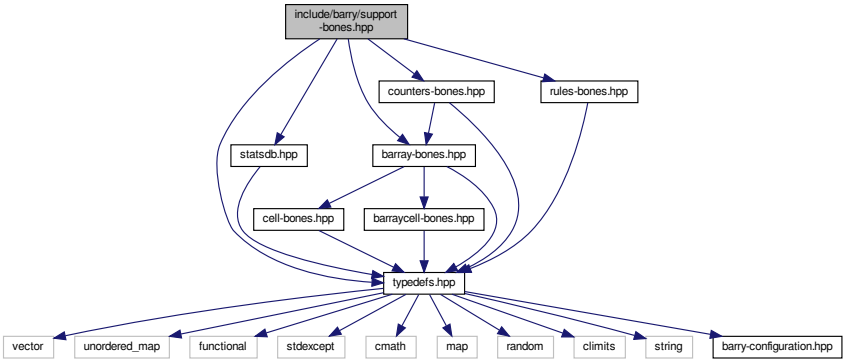
  Database of statistics.

8.38 include/barry/support-bones.hpp File Reference

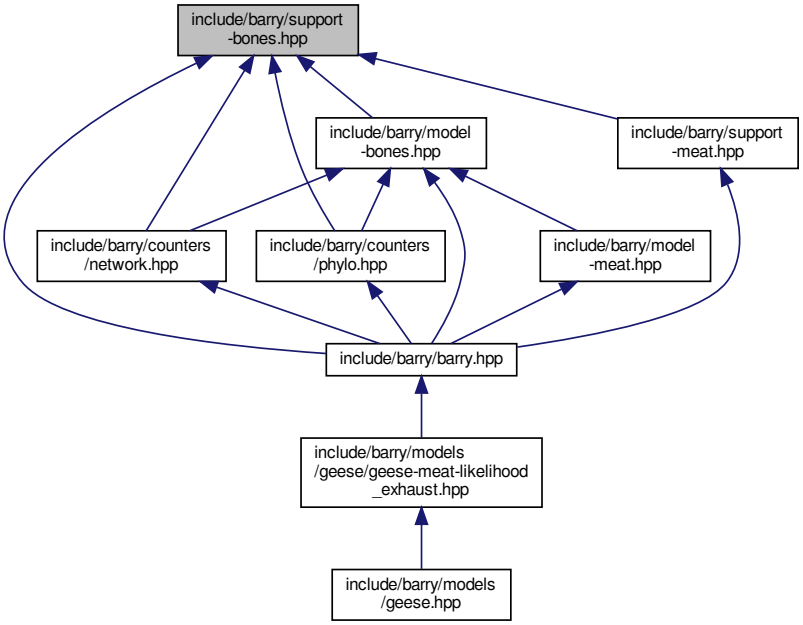
```
#include "typedefs.hpp"
#include "barray-bones.hpp"
#include "statsdb.hpp"
#include "counters-bones.hpp"
```

```
#include "rules-bones.hpp"
```

Include dependency graph for support-bones.hpp:



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



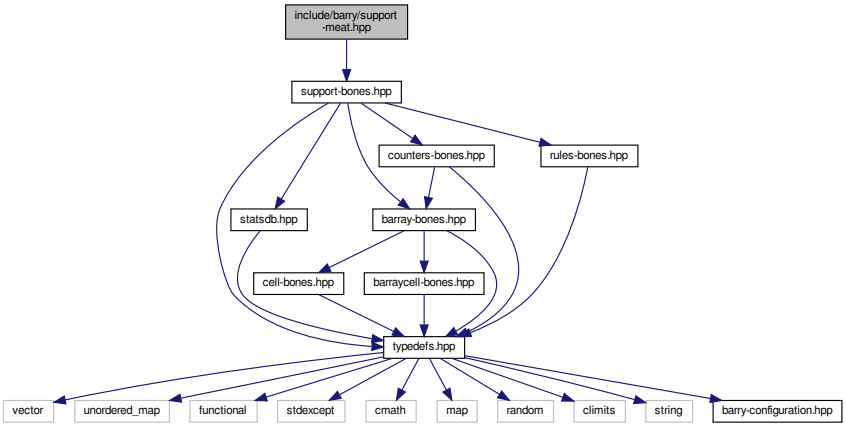
Classes

- class Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >

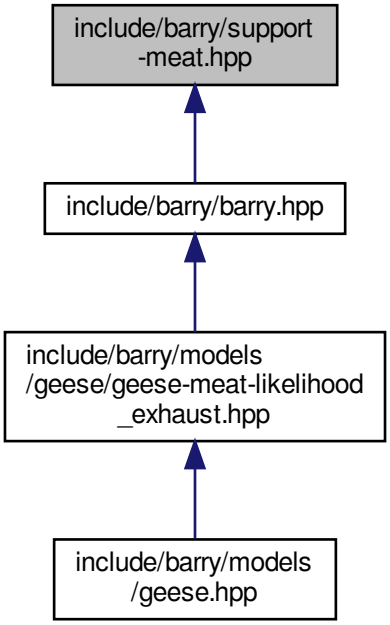
  Compute the support of sufficient statistics.

8.39 include/barry/support-meat.hpp File Reference

`#include "support-bones.hpp"`
Include dependency graph for support-meat.hpp:



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



Macros

- #define BARRY_SUPPORT_MEAT_HPP 1

8.39.1 Macro Definition Documentation

8.39.1.1 BARRY_SUPPORT_MEAT_HPP

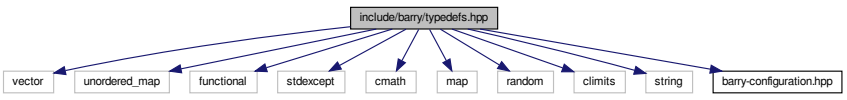
```
#define BARRY_SUPPORT_MEAT_HPP 1
```

Definition at line 4 of file support-meat.hpp.

