barry: Your go-to motif accountant

0.0-1

Generated by Doxygen 1.9.1

1 Main Page 1

2 Module Index 5

2.1 Modules . . . 5

3 Hierarchical Index 7

3.1 Class Hierarchy . . . 7

4 Class Index 9

4.1 Class List . . . 9

5 File Index 11

5.1 File List . . . 11

6 Module Documentation 13

6.1 Counting . . . 13

6.1.1 Detailed Description . . . 15

6.1.2 Macro Definition Documentation . . . 15

6.1.2.1 IF_MATCHES . . . 15

6.1.2.2 IF_NOTMATCHES . . . 15

6.1.2.3 IS_DUPLICATION . . . 15

6.1.2.4 IS_EITHER . . . 16

6.1.2.5 IS_SPECIATION . . . 16

6.1.2.6 MAKE_DEFM_HASHER . . . 16

6.1.2.7 MAKE_DUPL_VARS . . . 16

6.1.2.8 PHYLO_CHECK_MISSING . . . 17

6.1.2.9 PHYLO_COUNTER_LAMBDA . . . 17

6.1.2.10 PHYLO_RULE_DYN_LAMBDA . . . 17

6.1.3 Function Documentation . . . 17

6.1.3.1 counter_co_opt() . . . 18

6.1.3.2 counter_cogain() . . . 18

6.1.3.3 counter_gains() . . . 18

6.1.3.4 counter_gains_from_0() . . . 19

6.1.3.5 counter_gains_k_offspring() . . . 19

6.1.3.6 counter_genes_changing() . . . 19

6.1.3.7 counter_k_genes_changing() . . . 19

6.1.3.8 counter_less_than_p_prop_genes_changing() . . . 20

6.1.3.9 counter_longest() . . . 20

6.1.3.10 counter_loss() . . . 20

6.1.3.11 counter_maxfuns() . . . 20

6.1.3.12 counter_neofun() . . . 21

6.1.3.13 counter_neofun_a2b() . . . 21

6.1.3.14 counter_overall_changes() . . . 21

6.1.3.15 counter_overall_gains() . . . 21

6.1.3.16 counter_overall_gains_from_0() . . . 22
6.1.3.17 counter_overall_loss() . . . 22
6.1.3.18 counter_pairwise_first_gain() . . . 22
6.1.3.19 counter_pairwise_neofun_singlefun() . . . 22
6.1.3.20 counter_pairwise_overall_change() . . . 23
6.1.3.21 counter_pairwise_preserving() . . . 23
6.1.3.22 counter_preserve_pseudogene() . . . 23
6.1.3.23 counter_prop_genes_changing() . . . 23
6.1.3.24 counter_subfun() . . . 24
6.1.3.25 get_last_name() . . . 24
6.2 Statistical Models . . . 24
6.2.1 Detailed Description . . . 24
6.3 Network counters . . . 24
6.3.1 Detailed Description . . . 26
6.3.2 Function Documentation . . . 26
6.3.2.1 counter_absdiff() . . . 27
6.3.2.2 counter_ctriads() [1/2] . . . 27
6.3.2.3 counter_ctriads() [2/2] . . . 27
6.3.2.4 counter_degree() . . . 27
6.3.2.5 counter_density() . . . 28
6.3.2.6 counter_diff() . . . 28
6.3.2.7 counter_edges() . . . 28
6.3.2.8 counter_fixed_effect() . . . 28
6.3.2.9 counter_idegree() [1/2] . . . 29
6.3.2.10 counter_idegree() [2/2] . . . 29
6.3.2.11 counter_idegree15() [1/2] . . . 29
6.3.2.12 counter_idegree15() [2/2] . . . 29
6.3.2.13 counter_isolates() [1/2] . . . 30
6.3.2.14 counter_isolates() [2/2] . . . 30
6.3.2.15 counter_istar2() [1/2] . . . 30
6.3.2.16 counter_istar2() [2/2] . . . 30
6.3.2.17 counter_logit_intercept() . . . 30
6.3.2.18 counter_mutual() . . . 31
6.3.2.19 counter_nodecov() . . . 31
6.3.2.20 counter_nodeicov() . . . 31
6.3.2.21 counter_nodematch() . . . 31
6.3.2.22 counter_nodeocov() . . . 31
6.3.2.23 counter_odegree() [1/2] . . . 32
6.3.2.24 counter_odegree() [2/2] . . . 32
6.3.2.25 counter_odegree15() [1/2] . . . 32
6.3.2.26 counter_odegree15() [2/2] . . . 32
6.3.2.27 counter_ones() . . . 32

6.3.2.28 counter_ostar2() [1/2] . . . 33
6.3.2.29 counter_ostar2() [2/2] . . . 33
6.3.2.30 counter_transition() . . . 33
6.3.2.31 counter_transition_formula() . . . 34
6.3.2.32 counter_ttriads() [1/2] . . . 34
6.3.2.33 counter_ttriads() [2/2] . . . 34
6.3.2.34 NETWORK_COUNTER() . . . 34
6.3.2.35 rules_dont_become_zero() . . . 35
6.3.2.36 rules_markov_fixed() . . . 35
6.4 Phylo rules . . . 35
6.4.1 Detailed Description . . . 37
6.4.2 Typedef Documentation . . . 37
6.4.2.1 DEFMCounter . . . 37
6.4.2.2 DEFMCounters . . . 37
6.4.2.3 DEFMModel . . . 38
6.4.2.4 DEFMRule . . . 38
6.4.2.5 DEFMRuleDyn . . . 38
6.4.2.6 DEFMRules . . . 38
6.4.2.7 DEFMRulesDyn . . . 38
6.4.2.8 DEFMStatsCounter . . . 38
6.4.2.9 DEFMSupport . . . 39
6.4.3 Function Documentation . . . 39
6.4.3.1 at() . . . 39
6.4.3.2 DEFMCounterData() [1/2] . . . 39
6.4.3.3 DEFMCounterData() [2/2] . . . 39
6.4.3.4 DEFMData() [1/2] . . . 39
6.4.3.5 DEFMData() [2/2] . . . 39
6.4.3.6 DEFMRuleData() [1/3] . . . 40
6.4.3.7 DEFMRuleData() [2/3] . . . 40
6.4.3.8 DEFMRuleData() [3/3] . . . 40
6.4.3.9 DEFMRuleDynData() . . . 40
6.4.3.10 idx() [1/2] . . . 41
6.4.3.11 idx() [2/2] . . . 41
6.4.3.12 is_true() [1/2] . . . 41
6.4.3.13 is_true() [2/2] . . . 41
6.4.3.14 ncol() . . . 41
6.4.3.15 nrow() . . . 41
6.4.3.16 num() [1/2] . . . 42
6.4.3.17 num() [2/2] . . . 42
6.4.3.18 operator()() . . . 42
6.4.3.19 print() . . . 42
6.4.3.20 rule_dyn_limit_changes() . . . 43