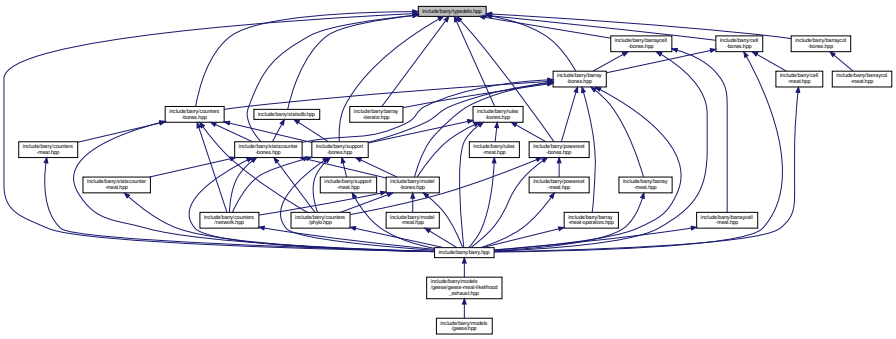
8.40 include/barry/typedefs.hpp File Reference

```
#include <vector>
#include <unordered_map>
#include <functional>
#include <stdexcept>
#include <cmath>
#include <map>
#include <random>
#include <climits>
#include <string>
#include "barry-configuration.hpp"
```

Include dependency graph for typedefs.hpp:



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



Classes

- class Entries< Cell_Type >

  A wrapper class to store `source`, `target`, `val` *from a* `BArray` *object.*
- struct vecHasher< T >

Namespaces

- CHECK

  Integer constants used to specify which cell should be check.
- EXISTS

  Integer constants used to specify which cell should be check to exist or not.

Typedefs

- typedef unsigned int uint
- typedef std::vector< std::pair< std::vector< double >, uint > > Counts_type
- template<typename Cell_Type >
  using Row_type = Map< uint, Cell< Cell_Type > >
- template<typename Cell_Type >
  using Col_type = Map< uint, Cell< Cell_Type > * >
- template<typename Ta = double, typename Tb = uint>
  using MapVec_type = std::unordered_map< std::vector< Ta >, Tb, vecHasher< Ta > >
- template<typename Array_Type , typename Data_Type >
  using Counter_fun_type = std::function< double(const Array_Type &, uint, uint, Data_Type *)>

  Counter and rule functions.
- template<typename Array_Type , typename Data_Type >
  using Rule_fun_type = std::function< bool(const Array_Type &, uint, uint, Data_Type *)>

Functions

- template<typename T >
  T vec_inner_prod (const std::vector< T > &a, const std::vector< T > &b)
- template<typename T >
  bool vec_equal (const std::vector< T > &a, const std::vector< T > &b)

  Compares if -a- and -b- are equal.
- template<typename T >
  bool vec_equal_approx (const std::vector< T > &a, const std::vector< T > &b, double eps=1e-10)

Variables

- const int CHECK::BOTH = -1
- const int CHECK::NONE = 0
- const int CHECK::ONE = 1
- const int CHECK::TWO = 2
- const int EXISTS::BOTH = -1
- const int EXISTS::NONE = 0
- const int EXISTS::ONE = 1
- const int EXISTS::TWO = 1
- const int EXISTS::UKNOWN = -1
- const int EXISTS::AS_ZERO = 0
- const int EXISTS::AS_ONE = 1

8.40.1 Typedef Documentation

8.40.1.1 Col_type

```
template<typename Cell_Type >
using Col_type = Map< uint, Cell<Cell_Type>* >
```

Definition at line 61 of file typedefs.hpp.

8.40.1.2 Counter_fun_type

```
template<typename Array_Type , typename Data_Type >
using Counter_fun_type = std::function<double(const Array_Type &, uint, uint, Data_Type *)>
```

Counter and rule functions.

Parameters

Array_Type	a BArray
unit,uint	Focal cell
Data_Type	Data associated with the function, for example, id of the attribute in the Array.

Returns

`Counter_fun_type` a double (the change statistic)

`Rule_fun_type` a bool. True if the cell is blocked.

Definition at line 133 of file typedefs.hpp.

8.40.1.3 Counts_type

```
typedef std::vector< std::pair< std::vector<double>, uint > > Counts_type
```

Definition at line 54 of file typedefs.hpp.

8.40.1.4 MapVec_type

```
template<typename Ta = double, typename Tb = uint>
using MapVec_type = std::unordered_map< std::vector< Ta >, Tb, vecHasher<Ta> >
```

Definition at line 115 of file typedefs.hpp.

8.40.1.5 Row_type

```
template<typename Cell_Type >
using Row_type = Map< uint, Cell<Cell_Type> >
```

Definition at line 58 of file typedefs.hpp.

8.40.1.6 Rule_fun_type

```
template<typename Array_Type , typename Data_Type >
using Rule_fun_type = std::function<bool(const Array_Type &, uint, uint, Data_Type *)>
```

Definition at line 136 of file typedefs.hpp.

8.40.1.7 uint

```
typedef unsigned int uint
```

Definition at line 20 of file typedefs.hpp.

8.40.2 Function Documentation

8.40.2.1 vec_equal()

```
template<typename T >
bool vec_equal (
            const std::vector< T > & a,
            const std::vector< T > & b )  [inline]
```

Compares if -a- and -b- are equal.

Parameters

a,b	Two vectors of the same length

Returns

`true` if all elements are equal.

Definition at line 147 of file typedefs.hpp.

8.40.2.2 vec_equal_approx()

```
template<typename T >
bool vec_equal_approx (
             const std::vector< T > & a,
             const std::vector< T > & b,
             double eps = 1e-10 )  [inline]
```

Definition at line 165 of file typedefs.hpp.

8.40.2.3 vec_inner_prod()

```
template<typename T >
T vec_inner_prod (
             const std::vector< T > & a,
             const std::vector< T > & b )  [inline]
```

Definition at line 185 of file typedefs.hpp.

8.41 README.md File Reference

Index

~BArray
BArray< Cell_Type, Data_Type >, 29
~BArrayCell
BArrayCell< Cell_Type, Data_Type >, 40
~BArrayCell_const
BArrayCell_const< Cell_Type, Data_Type >, 42
~BArrayCol
BArrayCol< Cell_Type, Data_Type >, 45
~BArrayCol_const
BArrayCol_const< Cell_Type, Data_Type >, 48
~Cell
Cell< Cell_Type >, 51
~ConstBArrayRowIter
ConstBArrayRowIter< Cell_Type, Data_Type >, 55
~Counter
Counter< Array_Type, Data_Type >, 58
~Counters
Counters< Array_Type, Data_Type >, 61
~Entries
Entries< Cell_Type >, 64
~Flock
Flock, 66
~FreqTable
FreqTable< T >, 71
~Geese
Geese, 76
~Model
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 87
~NetCounterData
NetCounterData, 100
~NetworkData
NetworkData, 103
~Node
Node, 105
~NodeData
NodeData, 109
~PowerSet
PowerSet< Array_Type, Data_Rule_Type >, 113
~Rule
Rule< Array_Type, Data_Type >, 118
~Rules
Rules< Array_Type, Data_Type >, 120
~StatsCounter
StatsCounter< Array_Type, Data_Type >, 124
~Support
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 131

add
Cell< Cell_Type >, 52
FreqTable< T >, 72
add_array
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 87
add_counter
Counters< Array_Type, Data_Type >, 61, 62
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 87, 88
StatsCounter< Array_Type, Data_Type >, 125
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 131
add_data
Flock, 67
add_rule
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 88, 89
PowerSet< Array_Type, Data_Rule_Type >, 113
Rules< Array_Type, Data_Type >, 121
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 131, 132
add_rule_dyn
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 89
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 132
annotations
Node, 106
Array
ConstBArrayRowIter< Cell_Type, Data_Type >, 55
StatsCounter< Array_Type, Data_Type >, 126
array
Node, 106
array_frequency
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 95
arrays
Node, 106
arrays2support

Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 95
AS_ONE
EXISTS, 23
as_vector
FreqTable< T >, 72
AS_ZERO
EXISTS, 23

BArray
BArray< Cell_Type, Data_Type >, 28, 29
BArray< Cell_Type, Data_Type >, 25
∼BArray, 29
BArray, 28, 29
BArrayCell< Cell_Type, Data_Type >, 38
BArrayCell_const< Cell_Type, Data_Type >, 38
clear, 29
col, 29
D, 30
default_val, 30
get_cell, 30
get_col, 30
get_col_vec, 30, 31
get_entries, 31
get_row, 31
get_row_vec, 31
insert_cell, 32
is_empty, 32
ncol, 32
nnozero, 33
nrow, 33
operator∗=, 33
operator(), 33
operator+=, 33, 34
operator-=, 34
operator/=, 34
operator=, 35
operator==, 35
out_of_range, 35
print, 35
reserve, 35
resize, 36
rm_cell, 36
row, 36
set_data, 36
swap_cells, 37
swap_cols, 37
swap_rows, 37
toggle_cell, 37
toggle_lock, 37
transpose, 38
visited, 39
zero_col, 38
zero_row, 38
barray-bones.hpp
BARRAY_BONES_HPP, 142
barray-iterator.hpp
BARRAY_ITERATOR_HPP, 143
barray-meat-operators.hpp
checkdim_, 145
COL, 144
ROW, 144
barray-meat.hpp
COL, 146
ROW, 146
BARRAY_BONES_HPP
barray-bones.hpp, 142
BARRAY_ITERATOR_HPP
barray-iterator.hpp, 143
BArrayCell
BArrayCell< Cell_Type, Data_Type >, 40
BArrayCell< Cell_Type, Data_Type >, 39
∼BArrayCell, 40
BArray< Cell_Type, Data_Type >, 38
BArrayCell, 40
operator Cell_Type, 40
operator∗=, 40
operator+=, 40
operator-=, 41
operator/=, 41
operator=, 41
operator==, 41
BArrayCell_const
BArrayCell_const< Cell_Type, Data_Type >, 42
BArrayCell_const< Cell_Type, Data_Type >, 42
∼BArrayCell_const, 42
BArray< Cell_Type, Data_Type >, 38
BArrayCell_const, 42
operator Cell_Type, 43
operator!=, 43
operator<, 43
operator<=, 43
operator>, 43
operator>=, 44
operator==, 43
BArrayCol
BArrayCol< Cell_Type, Data_Type >, 45
BArrayCol_const< Cell_Type, Data_Type >, 48
BArrayCol< Cell_Type, Data_Type >, 44
∼BArrayCol, 45
BArrayCol, 45
begin, 45
end, 45
operator Cell_Type, 45
operator∗=, 46
operator+=, 46
operator-=, 46
operator/=, 46
operator=, 46
operator==, 47
barraycol-meat.hpp
BARRY_BARRAYCOL_MEAT_HPP, 149
BArrayCol_const< Cell_Type, Data_Type >, 47
∼BArrayCol_const, 48
BArrayCol, 48
operator!=, 48