6.4.3.21 ~DEFMCounterData() . . . 43
6.4.3.22 ~DEFMData() . . . 43
6.4.3.23 ~DEFMRuleDynData() . . . 43
6.4.4 Variable Documentation . . . 44
6.4.4.1 array . . . 44
6.4.4.2 counts . . . 44
6.4.4.3 covar_sort . . . 44
6.4.4.4 covar_used . . . 44
6.4.4.5 covariates . . . 44
6.4.4.6 indices [1/2] . . . 45
6.4.4.7 indices [2/2] . . . 45
6.4.4.8 init . . . 45
6.4.4.9 is_motif . . . 45
6.4.4.10 logical [1/2] . . . 45
6.4.4.11 logical [2/2] . . . 45
6.4.4.12 numbers [1/2] . . . 46
6.4.4.13 numbers [2/2] . . . 46
6.4.4.14 obs_start . . . 46
6.4.4.15 X_ncol . . . 46
6.4.4.16 X_nrow . . . 46
6.5 Phylo counters . . . 47
6.5.1 Detailed Description . . . 48
6.5.2 Function Documentation . . . 48
6.5.2.1 counter_co_opt() . . . 48
6.5.2.2 counter_cogain() . . . 49
6.5.2.3 counter_gains() . . . 49
6.5.2.4 counter_gains_from_0() . . . 49
6.5.2.5 counter_gains_k_offspring() . . . 50
6.5.2.6 counter_genes_changing() . . . 50
6.5.2.7 counter_k_genes_changing() . . . 50
6.5.2.8 counter_less_than_p_prop_genes_changing() . . . 50
6.5.2.9 counter_longest() . . . 51
6.5.2.10 counter_loss() . . . 51
6.5.2.11 counter_maxfuns() . . . 51
6.5.2.12 counter_neofun() . . . 51
6.5.2.13 counter_neofun_a2b() . . . 52
6.5.2.14 counter_overall_changes() . . . 52
6.5.2.15 counter_overall_gains() . . . 52
6.5.2.16 counter_overall_gains_from_0() . . . 52
6.5.2.17 counter_overall_loss() . . . 53
6.5.2.18 counter_pairwise_first_gain() . . . 53
6.5.2.19 counter_pairwise_neofun_singlefun() . . . 53

6.5.2.20 counter_pairwise_overall_change() . . . 53
6.5.2.21 counter_pairwise_preserving() . . . 54
6.5.2.22 counter_preserve_pseudogene() . . . 54
6.5.2.23 counter_prop_genes_changing() . . . 54
6.5.2.24 counter_subfun() . . . 54

7 Namespace Documentation **55**

7.1 barry Namespace Reference . . . 55
7.1.1 Detailed Description . . . 55
7.2 barry::counters Namespace Reference . . . 55
7.2.1 Detailed Description . . . 55
7.3 barry::counters::network Namespace Reference . . . 56
7.4 CHECK Namespace Reference . . . 56
7.4.1 Detailed Description . . . 56
7.4.2 Variable Documentation . . . 56
7.4.2.1 BOTH . . . 56
7.4.2.2 NONE . . . 56
7.4.2.3 ONE . . . 56
7.4.2.4 TWO . . . 57
7.5 defm Namespace Reference . . . 57
7.6 EXISTS Namespace Reference . . . 57
7.6.1 Detailed Description . . . 57
7.6.2 Variable Documentation . . . 57
7.6.2.1 AS_ONE . . . 57
7.6.2.2 AS_ZERO . . . 57
7.6.2.3 BOTH . . . 58
7.6.2.4 NONE . . . 58
7.6.2.5 ONE . . . 58
7.6.2.6 TWO . . . 58
7.6.2.7 UKNOWN . . . 58
7.7 geese Namespace Reference . . . 58

8 Class Documentation **59**

8.1 BArray< Cell_Type, Data_Type > Class Template Reference . . . 59
8.1.1 Detailed Description . . . 61
8.1.2 Constructor & Destructor Documentation . . . 62
8.1.2.1 BArray() [1/6] . . . 62
8.1.2.2 BArray() [2/6] . . . 62
8.1.2.3 BArray() [3/6] . . . 62
8.1.2.4 BArray() [4/6] . . . 63
8.1.2.5 BArray() [5/6] . . . 63
8.1.2.6 BArray() [6/6] . . . 63
8.1.2.7 ~BArray() . . . 63

8.1.3 Member Function Documentation 63
8.1.3.1 clear() 63
8.1.3.2 col() 64
8.1.3.3 D() [1/2] 64
8.1.3.4 D() [2/2] 64
8.1.3.5 D_ptr() [1/2] 64
8.1.3.6 D_ptr() [2/2] 64
8.1.3.7 default_val() 64
8.1.3.8 flush_data() 64
8.1.3.9 get_cell() 65
8.1.3.10 get_col_vec() [1/2] 65
8.1.3.11 get_col_vec() [2/2] 65
8.1.3.12 get_entries() 65
8.1.3.13 get_row_vec() [1/2] 65
8.1.3.14 get_row_vec() [2/2] 66
8.1.3.15 insert_cell() [1/3] 66
8.1.3.16 insert_cell() [2/3] 66
8.1.3.17 insert_cell() [3/3] 66
8.1.3.18 is_dense() 66
8.1.3.19 is_empty() 67
8.1.3.20 ncol() 67
8.1.3.21 nnozero() 67
8.1.3.22 nrow() 67
8.1.3.23 operator()() [1/2] 67
8.1.3.24 operator()() [2/2] 67
8.1.3.25 operator*=() 68
8.1.3.26 operator+=() [1/3] 68
8.1.3.27 operator+=() [2/3] 68
8.1.3.28 operator+=() [3/3] 68
8.1.3.29 operator-=() [1/3] 68
8.1.3.30 operator-=() [2/3] 68
8.1.3.31 operator-=() [3/3] 69
8.1.3.32 operator/=() 69
8.1.3.33 operator=() [1/2] 69
8.1.3.34 operator=() [2/2] 69
8.1.3.35 operator==() 69
8.1.3.36 out_of_range() 69
8.1.3.37 print() 70
8.1.3.38 print_n() 70
8.1.3.39 reserve() 70
8.1.3.40 resize() 70
8.1.3.41 rm_cell() 70