operator<, 48
operator<=, 48
operator>, 49
operator>=, 49
operator==, 49
barry, 21
barry-configuration.hpp
BARRY_CHECK_SUPPORT, 150
BARRY_ISFINITE, 151
BARRY_MAX_NUM_ELEMENTS, 151
BARRY_SAFE_EXP, 151
Map, 151
barry.hpp
COUNTER_FUNCTION, 153
COUNTER_LAMBDA, 153
RULE_FUNCTION, 154
RULE_LAMBDA, 154
barry::counters, 21
barry::counters::network, 22
barry::counters::phylo, 22
BARRY_BARRAYCOL_MEAT_HPP
barraycol-meat.hpp, 149
BARRY_CHECK_SUPPORT
barry-configuration.hpp, 150
BARRY_ISFINITE
barry-configuration.hpp, 151
BARRY_MAX_NUM_ELEMENTS
barry-configuration.hpp, 151
BARRY_SAFE_EXP
barry-configuration.hpp, 151
BARRY_SUPPORT_MEAT_HPP
support-meat.hpp, 192
begin
BArrayCol< Cell_Type, Data_Type >, 45
PowerSet< Array_Type, Data_Rule_Type >, 113
blengths
NodeData, 110
BOTH
CHECK, 22
EXISTS, 23

calc
PowerSet< Array_Type, Data_Rule_Type >, 114
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 132
calc_reduced_sequence
Geese, 76
calc_sequence
Geese, 76
Cell
Cell< Cell_Type >, 50–52
Cell< Cell_Type >, 49
~Cell, 51
add, 52
Cell, 50–52
operator Cell_Type, 53
operator=, 53
value, 53
visited, 53
change_stats
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 135
CHECK, 22
BOTH, 22
NONE, 22
ONE, 22
TWO, 22
checkdim_
barray-meat-operators.hpp, 145
clear
BArray< Cell_Type, Data_Type >, 29
Counters< Array_Type, Data_Type >, 62
FreqTable< T >, 72
Rules< Array_Type, Data_Type >, 121
COL
barray-meat-operators.hpp, 144
barray-meat.hpp, 146
col
BArray< Cell_Type, Data_Type >, 29
Col_type
typedefs.hpp, 194
ConstBArrayRowIter
ConstBArrayRowIter< Cell_Type, Data_Type >, 55
ConstBArrayRowIter< Cell_Type, Data_Type >, 54
~ConstBArrayRowIter, 55
Array, 55
ConstBArrayRowIter, 55
current_col, 55
current_row, 55
iter, 56
coordinates_free
PowerSet< Array_Type, Data_Rule_Type >, 115
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 135
coordinates_locked
PowerSet< Array_Type, Data_Rule_Type >, 115
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 135
count
Counter< Array_Type, Data_Type >, 58
count_all
StatsCounter< Array_Type, Data_Type >, 125
count_current
StatsCounter< Array_Type, Data_Type >, 125
count_fun
Counter< Array_Type, Data_Type >, 59
count_init
StatsCounter< Array_Type, Data_Type >, 125
Counter
Counter< Array_Type, Data_Type >, 57
Counter< Array_Type, Data_Type >, 56
~Counter, 58
count, 58

count_fun, 59
Counter, 57
data, 59
delete_data, 59
desc, 59
init, 58
init_fun, 59
name, 59
operator=, 58
counter_absdiff
Network counters, 11
counter_co_opt
Phylo counters, 16
counter_cogain
Phylo counters, 16
counter_ctriads
Network counters, 11
counter_degree
Network counters, 11
counter_deleted
StatsCounter< Array_Type, Data_Type >, 126
counter_density
Network counters, 11
counter_diff
Network counters, 11
counter_edges
Network counters, 12
counter_fun
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 95
Counter_fun_type
typedefs.hpp, 194
COUNTER_FUNCTION
barry.hpp, 153
counter_gains
Phylo counters, 16
counter_gains_k_offspring
Phylo counters, 17
counter_genes_changing
Phylo counters, 17
counter_idegree
Network counters, 12
counter_idegree15
Network counters, 12
counter_isolates
Network counters, 12
counter_istar2
Network counters, 12
COUNTER_LAMBDA
barry.hpp, 153
counter_longest
Phylo counters, 17
counter_loss
Phylo counters, 17
counter_maxfuns
Phylo counters, 18
counter_mutual
Network counters, 13
counter_neofun
Phylo counters, 18
counter_neofun_a2b
Phylo counters, 18
counter_nodecov
Network counters, 13
counter_nodeicov
Network counters, 13
counter_nodematch
Network counters, 13
counter_nodeocov
Network counters, 13
counter_odegree
Network counters, 14
counter_odegree15
Network counters, 14
counter_ostar2
Network counters, 14
counter_overall_changes
Phylo counters, 18
counter_overall_gains
Phylo counters, 19
counter_overall_loss
Phylo counters, 19
counter_subfun
Phylo counters, 19
counter_ttriads
Network counters, 14
Counters
Counters< Array_Type, Data_Type >, 61
counters
Geese, 80
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 95
StatsCounter< Array_Type, Data_Type >, 127
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 135
Counters< Array_Type, Data_Type >, 60
~Counters, 61
add_counter, 61, 62
clear, 62
Counters, 61
operator=, 62
operator[], 62
size, 63
counters_ptr
Flock, 67
Counting, 9
Counts_type
typedefs.hpp, 194
current_col
ConstBArrayRowIter< Cell_Type, Data_Type >, 55
current_row
ConstBArrayRowIter< Cell_Type, Data_Type >, 55
current_stats

StatsCounter< Array_Type, Data_Type >, 127
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 136

D
BArray< Cell_Type, Data_Type >, 30
Rule< Array_Type, Data_Type >, 119
dat
Flock, 70
data
Counter< Array_Type, Data_Type >, 59
PowerSet< Array_Type, Data_Rule_Type >, 116
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 136
default_val
BArray< Cell_Type, Data_Type >, 30
delete_counters
Geese, 80
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 136
delete_data
Counter< Array_Type, Data_Type >, 59
delete_rengine
Geese, 80
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 96
delete_rules
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 136
delete_rules_dyn
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 137
delete_support
Geese, 81
desc
Counter< Array_Type, Data_Type >, 59
directed
NetworkData, 103
duplication
Node, 107
NodeData, 110

EmptyArray
PowerSet< Array_Type, Data_Rule_Type >, 116
StatsCounter< Array_Type, Data_Type >, 127
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 137
end
BArrayCol< Cell_Type, Data_Type >, 45
PowerSet< Array_Type, Data_Rule_Type >, 114
Entries
Entries< Cell_Type >, 64
Entries< Cell_Type >, 63
~Entries, 64
Entries, 64
resize, 64
source, 65
target, 65
val, 65
EXISTS, 23
AS_ONE, 23
AS_ZERO, 23
BOTH, 23
NONE, 24
ONE, 24
TWO, 24
UKNOWN, 24

first_calc_done
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 96
Flock, 65
~Flock, 66
add_data, 67
counters_ptr, 67
dat, 70
Flock, 66
init, 67
initialized, 70
likelihood_joint, 67
nfunctions, 70
nfuns, 68
nleafs, 68
nnodes, 68
nterms, 68
ntrees, 69
operator(), 69
rengine, 70
set_seed, 69
support, 70
FreqTable
FreqTable< T >, 71
FreqTable< T >, 71
~FreqTable, 71
add, 72
as_vector, 72
clear, 72
FreqTable, 71
get_data, 72
get_data_ptr, 72
print, 72
reserve, 73
size, 73

Geese, 73
~Geese, 76
calc_reduced_sequence, 76
calc_sequence, 76
counters, 80
delete_counters, 80

delete_rengine, 80
delete_support, 81
Geese, 75, 76
get_probabilities, 76
inherit_support, 77
init, 77
init_node, 77
initialized, 81
likelihood, 77
likelihood_exhaust, 77
map_to_nodes, 81
nfunctions, 81
nfuns, 78
nleafs, 78
nnodes, 78
nodes, 81
nterms, 78
observed_counts, 78
operator=, 78, 79
predict, 79
predict_backend, 79
print_observed_counts, 79
reduced_sequence, 81
rengine, 82
sequence, 82
set_seed, 79
simulate, 80
states, 82
support, 82
update_annotations, 80
geese-bones.hpp
INITIALIZED, 175
keygen_full, 175
RULE_FUNCTION, 175
vec_diff, 176
vector_caster, 176
geese-meat-constructors.hpp
GEESE_MEAT_CONSTRUCTORS_HPP, 177
GEESE_MEAT_CONSTRUCTORS_HPP
geese-meat-constructors.hpp, 177
get_cell
BArray< Cell_Type, Data_Type >, 30
get_col
BArray< Cell_Type, Data_Type >, 30
get_col_vec
BArray< Cell_Type, Data_Type >, 30, 31
get_counts
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 133
get_counts_ptr
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 133
get_current_stats
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 133
get_data
FreqTable< T >, 72
PowerSet< Array_Type, Data_Rule_Type >, 114
get_data_ptr
FreqTable< T >, 72
PowerSet< Array_Type, Data_Rule_Type >, 114
get_entries
BArray< Cell_Type, Data_Type >, 31
get_last_name
phylo.hpp, 169
get_norm_const
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 90
get_parent
Node, 106
get_probabilities
Geese, 76
get_pset
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 90
get_row
BArray< Cell_Type, Data_Type >, 31
get_row_vec
BArray< Cell_Type, Data_Type >, 31
get_seq
Rules< Array_Type, Data_Type >, 121
get_stats
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 90

id
Node, 107
include/barry/barray-bones.hpp, 141
include/barry/barray-iterator.hpp, 142
include/barry/barray-meat-operators.hpp, 143
include/barry/barray-meat.hpp, 145
include/barry/barraycell-bones.hpp, 147
include/barry/barraycell-meat.hpp, 147
include/barry/barraycol-bones.hpp, 148
include/barry/barraycol-meat.hpp, 149
include/barry/barry-configuration.hpp, 150
include/barry/barry.hpp, 152
include/barry/cell-bones.hpp, 154
include/barry/cell-meat.hpp, 155
include/barry/col-bones.hpp, 156
include/barry/counters-bones.hpp, 156
include/barry/counters-meat.hpp, 157
include/barry/counters/network.hpp, 158
include/barry/counters/phylo.hpp, 164
include/barry/model-bones.hpp, 170
include/barry/model-meat.hpp, 172
include/barry/models/geese.hpp, 173
include/barry/models/geese/flock-bones.hpp, 173
include/barry/models/geese/flock-meet.hpp, 174
include/barry/models/geese/geese-bones.hpp, 174