8.1.3.42 row() . . . 71
8.1.3.43 set_data() . . . 71
8.1.3.44 swap_cells() . . . 71
8.1.3.45 swap_cols() . . . 71
8.1.3.46 swap_rows() . . . 72
8.1.3.47 toggle_cell() . . . 72
8.1.3.48 toggle_lock() . . . 72
8.1.3.49 transpose() . . . 72
8.1.3.50 zero_col() . . . 72
8.1.3.51 zero_row() . . . 73
8.1.4 Friends And Related Function Documentation . . . 73
8.1.4.1 BArrayCell< Cell_Type, Data_Type > . . . 73
8.1.4.2 BArrayCell_const< Cell_Type, Data_Type > . . . 73
8.1.5 Member Data Documentation . . . 73
8.1.5.1 visited . . . 73
8.2 BArrayCell< Cell_Type, Data_Type > Class Template Reference . . . 74
8.2.1 Detailed Description . . . 74
8.2.2 Constructor & Destructor Documentation . . . 74
8.2.2.1 BArrayCell() . . . 74
8.2.2.2 ~BArrayCell() . . . 74
8.2.3 Member Function Documentation . . . 75
8.2.3.1 operator Cell_Type() . . . 75
8.2.3.2 operator*=() . . . 75
8.2.3.3 operator+=() . . . 75
8.2.3.4 operator-=() . . . 75
8.2.3.5 operator/=() . . . 75
8.2.3.6 operator=() . . . 76
8.2.3.7 operator==() . . . 76
8.3 BArrayCell_const< Cell_Type, Data_Type > Class Template Reference . . . 76
8.3.1 Detailed Description . . . 76
8.3.2 Constructor & Destructor Documentation . . . 77
8.3.2.1 BArrayCell_const() . . . 77
8.3.2.2 ~BArrayCell_const() . . . 77
8.3.3 Member Function Documentation . . . 77
8.3.3.1 operator Cell_Type() . . . 77
8.3.3.2 operator"!=() . . . 77
8.3.3.3 operator<() . . . 78
8.3.3.4 operator<=() . . . 78
8.3.3.5 operator==() . . . 78
8.3.3.6 operator>() . . . 78
8.3.3.7 operator>=() . . . 78
8.4 BArrayDense< Cell_Type, Data_Type > Class Template Reference . . . 79

8.4.1 Detailed Description . . . 81
8.4.2 Constructor & Destructor Documentation . . . 82
8.4.2.1 BArrayDense() [1/6] . . . 82
8.4.2.2 BArrayDense() [2/6] . . . 82
8.4.2.3 BArrayDense() [3/6] . . . 82
8.4.2.4 BArrayDense() [4/6] . . . 83
8.4.2.5 BArrayDense() [5/6] . . . 83
8.4.2.6 BArrayDense() [6/6] . . . 83
8.4.2.7 ∼BArrayDense() . . . 83
8.4.3 Member Function Documentation . . . 83
8.4.3.1 clear() . . . 84
8.4.3.2 col() [1/2] . . . 84
8.4.3.3 col() [2/2] . . . 84
8.4.3.4 colsum() . . . 84
8.4.3.5 D() [1/2] . . . 84
8.4.3.6 D() [2/2] . . . 85
8.4.3.7 D_ptr() [1/2] . . . 85
8.4.3.8 D_ptr() [2/2] . . . 85
8.4.3.9 default_val() . . . 85
8.4.3.10 get_cell() . . . 85
8.4.3.11 get_col_vec() [1/2] . . . 86
8.4.3.12 get_col_vec() [2/2] . . . 86
8.4.3.13 get_data() . . . 86
8.4.3.14 get_entries() . . . 86
8.4.3.15 get_row_vec() [1/2] . . . 87
8.4.3.16 get_row_vec() [2/2] . . . 87
8.4.3.17 insert_cell() [1/2] . . . 87
8.4.3.18 insert_cell() [2/2] . . . 87
8.4.3.19 is_dense() . . . 88
8.4.3.20 is_empty() . . . 88
8.4.3.21 ncol() . . . 88
8.4.3.22 nnozero() . . . 88
8.4.3.23 nrow() . . . 88
8.4.3.24 operator()() [1/2] . . . 89
8.4.3.25 operator()() [2/2] . . . 89
8.4.3.26 operator∗=() . . . 89
8.4.3.27 operator+=() [1/3] . . . 89
8.4.3.28 operator+=() [2/3] . . . 89
8.4.3.29 operator+=() [3/3] . . . 90
8.4.3.30 operator-=() [1/3] . . . 90
8.4.3.31 operator-=() [2/3] . . . 90
8.4.3.32 operator-=() [3/3] . . . 90

8.4.3.33 operator/=() . . . 90
8.4.3.34 operator=() [1/2] . . . 91
8.4.3.35 operator=() [2/2] . . . 91
8.4.3.36 operator==() . . . 91
8.4.3.37 out_of_range() . . . 91
8.4.3.38 print() . . . 91
8.4.3.39 reserve() . . . 92
8.4.3.40 resize() . . . 92
8.4.3.41 rm_cell() . . . 92
8.4.3.42 row() [1/2] . . . 92
8.4.3.43 row() [2/2] . . . 92
8.4.3.44 rowsum() . . . 93
8.4.3.45 set_data() . . . 93
8.4.3.46 swap_cells() . . . 93
8.4.3.47 swap_cols() . . . 93
8.4.3.48 swap_rows() . . . 94
8.4.3.49 toggle_cell() . . . 94
8.4.3.50 toggle_lock() . . . 94
8.4.3.51 transpose() . . . 94
8.4.3.52 zero_col() . . . 94
8.4.3.53 zero_row() . . . 95
8.4.4 Friends And Related Function Documentation . . . 95
8.4.4.1 BArrayDenseCell< Cell_Type, Data_Type > . . . 95
8.4.4.2 BArrayDenseCol< Cell_Type, Data_Type > . . . 95
8.4.4.3 BArrayDenseCol_const< Cell_Type, Data_Type > . . . 95
8.4.4.4 BArrayDenseRow< Cell_Type, Data_Type > . . . 95
8.4.4.5 BArrayDenseRow_const< Cell_Type, Data_Type > . . . 96
8.4.5 Member Data Documentation . . . 96
8.4.5.1 visited . . . 96
8.5 BArrayDenseCell< Cell_Type, Data_Type > Class Template Reference . . . 96
8.5.1 Detailed Description . . . 97
8.5.2 Constructor & Destructor Documentation . . . 97
8.5.2.1 BArrayDenseCell() . . . 97
8.5.2.2 ~BArrayDenseCell() . . . 97
8.5.3 Member Function Documentation . . . 97
8.5.3.1 operator Cell_Type() . . . 97
8.5.3.2 operator*=() . . . 98
8.5.3.3 operator+=() . . . 98
8.5.3.4 operator-=() . . . 98
8.5.3.5 operator/=() . . . 98
8.5.3.6 operator=() [1/2] . . . 98
8.5.3.7 operator=() [2/2] . . . 99