include/barry/models/geese/geese-meat-constructors.hpp, 176
include/barry/models/geese/geese-meat-likelihood.hpp, 177
include/barry/models/geese/geese-meat-likelihood_exhaust.hpp, 178
include/barry/models/geese/geese-meat-predict.hpp, 179
include/barry/models/geese/geese-meat-simulate.hpp, 180
include/barry/models/geese/geese-meat.hpp, 181
include/barry/models/geese/geese-node-bones.hpp, 182
include/barry/powerset-bones.hpp, 182
include/barry/powerset-meat.hpp, 184
include/barry/rules-bones.hpp, 184
include/barry/rules-meat.hpp, 186
include/barry/statscounter-bones.hpp, 187
include/barry/statscounter-meat.hpp, 188
include/barry/statsdb.hpp, 188
include/barry/support-bones.hpp, 189
include/barry/support-meat.hpp, 191
include/barry/typedefs.hpp, 192
indices
  NetCounterData, 101
inherit_support
  Geese, 77
init
  Counter< Array_Type, Data_Type >, 58
  Flock, 67
  Geese, 77
init_fun
  Counter< Array_Type, Data_Type >, 59
init_node
  Geese, 77
init_support
  PowerSet< Array_Type, Data_Rule_Type >, 114
  Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 133
INITIALIZED
  geese-bones.hpp, 175
initialized
  Flock, 70
  Geese, 81
insert_cell
  BArray< Cell_Type, Data_Type >, 32
is_empty
  BArray< Cell_Type, Data_Type >, 32
is_leaf
  Node, 106
iter
  ConstBArrayRowIter< Cell_Type, Data_Type >, 56

keygen
  Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 96
keygen_default
  model-bones.hpp, 171
keygen_full
  geese-bones.hpp, 175
keys2support
  Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 96

likelihood
  Geese, 77
  Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 90, 91
likelihood_
  model-bones.hpp, 171
likelihood_exhaust
  Geese, 77
likelihood_joint
  Flock, 67
likelihood_total
  Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 91

M
  PowerSet< Array_Type, Data_Rule_Type >, 116
  Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 137
Map
  barry-configuration.hpp, 151
map_to_nodes
  Geese, 81
MapVec_type
  typedefs.hpp, 194
max_num_elements
  Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 137
Model
  Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 86
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 83
  ~Model, 87
  add_array, 87
  add_counter, 87, 88
  add_rule, 88, 89
  add_rule_dyn, 89
  array_frequency, 95
  arrays2support, 95
  counter_fun, 95
  counters, 95
  delete_rengine, 96
  first_calc_done, 96
  get_norm_const, 90
  get_pset, 90
  get_stats, 90

keygen, 96
keys2support, 96
likelihood, 90, 91
likelihood_total, 91
Model, 86
n_arrays_per_stats, 97
normalizing_constants, 97
nterms, 91
operator=, 92
params_last, 97
print_stats, 92
pset_arrays, 97
pset_probs, 98
pset_stats, 98
rengine, 98
rules, 98
rules_dyn, 98
sample, 92
set_counters, 93
set_keygen, 93
set_rengine, 93
set_rules, 93
set_rules_dyn, 94
set_seed, 94
size, 94
size_unique, 94
stats, 99
store_psets, 94
support_fun, 99
target_stats, 99
with_pset, 99
model-bones.hpp
keygen_default, 171
likelihood_, 171
update_normalizing_constant, 171

N
PowerSet< Array_Type, Data_Rule_Type >, 116
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 138
n_arrays_per_stats
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 97
name
Counter< Array_Type, Data_Type >, 59
narray
Node, 107
ncol
BArray< Cell_Type, Data_Type >, 32
NET_C_DATA_IDX
network.hpp, 161
NET_C_DATA_NUM
network.hpp, 161
NetCounter
network.hpp, 162
NetCounterData, 100
~NetCounterData, 100
indices, 101
NetCounterData, 100
numbers, 101
NetCounters
network.hpp, 162
NetModel
network.hpp, 162
NetRule
network.hpp, 163
NetRules
network.hpp, 163
NetStatsCounter
network.hpp, 163
NetSupport
network.hpp, 163
Network
network.hpp, 163
Network counters, 10
counter_absdiff, 11
counter_ctriads, 11
counter_degree, 11
counter_density, 11
counter_diff, 11
counter_edges, 12
counter_idegree, 12
counter_idegree15, 12
counter_isolates, 12
counter_istar2, 12
counter_mutual, 13
counter_nodecov, 13
counter_nodeicov, 13
counter_nodematch, 13
counter_nodeocov, 13
counter_odegree, 14
counter_odegree15, 14
counter_ostar2, 14
counter_ttriads, 14
NETWORK_COUNTER, 14
network.hpp
NET_C_DATA_IDX, 161
NET_C_DATA_NUM, 161
NetCounter, 162
NetCounters, 162
NetModel, 162
NetRule, 163
NetRules, 163
NetStatsCounter, 163
NetSupport, 163
Network, 163
NETWORK_COUNTER, 161
NETWORK_COUNTER_LAMBDA, 161
NETWORK_RULE, 161
NETWORK_RULE_LAMBDA, 162
rules_zerodiag, 164
NETWORK_COUNTER
Network counters, 14
network.hpp, 161
NETWORK_COUNTER_LAMBDA

network.hpp, 161
NETWORK_RULE
network.hpp, 161
NETWORK_RULE_LAMBDA
network.hpp, 162
NetworkData, 101
~NetworkData, 103
directed, 103
NetworkData, 102
vertex_attr, 103
nfunctions
Flock, 70
Geese, 81
nfuns
Flock, 68
Geese, 78
nleafs
Flock, 68
Geese, 78
nnodes
Flock, 68
Geese, 78
nnozero
BArray< Cell_Type, Data_Type >, 33
Node, 103
~Node, 105
annotations, 106
array, 106
arrays, 106
duplication, 107
get_parent, 106
id, 107
is_leaf, 106
narray, 107
Node, 104, 105
offspring, 107
ord, 107
parent, 108
probability, 108
subtree_prob, 108
visited, 108
NodeData, 109
~NodeData, 109
blengths, 110
duplication, 110
NodeData, 109
states, 110
nodes
Geese, 81
NONE
CHECK, 22
EXISTS, 24
normalizing_constants
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 97
nrow
BArray< Cell_Type, Data_Type >, 33
nterms
Flock, 68
Geese, 78
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 91
ntrees
Flock, 69
numbers
NetCounterData, 101