8.5.3.8 operator==() . . . 99
8.5.4 Friends And Related Function Documentation . . . 99
8.5.4.1 BArrayDense< Cell_Type, Data_Type > . . . 99
8.5.4.2 BArrayDenseCol< Cell_Type, Data_Type > . . . 99
8.5.4.3 BArrayDenseCol_const< Cell_Type, Data_Type > . . . 99
8.6 BArrayDenseCell_const< Cell_Type, Data_Type > Class Template Reference . . . 100
8.6.1 Detailed Description . . . 100
8.7 BArrayDenseCol< Cell_Type, Data_Type > Class Template Reference . . . 100
8.7.1 Detailed Description . . . 100
8.7.2 Constructor & Destructor Documentation . . . 100
8.7.2.1 BArrayDenseCol() . . . 101
8.7.3 Member Function Documentation . . . 101
8.7.3.1 begin() . . . 101
8.7.3.2 end() . . . 101
8.7.3.3 operator()() . . . 101
8.7.3.4 size() . . . 101
8.7.4 Friends And Related Function Documentation . . . 102
8.7.4.1 BArrayDense< Cell_Type, Data_Type > . . . 102
8.7.4.2 BArrayDenseCell< Cell_Type, Data_Type > . . . 102
8.7.4.3 BArrayDenseCell_const< Cell_Type, Data_Type > . . . 102
8.8 BArrayDenseCol_const< Cell_Type, Data_Type > Class Template Reference . . . 102
8.8.1 Detailed Description . . . 103
8.8.2 Constructor & Destructor Documentation . . . 103
8.8.2.1 BArrayDenseCol_const() . . . 103
8.8.3 Member Function Documentation . . . 103
8.8.3.1 begin() . . . 103
8.8.3.2 end() . . . 103
8.8.3.3 operator()() . . . 104
8.8.3.4 size() . . . 104
8.8.4 Friends And Related Function Documentation . . . 104
8.8.4.1 BArrayDenseCell< Cell_Type, Data_Type > . . . 104
8.8.4.2 BArrayDenseCell_const< Cell_Type, Data_Type > . . . 104
8.9 BArrayDenseRow< Cell_Type, Data_Type > Class Template Reference . . . 104
8.9.1 Detailed Description . . . 105
8.9.2 Constructor & Destructor Documentation . . . 105
8.9.2.1 BArrayDenseRow() . . . 105
8.9.3 Member Function Documentation . . . 105
8.9.3.1 begin() . . . 105
8.9.3.2 end() . . . 106
8.9.3.3 operator()() . . . 106
8.9.3.4 size() . . . 106
8.9.4 Friends And Related Function Documentation . . . 106

8.9.4.1 BArrayDense< Cell_Type, Data_Type > . . . 106
8.9.4.2 BArrayDenseCell< Cell_Type, Data_Type > . . . 106
8.9.4.3 BArrayDenseCell_const< Cell_Type, Data_Type > . . . 107
8.10 BArrayDenseRow_const< Cell_Type, Data_Type > Class Template Reference . . . 107
8.10.1 Detailed Description . . . 107
8.10.2 Constructor & Destructor Documentation . . . 107
8.10.2.1 BArrayDenseRow_const() . . . 108
8.10.3 Member Function Documentation . . . 108
8.10.3.1 begin() . . . 108
8.10.3.2 end() . . . 108
8.10.3.3 operator()() . . . 108
8.10.3.4 size() . . . 108
8.10.4 Friends And Related Function Documentation . . . 109
8.10.4.1 BArrayDenseCell< Cell_Type, Data_Type > . . . 109
8.10.4.2 BArrayDenseCell_const< Cell_Type, Data_Type > . . . 109
8.11 BArrayRow< Cell_Type, Data_Type > Class Template Reference . . . 109
8.11.1 Detailed Description . . . 109
8.11.2 Constructor & Destructor Documentation . . . 110
8.11.2.1 BArrayRow() . . . 110
8.11.2.2 ~BArrayRow() . . . 110
8.11.3 Member Function Documentation . . . 110
8.11.3.1 operator BArrayRow< Cell_Type, Data_Type >() . . . 110
8.11.3.2 operator*=() . . . 110
8.11.3.3 operator+=() . . . 110
8.11.3.4 operator-=() . . . 111
8.11.3.5 operator/=() . . . 111
8.11.3.6 operator=() . . . 111
8.11.3.7 operator==() . . . 111
8.12 BArrayRow_const< Cell_Type, Data_Type > Class Template Reference . . . 111
8.12.1 Detailed Description . . . 112
8.12.2 Constructor & Destructor Documentation . . . 112
8.12.2.1 BArrayRow_const() . . . 112
8.12.2.2 ~BArrayRow_const() . . . 112
8.12.3 Member Function Documentation . . . 112
8.12.3.1 operator BArrayRow_const< Cell_Type, Data_Type >() . . . 112
8.12.3.2 operator"!=() . . . 112
8.12.3.3 operator<() . . . 113
8.12.3.4 operator<=() . . . 113
8.12.3.5 operator==() . . . 113
8.12.3.6 operator>() . . . 113
8.12.3.7 operator>=() . . . 113
8.13 BArrayVector< Cell_Type, Data_Type > Class Template Reference . . . 113

8.13.1 Detailed Description . . . 114
8.13.2 Constructor & Destructor Documentation . . . 114
8.13.2.1 BArrayVector() . . . 114
8.13.2.2 ~BArrayVector() . . . 115
8.13.3 Member Function Documentation . . . 115
8.13.3.1 begin() . . . 115
8.13.3.2 end() . . . 115
8.13.3.3 is_col() . . . 115
8.13.3.4 is_row() . . . 116
8.13.3.5 operator std::vector< Cell_Type >() . . . 116
8.13.3.6 operator*=() . . . 116
8.13.3.7 operator+=() . . . 116
8.13.3.8 operator-=() . . . 116
8.13.3.9 operator/=() . . . 117
8.13.3.10 operator=() . . . 117
8.13.3.11 operator==() . . . 117
8.13.3.12 size() . . . 117
8.14 BArrayVector_const< Cell_Type, Data_Type > Class Template Reference . . . 117
8.14.1 Detailed Description . . . 118
8.14.2 Constructor & Destructor Documentation . . . 118
8.14.2.1 BArrayVector_const() . . . 118
8.14.2.2 ~BArrayVector_const() . . . 118
8.14.3 Member Function Documentation . . . 119
8.14.3.1 begin() . . . 119
8.14.3.2 end() . . . 119
8.14.3.3 is_col() . . . 119
8.14.3.4 is_row() . . . 119
8.14.3.5 operator std::vector< Cell_Type >() . . . 119
8.14.3.6 operator"!=() . . . 120
8.14.3.7 operator<() . . . 120
8.14.3.8 operator<=() . . . 120
8.14.3.9 operator==() . . . 120
8.14.3.10 operator>() . . . 120
8.14.3.11 operator>=() . . . 121
8.14.3.12 size() . . . 121
8.15 Cell< Cell_Type > Class Template Reference . . . 121
8.15.1 Detailed Description . . . 122
8.15.2 Constructor & Destructor Documentation . . . 122
8.15.2.1 Cell() [1/7] . . . 122
8.15.2.2 Cell() [2/7] . . . 122
8.15.2.3 ~Cell() . . . 122
8.15.2.4 Cell() [3/7] . . . 123

8.15.2.5 Cell() [4/7] . . . 123
8.15.2.6 Cell() [5/7] . . . 123
8.15.2.7 Cell() [6/7] . . . 123
8.15.2.8 Cell() [7/7] . . . 123
8.15.3 Member Function Documentation . . . 123
8.15.3.1 add() [1/4] . . . 124
8.15.3.2 add() [2/4] . . . 124
8.15.3.3 add() [3/4] . . . 124
8.15.3.4 add() [4/4] . . . 124
8.15.3.5 operator Cell_Type() . . . 124
8.15.3.6 operator"!=() . . . 124
8.15.3.7 operator=() [1/2] . . . 125
8.15.3.8 operator=() [2/2] . . . 125
8.15.3.9 operator==() . . . 125
8.15.4 Member Data Documentation . . . 125
8.15.4.1 active . . . 125
8.15.4.2 value . . . 125
8.15.4.3 visited . . . 126
8.16 Cell_const< Cell_Type > Class Template Reference . . . 126
8.16.1 Detailed Description . . . 126
8.17 ConstBArrayRowIter< Cell_Type, Data_Type > Class Template Reference . . . 126
8.17.1 Detailed Description . . . 127
8.17.2 Constructor & Destructor Documentation . . . 127
8.17.2.1 ConstBArrayRowIter() . . . 127
8.17.2.2 ~ConstBArrayRowIter() . . . 127
8.17.3 Member Data Documentation . . . 127
8.17.3.1 Array . . . 128
8.17.3.2 current_col . . . 128
8.17.3.3 current_row . . . 128
8.17.3.4 iter . . . 128
8.18 Counter< Array_Type, Data_Type > Class Template Reference . . . 128
8.18.1 Detailed Description . . . 129
8.18.2 Constructor & Destructor Documentation . . . 130
8.18.2.1 Counter() [1/4] . . . 130
8.18.2.2 Counter() [2/4] . . . 130
8.18.2.3 Counter() [3/4] . . . 130
8.18.2.4 Counter() [4/4] . . . 130
8.18.2.5 ~Counter() . . . 131
8.18.3 Member Function Documentation . . . 131
8.18.3.1 count() . . . 131
8.18.3.2 get_description() . . . 131
8.18.3.3 get_hasher() . . . 131

8.18.3.4 get_name() . . . 131
8.18.3.5 init() . . . 131
8.18.3.6 operator=() [1/2] . . . 132
8.18.3.7 operator=() [2/2] . . . 132
8.18.3.8 set_hasher() . . . 132
8.18.4 Member Data Documentation . . . 132
8.18.4.1 count_fun . . . 132
8.18.4.2 data . . . 133
8.18.4.3 desc . . . 133
8.18.4.4 hasher_fun . . . 133
8.18.4.5 init_fun . . . 133
8.18.4.6 name . . . 133
8.19 Counters< Array_Type, Data_Type > Class Template Reference . . . 134
8.19.1 Detailed Description . . . 134
8.19.2 Constructor & Destructor Documentation . . . 134
8.19.2.1 Counters() [1/3] . . . 135
8.19.2.2 ~Counters() . . . 135
8.19.2.3 Counters() [2/3] . . . 135
8.19.2.4 Counters() [3/3] . . . 135
8.19.3 Member Function Documentation . . . 135
8.19.3.1 add_counter() [1/2] . . . 136
8.19.3.2 add_counter() [2/2] . . . 136
8.19.3.3 add_hash() . . . 136
8.19.3.4 gen_hash() . . . 136
8.19.3.5 get_descriptions() . . . 137
8.19.3.6 get_names() . . . 137
8.19.3.7 operator=() [1/2] . . . 137
8.19.3.8 operator=() [2/2] . . . 137
8.19.3.9 operator[]() . . . 138
8.19.3.10 size() . . . 138
8.20 DEFM Class Reference . . . 139
8.20.1 Detailed Description . . . 140
8.20.2 Constructor & Destructor Documentation . . . 140
8.20.2.1 DEFM() . . . 140
8.20.3 Member Function Documentation . . . 140
8.20.3.1 get_column_major() . . . 140
8.20.3.2 get_ID() . . . 140
8.20.3.3 get_m_order() . . . 141
8.20.3.4 get_model() . . . 141
8.20.3.5 get_n_covars() . . . 141
8.20.3.6 get_n_obs() . . . 141
8.20.3.7 get_n_rows() . . . 141

8.20.3.8 get_n_y() . . . 141
8.20.3.9 get_X() . . . 142
8.20.3.10 get_X_names() . . . 142
8.20.3.11 get_Y() . . . 142
8.20.3.12 get_Y_names() . . . 142
8.20.3.13 init() . . . 142
8.20.3.14 is_motif() . . . 142
8.20.3.15 logodds() . . . 143
8.20.3.16 motif_census() . . . 143
8.20.3.17 print() . . . 143
8.20.3.18 set_names() . . . 143
8.20.3.19 simulate() . . . 143
8.21 DEFMCounterData Class Reference . . . 144
8.21.1 Detailed Description . . . 144
8.22 DEFMData Class Reference . . . 144
8.22.1 Detailed Description . . . 145
8.23 DEFMRuleData Class Reference . . . 146
8.23.1 Detailed Description . . . 146
8.24 DEFMRuleDynData Class Reference . . . 147
8.24.1 Detailed Description . . . 147
8.25 Entries< Cell_Type > Class Template Reference . . . 148
8.25.1 Detailed Description . . . 148
8.25.2 Constructor & Destructor Documentation . . . 148
8.25.2.1 Entries() [1/2] . . . 148
8.25.2.2 Entries() [2/2] . . . 149
8.25.2.3 ~Entries() . . . 149
8.25.3 Member Function Documentation . . . 149
8.25.3.1 resize() . . . 149
8.25.4 Member Data Documentation . . . 149
8.25.4.1 source . . . 149
8.25.4.2 target . . . 149
8.25.4.3 val . . . 150
8.26 Flock Class Reference . . . 150
8.26.1 Detailed Description . . . 151
8.26.2 Constructor & Destructor Documentation . . . 151
8.26.2.1 Flock() . . . 151
8.26.2.2 ~Flock() . . . 151
8.26.3 Member Function Documentation . . . 151
8.26.3.1 add_data() . . . 151
8.26.3.2 colnames() . . . 152
8.26.3.3 get_counters() . . . 152
8.26.3.4 get_model() . . . 152