observed_counts
Geese, 78
offspring
Node, 107
ONE
CHECK, 22
EXISTS, 24
operator Cell_Type
BArrayCell< Cell_Type, Data_Type >, 40
BArrayCell_const< Cell_Type, Data_Type >, 43
BArrayCol< Cell_Type, Data_Type >, 45
Cell< Cell_Type >, 53
operator!=
BArrayCell_const< Cell_Type, Data_Type >, 43
BArrayCol_const< Cell_Type, Data_Type >, 48
operator<
BArrayCell_const< Cell_Type, Data_Type >, 43
BArrayCol_const< Cell_Type, Data_Type >, 48
operator<=
BArrayCell_const< Cell_Type, Data_Type >, 43
BArrayCol_const< Cell_Type, Data_Type >, 48
operator>
BArrayCell_const< Cell_Type, Data_Type >, 43
BArrayCol_const< Cell_Type, Data_Type >, 49
operator>=
BArrayCell_const< Cell_Type, Data_Type >, 44
BArrayCol_const< Cell_Type, Data_Type >, 49
operator*=
BArray< Cell_Type, Data_Type >, 33
BArrayCell< Cell_Type, Data_Type >, 40
BArrayCol< Cell_Type, Data_Type >, 46
operator()
BArray< Cell_Type, Data_Type >, 33
Flock, 69
Rule< Array_Type, Data_Type >, 119
Rules< Array_Type, Data_Type >, 122
vecHasher< T >, 139
operator+=
BArray< Cell_Type, Data_Type >, 33, 34
BArrayCell< Cell_Type, Data_Type >, 40
BArrayCol< Cell_Type, Data_Type >, 46
operator-=
BArray< Cell_Type, Data_Type >, 34
BArrayCell< Cell_Type, Data_Type >, 41
BArrayCol< Cell_Type, Data_Type >, 46
operator/=
BArray< Cell_Type, Data_Type >, 34
BArrayCell< Cell_Type, Data_Type >, 41

BArrayCol< Cell_Type, Data_Type >, 46
operator=
BArray< Cell_Type, Data_Type >, 35
BArrayCell< Cell_Type, Data_Type >, 41
BArrayCol< Cell_Type, Data_Type >, 46
Cell< Cell_Type >, 53
Counter< Array_Type, Data_Type >, 58
Counters< Array_Type, Data_Type >, 62
Geese, 78, 79
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 92
Rules< Array_Type, Data_Type >, 122
operator==
BArray< Cell_Type, Data_Type >, 35
BArrayCell< Cell_Type, Data_Type >, 41
BArrayCell_const< Cell_Type, Data_Type >, 43
BArrayCol< Cell_Type, Data_Type >, 47
BArrayCol_const< Cell_Type, Data_Type >, 49
operator[]
Counters< Array_Type, Data_Type >, 62
PowerSet< Array_Type, Data_Rule_Type >, 115
ord
Node, 107
out_of_range
BArray< Cell_Type, Data_Type >, 35

params_last
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 97
parent
Node, 108
Phylo counters, 15
counter_co_opt, 16
counter_cogain, 16
counter_gains, 16
counter_gains_k_offspring, 17
counter_genes_changing, 17
counter_longest, 17
counter_loss, 17
counter_maxfuns, 18
counter_neofun, 18
counter_neofun_a2b, 18
counter_overall_changes, 18
counter_overall_gains, 19
counter_overall_loss, 19
counter_subfun, 19
phylo.hpp
get_last_name, 169
PHYLO_C_DATA_IDX, 166
PHYLO_CHECK_MISSING, 166
PHYLO_COUNTER, 167
PHYLO_COUNTER_LAMBDA, 167
PhyloArray, 167
PhyloCounter, 167
PhyloCounterData, 168
PhyloCounters, 168
PhyloModel, 168
PhyloPowerSet, 168
PhyloRule, 168
PhyloRuleData, 168
PhyloRules, 169
PhyloStatsCounter, 169
PhyloSupport, 169
PHYLO_C_DATA_IDX
phylo.hpp, 166
PHYLO_CHECK_MISSING
phylo.hpp, 166
PHYLO_COUNTER
phylo.hpp, 167
PHYLO_COUNTER_LAMBDA
phylo.hpp, 167
PhyloArray
phylo.hpp, 167
PhyloCounter
phylo.hpp, 167
PhyloCounterData
phylo.hpp, 168
PhyloCounters
phylo.hpp, 168
PhyloModel
phylo.hpp, 168
PhyloPowerSet
phylo.hpp, 168
PhyloRule
phylo.hpp, 168
PhyloRuleData
phylo.hpp, 168
PhyloRules
phylo.hpp, 169
PhyloStatsCounter
phylo.hpp, 169
PhyloSupport
phylo.hpp, 169
PowerSet
PowerSet< Array_Type, Data_Rule_Type >, 112
PowerSet< Array_Type, Data_Rule_Type >, 111
~PowerSet, 113
add_rule, 113
begin, 113
calc, 114
coordinates_free, 115
coordinates_locked, 115
data, 116
EmptyArray, 116
end, 114
get_data, 114
get_data_ptr, 114
init_support, 114
M, 116
N, 116
operator[], 115
PowerSet, 112
reset, 115
rules, 116
rules_deleted, 117

size, 115
predict
Geese, 79
predict_backend
Geese, 79
print
BArray< Cell_Type, Data_Type >, 35
FreqTable< T >, 72
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 133
print_observed_counts
Geese, 79
print_stats
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 92
probability
Node, 108
pset_arrays
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 97
pset_probs
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 98
pset_stats
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 98