8.26.3.5 get_stats_support() . . . 152
8.26.3.6 get_stats_target() . . . 153
8.26.3.7 get_support_fun() . . . 153
8.26.3.8 init() . . . 153
8.26.3.9 likelihood_joint() . . . 153
8.26.3.10 nfuns() . . . 154
8.26.3.11 nleafs() . . . 154
8.26.3.12 nnodes() . . . 154
8.26.3.13 nterms() . . . 154
8.26.3.14 ntrees() . . . 154
8.26.3.15 operator()() . . . 154
8.26.3.16 parse_polytomies() . . . 155
8.26.3.17 print() . . . 155
8.26.3.18 set_seed() . . . 155
8.26.3.19 support_size() . . . 156
8.26.4 Member Data Documentation . . . 156
8.26.4.1 dat . . . 156
8.26.4.2 initialized . . . 156
8.26.4.3 model . . . 156
8.26.4.4 nfunctions . . . 156
8.26.4.5 rengine . . . 157
8.27 FreqTable< T > Class Template Reference . . . 157
8.27.1 Detailed Description . . . 157
8.27.2 Constructor & Destructor Documentation . . . 158
8.27.2.1 FreqTable() . . . 158
8.27.2.2 ~FreqTable() . . . 158
8.27.3 Member Function Documentation . . . 158
8.27.3.1 add() . . . 158
8.27.3.2 as_vector() . . . 158
8.27.3.3 clear() . . . 159
8.27.3.4 get_data() . . . 159
8.27.3.5 get_index() . . . 159
8.27.3.6 make_hash() . . . 159
8.27.3.7 print() . . . 159
8.27.3.8 reserve() . . . 160
8.27.3.9 size() . . . 160
8.28 Geese Class Reference . . . 160
8.28.1 Detailed Description . . . 163
8.28.2 Constructor & Destructor Documentation . . . 163
8.28.2.1 Geese() [1/4] . . . 164
8.28.2.2 Geese() [2/4] . . . 164
8.28.2.3 Geese() [3/4] . . . 164

8.28.2.4 Geese() [4/4] . . . 164
8.28.2.5 ~Geese() . . . 164
8.28.3 Member Function Documentation . . . 164
8.28.3.1 calc_reduced_sequence() . . . 165
8.28.3.2 calc_sequence() . . . 165
8.28.3.3 colnames() . . . 165
8.28.3.4 get_annotated_nodes() . . . 165
8.28.3.5 get_annotations() . . . 165
8.28.3.6 get_counters() . . . 166
8.28.3.7 get_model() . . . 166
8.28.3.8 get_probabilities() . . . 166
8.28.3.9 get_rengine() . . . 166
8.28.3.10 get_states() . . . 166
8.28.3.11 get_support_fun() . . . 167
8.28.3.12 inherit_support() . . . 167
8.28.3.13 init() . . . 167
8.28.3.14 init_node() . . . 167
8.28.3.15 likelihood() . . . 167
8.28.3.16 likelihood_exhaust() . . . 168
8.28.3.17 nannotations() . . . 168
8.28.3.18 nfuns() . . . 168
8.28.3.19 nleafs() . . . 168
8.28.3.20 nnodes() . . . 168
8.28.3.21 nterms() . . . 169
8.28.3.22 observed_counts() . . . 169
8.28.3.23 operator=() [1/2] . . . 169
8.28.3.24 operator=() [2/2] . . . 169
8.28.3.25 parse_polytomies() . . . 169
8.28.3.26 predict() . . . 170
8.28.3.27 predict_backend() . . . 170
8.28.3.28 predict_exhaust() . . . 170
8.28.3.29 predict_exhaust_backend() . . . 170
8.28.3.30 predict_sim() . . . 170
8.28.3.31 print() . . . 171
8.28.3.32 print_nodes() . . . 171
8.28.3.33 print_observed_counts() . . . 171
8.28.3.34 set_seed() . . . 171
8.28.3.35 simulate() . . . 171
8.28.3.36 support_size() . . . 171
8.28.3.37 update_annotations() . . . 172
8.28.4 Member Data Documentation . . . 172
8.28.4.1 delete_rengine . . . 172

8.28.4.2 delete_support . . . 172
8.28.4.3 etype_default . . . 172
8.28.4.4 etype_duplication . . . 172
8.28.4.5 etype_either . . . 173
8.28.4.6 etype_speciation . . . 173
8.28.4.7 initialized . . . 173
8.28.4.8 map_to_state_id . . . 173
8.28.4.9 nfunctions . . . 173
8.28.4.10 nodes . . . 173
8.28.4.11 pset_loc . . . 174
8.28.4.12 reduced_sequence . . . 174
8.28.4.13 sequence . . . 174
8.29 Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type > Class Template Reference . . . 175
8.29.1 Detailed Description . . . 179
8.29.2 Constructor & Destructor Documentation . . . 179
8.29.2.1 Model() [1/3] . . . 179
8.29.2.2 Model() [2/3] . . . 180
8.29.2.3 Model() [3/3] . . . 180
8.29.2.4 ~Model() . . . 180
8.29.3 Member Function Documentation . . . 180
8.29.3.1 add_array() . . . 180
8.29.3.2 add_counter() [1/2] . . . 181
8.29.3.3 add_counter() [2/2] . . . 181
8.29.3.4 add_hasher() . . . 181
8.29.3.5 add_rule() [1/2] . . . 181
8.29.3.6 add_rule() [2/2] . . . 182
8.29.3.7 add_rule_dyn() [1/2] . . . 182
8.29.3.8 add_rule_dyn() [2/2] . . . 182
8.29.3.9 colnames() . . . 182
8.29.3.10 conditional_prob() . . . 182
8.29.3.11 gen_key() . . . 183
8.29.3.12 get_arrays2support() . . . 183
8.29.3.13 get_counters() . . . 183
8.29.3.14 get_likelihoods() . . . 184
8.29.3.15 get_normalizing_constants() . . . 184
8.29.3.16 get_pset() . . . 184
8.29.3.17 get_pset_arrays() . . . 184
8.29.3.18 get_pset_probs() . . . 184
8.29.3.19 get_pset_stats() [1/2] . . . 185
8.29.3.20 get_pset_stats() [2/2] . . . 185
8.29.3.21 get_rengine() . . . 185

8.29.3.22 get_rules() . . . 185
8.29.3.23 get_rules_dyn() . . . 186
8.29.3.24 get_stats_support() . . . 186
8.29.3.25 get_stats_support_sizes() . . . 186
8.29.3.26 get_stats_support_sizes_acc() . . . 186
8.29.3.27 get_stats_target() . . . 187
8.29.3.28 get_support_fun() . . . 187
8.29.3.29 likelihood() [1/4] . . . 187
8.29.3.30 likelihood() [2/4] . . . 187
8.29.3.31 likelihood() [3/4] . . . 188
8.29.3.32 likelihood() [4/4] . . . 188
8.29.3.33 likelihood_total() . . . 188
8.29.3.34 nrules() . . . 188
8.29.3.35 nrules_dyn() . . . 189
8.29.3.36 nterms() . . . 189
8.29.3.37 operator=() . . . 189
8.29.3.38 print() . . . 189
8.29.3.39 print_stats() . . . 190
8.29.3.40 sample() [1/2] . . . 190
8.29.3.41 sample() [2/2] . . . 190
8.29.3.42 set_counters() . . . 190
8.29.3.43 set_rengine() . . . 191
8.29.3.44 set_rules() . . . 191
8.29.3.45 set_rules_dyn() . . . 191
8.29.3.46 set_seed() . . . 191
8.29.3.47 set_transform_model() . . . 191
8.29.3.48 size() . . . 192
8.29.3.49 size_unique() . . . 192
8.29.3.50 store_psets() . . . 192
8.29.3.51 support_size() . . . 192
8.29.3.52 transform_model() . . . 193
8.29.3.53 update_likelihoods() . . . 193
8.29.3.54 update_normalizing_constants() . . . 193
8.29.3.55 update_pset_probs() . . . 193
8.29.4 Member Data Documentation . . . 194
8.29.4.1 arrays2support . . . 194
8.29.4.2 counter_fun . . . 194
8.29.4.3 counters . . . 194
8.29.4.4 delete_counters . . . 194
8.29.4.5 delete_rengine . . . 195
8.29.4.6 delete_rules . . . 195
8.29.4.7 delete_rules_dyn . . . 195

8.29.4.8 first_calc_done . . . 195
8.29.4.9 keys2support . . . 195
8.29.4.10 normalizing_constants . . . 196
8.29.4.11 params_last . . . 196
8.29.4.12 pset_arrays . . . 196
8.29.4.13 pset_probs . . . 196
8.29.4.14 pset_stats . . . 197
8.29.4.15 rengine . . . 197
8.29.4.16 rules . . . 197
8.29.4.17 rules_dyn . . . 197
8.29.4.18 stats_likelihood . . . 197
8.29.4.19 stats_support . . . 198
8.29.4.20 stats_support_n_arrays . . . 198
8.29.4.21 stats_support_sizes . . . 198
8.29.4.22 stats_support_sizes_acc . . . 198
8.29.4.23 stats_target . . . 199
8.29.4.24 support_fun . . . 199
8.29.4.25 transform_model_fun . . . 199
8.29.4.26 transform_model_term_names . . . 200
8.29.4.27 with_pset . . . 200
8.30 NetCounterData Class Reference . . . 200
8.30.1 Detailed Description . . . 200
8.30.2 Constructor & Destructor Documentation . . . 201
8.30.2.1 NetCounterData() [1/2] . . . 201
8.30.2.2 NetCounterData() [2/2] . . . 201
8.30.2.3 ∼NetCounterData() . . . 201
8.30.3 Member Data Documentation . . . 201
8.30.3.1 indices . . . 201
8.30.3.2 numbers . . . 201
8.31 NetworkData Class Reference . . . 202
8.31.1 Detailed Description . . . 202
8.31.2 Constructor & Destructor Documentation . . . 202
8.31.2.1 NetworkData() [1/3] . . . 202
8.31.2.2 NetworkData() [2/3] . . . 202
8.31.2.3 NetworkData() [3/3] . . . 203
8.31.2.4 ∼NetworkData() . . . 203
8.31.3 Member Data Documentation . . . 203
8.31.3.1 directed . . . 203
8.31.3.2 vertex_attr . . . 204
8.32 Node Class Reference . . . 204
8.32.1 Detailed Description . . . 205
8.32.2 Constructor & Destructor Documentation . . . 205

8.32.2.1 Node() [1/5] . . . 205
8.32.2.2 Node() [2/5] . . . 206
8.32.2.3 Node() [3/5] . . . 206
8.32.2.4 Node() [4/5] . . . 206
8.32.2.5 Node() [5/5] . . . 206
8.32.2.6 ~Node() . . . 206
8.32.3 Member Function Documentation . . . 206
8.32.3.1 get_parent() . . . 207
8.32.3.2 is_leaf() . . . 207
8.32.3.3 noffspring() . . . 207
8.32.4 Member Data Documentation . . . 207
8.32.4.1 annotations . . . 207
8.32.4.2 array . . . 207
8.32.4.3 arrays . . . 208
8.32.4.4 duplication . . . 208
8.32.4.5 id . . . 208
8.32.4.6 narray . . . 208
8.32.4.7 offspring . . . 208
8.32.4.8 ord . . . 209
8.32.4.9 parent . . . 209
8.32.4.10 probability . . . 209
8.32.4.11 subtree_prob . . . 209
8.32.4.12 visited . . . 209
8.33 NodeData Class Reference . . . 210
8.33.1 Detailed Description . . . 210
8.33.2 Constructor & Destructor Documentation . . . 210
8.33.2.1 NodeData() . . . 210
8.33.3 Member Data Documentation . . . 210
8.33.3.1 blengths . . . 211
8.33.3.2 duplication . . . 211
8.33.3.3 states . . . 211
8.34 PhyloCounterData Class Reference . . . 211
8.34.1 Detailed Description . . . 212
8.34.2 Constructor & Destructor Documentation . . . 212
8.34.2.1 PhyloCounterData() [1/2] . . . 212
8.34.2.2 PhyloCounterData() [2/2] . . . 212
8.34.3 Member Function Documentation . . . 212
8.34.3.1 at() . . . 212
8.34.3.2 begin() . . . 212
8.34.3.3 empty() . . . 213
8.34.3.4 end() . . . 213
8.34.3.5 get_counters() . . . 213

8.34.3.6 operator()() . . . 213
8.34.3.7 operator[]() . . . 213
8.34.3.8 push_back() . . . 213
8.34.3.9 reserve() . . . 214
8.34.3.10 shrink_to_fit() . . . 214
8.34.3.11 size() . . . 214
8.35 PhyloRuleDynData Class Reference . . . 214
8.35.1 Detailed Description . . . 215
8.35.2 Constructor & Destructor Documentation . . . 215
8.35.2.1 PhyloRuleDynData() . . . 215
8.35.2.2 ∼PhyloRuleDynData() . . . 215
8.35.3 Member Function Documentation . . . 215
8.35.3.1 operator()() . . . 215
8.35.4 Member Data Documentation . . . 215
8.35.4.1 counts . . . 215
8.35.4.2 duplication . . . 216
8.35.4.3 lb . . . 216
8.35.4.4 pos . . . 216
8.35.4.5 ub . . . 216
8.36 PowerSet< Array_Type, Data_Rule_Type > Class Template Reference . . . 216
8.36.1 Detailed Description . . . 217
8.36.2 Constructor & Destructor Documentation . . . 218
8.36.2.1 PowerSet() [1/3] . . . 218
8.36.2.2 PowerSet() [2/3] . . . 218
8.36.2.3 PowerSet() [3/3] . . . 218
8.36.2.4 ∼PowerSet() . . . 218
8.36.3 Member Function Documentation . . . 218
8.36.3.1 add_rule() [1/2] . . . 219
8.36.3.2 add_rule() [2/2] . . . 219
8.36.3.3 begin() . . . 219
8.36.3.4 calc() . . . 219
8.36.3.5 end() . . . 219
8.36.3.6 get_data() . . . 220
8.36.3.7 get_data_ptr() . . . 220
8.36.3.8 init_support() . . . 220
8.36.3.9 operator[]() . . . 220
8.36.3.10 reset() . . . 220
8.36.3.11 size() . . . 221
8.36.4 Member Data Documentation . . . 221
8.36.4.1 coordinates_free . . . 221
8.36.4.2 coordinates_locked . . . 221
8.36.4.3 data . . . 221