README.md, 196
reduced_sequence
Geese, 81
rengine
Flock, 70
Geese, 82
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 98
reserve
BArray< Cell_Type, Data_Type >, 35
FreqTable< T >, 73
reset
PowerSet< Array_Type, Data_Rule_Type >, 115
reset_array
StatsCounter< Array_Type, Data_Type >, 126
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 134
resize
BArray< Cell_Type, Data_Type >, 36
Entries< Cell_Type >, 64
rm_cell
BArray< Cell_Type, Data_Type >, 36
ROW
barray-meat-operators.hpp, 144
barray-meat.hpp, 146
row
BArray< Cell_Type, Data_Type >, 36
Row_type
typedefs.hpp, 194
Rule
Rule< Array_Type, Data_Type >, 118
Rule< Array_Type, Data_Type >, 117
~Rule, 118
D, 119
operator(), 119
Rule, 118
rule_fun_default
rules-bones.hpp, 185
Rule_fun_type
typedefs.hpp, 195
RULE_FUNCTION
barry.hpp, 154
geese-bones.hpp, 175
RULE_LAMBDA
barry.hpp, 154
Rules
Rules< Array_Type, Data_Type >, 120
rules
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 98
PowerSet< Array_Type, Data_Rule_Type >, 116
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 138
Rules< Array_Type, Data_Type >, 119
~Rules, 120
add_rule, 121
clear, 121
get_seq, 121
operator(), 122
operator=, 122
Rules, 120
size, 122
rules-bones.hpp
rule_fun_default, 185
rules_deleted
PowerSet< Array_Type, Data_Rule_Type >, 117
rules_dyn
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 98
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 138
rules_zerodiag
network.hpp, 164

sample
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 92
sequence
Geese, 82

set_counters
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 93
StatsCounter< Array_Type, Data_Type >, 126
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 134
set_data
BArray< Cell_Type, Data_Type >, 36
set_keygen
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 93
set_rengine
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 93
set_rules
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 93
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 134
set_rules_dyn
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 94
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 134
set_seed
Flock, 69
Geese, 79
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 94
simulate
Geese, 80
size
Counters< Array_Type, Data_Type >, 63
FreqTable< T >, 73
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 94
PowerSet< Array_Type, Data_Rule_Type >, 115
Rules< Array_Type, Data_Type >, 122
size_unique
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 94
source
Entries< Cell_Type >, 65
states
Geese, 82
NodeData, 110
Statistical Models, 9
stats
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 99
StatsCounter
StatsCounter< Array_Type, Data_Type >, 124
StatsCounter< Array_Type, Data_Type >, 123
~StatsCounter, 124
add_counter, 125
Array, 126
count_all, 125
count_current, 125
count_init, 125
counter_deleted, 126
counters, 127
current_stats, 127
EmptyArray, 127
reset_array, 126
set_counters, 126
StatsCounter, 124
store_psets
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 94
subtree_prob
Node, 108
Support
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 130
support
Flock, 70
Geese, 82
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 128
~Support, 131
add_counter, 131
add_rule, 131, 132
add_rule_dyn, 132
calc, 132
change_stats, 135
coordinates_free, 135
coordinates_locked, 135
counters, 135
current_stats, 136
data, 136
delete_counters, 136
delete_rules, 136
delete_rules_dyn, 137
EmptyArray, 137
get_counts, 133
get_counts_ptr, 133
get_current_stats, 133
init_support, 133
M, 137
max_num_elements, 137
N, 138
print, 133

reset_array, 134
rules, 138
rules_dyn, 138
set_counters, 134
set_rules, 134
set_rules_dyn, 134
Support, 130
support-meat.hpp
BARRY_SUPPORT_MEAT_HPP, 192
support_fun
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 99
swap_cells
BArray< Cell_Type, Data_Type >, 37
swap_cols
BArray< Cell_Type, Data_Type >, 37
swap_rows
BArray< Cell_Type, Data_Type >, 37

target
Entries< Cell_Type >, 65
target_stats
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 99
toggle_cell
BArray< Cell_Type, Data_Type >, 37
toggle_lock
BArray< Cell_Type, Data_Type >, 37
transpose
BArray< Cell_Type, Data_Type >, 38
TWO
CHECK, 22
EXISTS, 24
typedefs.hpp
Col_type, 194
Counter_fun_type, 194
Counts_type, 194
MapVec_type, 194
Row_type, 194
Rule_fun_type, 195
uint, 195
vec_equal, 195
vec_equal_approx, 195
vec_inner_prod, 196

uint
typedefs.hpp, 195
UKNOWN
EXISTS, 24
update_annotations
Geese, 80
update_normalizing_constant
model-bones.hpp, 171

val
Entries< Cell_Type >, 65
value
Cell< Cell_Type >, 53
vec_diff
geese-bones.hpp, 176
vec_equal
typedefs.hpp, 195
vec_equal_approx
typedefs.hpp, 195
vec_inner_prod
typedefs.hpp, 196
vecHasher< T >, 139
operator(), 139
vector_caster
geese-bones.hpp, 176
vertex_attr
NetworkData, 103
visited
BArray< Cell_Type, Data_Type >, 39
Cell< Cell_Type >, 53
Node, 108

with_pset
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 99

zero_col
BArray< Cell_Type, Data_Type >, 38
zero_row
BArray< Cell_Type, Data_Type >, 38