8.36.4.4 EmptyArray . . . 221
8.36.4.5 M . . . 222
8.36.4.6 N . . . 222
8.36.4.7 n_free . . . 222
8.36.4.8 n_locked . . . 222
8.36.4.9 rules . . . 222
8.36.4.10 rules_deleted . . . 223
8.37 Progress Class Reference . . . 223
8.37.1 Detailed Description . . . 223
8.37.2 Constructor & Destructor Documentation . . . 223
8.37.2.1 Progress() . . . 223
8.37.2.2 ∼Progress() . . . 224
8.37.3 Member Function Documentation . . . 224
8.37.3.1 end() . . . 224
8.37.3.2 next() . . . 224
8.38 Rule< Array_Type, Data_Type > Class Template Reference . . . 224
8.38.1 Detailed Description . . . 225
8.38.2 Constructor & Destructor Documentation . . . 225
8.38.2.1 Rule() [1/2] . . . 225
8.38.2.2 Rule() [2/2] . . . 225
8.38.2.3 ∼Rule() . . . 226
8.38.3 Member Function Documentation . . . 226
8.38.3.1 D() . . . 226
8.38.3.2 get_description() [1/2] . . . 226
8.38.3.3 get_description() [2/2] . . . 226
8.38.3.4 get_name() [1/2] . . . 226
8.38.3.5 get_name() [2/2] . . . 227
8.38.3.6 operator()() . . . 227
8.39 Rules< Array_Type, Data_Type > Class Template Reference . . . 227
8.39.1 Detailed Description . . . 228
8.39.2 Constructor & Destructor Documentation . . . 228
8.39.2.1 Rules() [1/2] . . . 228
8.39.2.2 Rules() [2/2] . . . 228
8.39.2.3 ∼Rules() . . . 229
8.39.3 Member Function Documentation . . . 229
8.39.3.1 add_rule() [1/2] . . . 229
8.39.3.2 add_rule() [2/2] . . . 229
8.39.3.3 begin() . . . 229
8.39.3.4 end() . . . 229
8.39.3.5 get_descriptions() . . . 230
8.39.3.6 get_names() . . . 230
8.39.3.7 get_seq() . . . 230

8.39.3.8 operator()() . . . 230
8.39.3.9 operator=() . . . 231
8.39.3.10 size() . . . 231
8.40 StatsCounter< Array_Type, Data_Type > Class Template Reference . . . 231
8.40.1 Detailed Description . . . 232
8.40.2 Constructor & Destructor Documentation . . . 232
8.40.2.1 StatsCounter() [1/3] . . . 232
8.40.2.2 StatsCounter() [2/3] . . . 233
8.40.2.3 StatsCounter() [3/3] . . . 233
8.40.2.4 ~StatsCounter() . . . 233
8.40.3 Member Function Documentation . . . 233
8.40.3.1 add_counter() . . . 233
8.40.3.2 count_all() . . . 234
8.40.3.3 count_current() . . . 234
8.40.3.4 count_init() . . . 234
8.40.3.5 get_counters() . . . 234
8.40.3.6 get_descriptions() . . . 234
8.40.3.7 get_names() . . . 234
8.40.3.8 reset_array() . . . 234
8.40.3.9 set_counters() . . . 235
8.40.3.10 size() . . . 235
8.41 Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type > Class Template Reference . . . 235
8.41.1 Detailed Description . . . 237
8.41.2 Constructor & Destructor Documentation . . . 237
8.41.2.1 Support() [1/3] . . . 237
8.41.2.2 Support() [2/3] . . . 238
8.41.2.3 Support() [3/3] . . . 238
8.41.2.4 ~Support() . . . 238
8.41.3 Member Function Documentation . . . 238
8.41.3.1 add_counter() . . . 238
8.41.3.2 add_rule() [1/2] . . . 239
8.41.3.3 add_rule() [2/2] . . . 239
8.41.3.4 add_rule_dyn() [1/2] . . . 239
8.41.3.5 add_rule_dyn() [2/2] . . . 239
8.41.3.6 calc() . . . 239
8.41.3.7 eval_rules_dyn() . . . 240
8.41.3.8 get_counters() . . . 240
8.41.3.9 get_counts() . . . 240
8.41.3.10 get_current_stats() . . . 241
8.41.3.11 get_data() . . . 241
8.41.3.12 get_rules() . . . 241

8.41.3.13 get_rules_dyn() . . . 241
8.41.3.14 init_support() . . . 242
8.41.3.15 print() . . . 242
8.41.3.16 reset_array() [1/2] . . . 242
8.41.3.17 reset_array() [2/2] . . . 242
8.41.3.18 set_counters() . . . 243
8.41.3.19 set_rules() . . . 243
8.41.3.20 set_rules_dyn() . . . 243
8.41.4 Member Data Documentation . . . 243
8.41.4.1 change_stats . . . 243
8.41.4.2 coordiantes_n_free . . . 244
8.41.4.3 coordiantes_n_locked . . . 244
8.41.4.4 coordinates_free . . . 244
8.41.4.5 coordinates_locked . . . 244
8.41.4.6 current_stats . . . 244
8.41.4.7 delete_counters . . . 245
8.41.4.8 delete_rules . . . 245
8.41.4.9 delete_rules_dyn . . . 245
8.41.4.10 hashes . . . 245
8.41.4.11 hashes_initialized . . . 245
8.41.4.12 M . . . 246
8.41.4.13 max_num_elements . . . 246
8.41.4.14 N . . . 246
8.41.4.15 n_counters . . . 246
8.42 vecHasher< T > Struct Template Reference . . . 246
8.42.1 Detailed Description . . . 247
8.42.2 Member Function Documentation . . . 247
8.42.2.1 operator()() . . . 247

9 File Documentation **249**
9.1 include/barry/barray-bones.hpp File Reference . . . 249
9.2 include/barry/barray-iterator.hpp File Reference . . . 249
9.3 include/barry/barray-meat-operators.hpp File Reference . . . 250
9.3.1 Macro Definition Documentation . . . 250
9.3.1.1 BARRAY_TEMPLATE . . . 251
9.3.1.2 BARRAY_TEMPLATE_ARGS . . . 251
9.3.1.3 BARRAY_TYPE . . . 251
9.3.1.4 COL . . . 251
9.3.1.5 ROW . . . 251
9.3.2 Function Documentation . . . 251
9.3.2.1 BARRAY_TEMPLATE() [1/6] . . . 252
9.3.2.2 BARRAY_TEMPLATE() [2/6] . . . 252

9.3.2.3 BARRAY_TEMPLATE() [3/6] . . . 252
9.3.2.4 BARRAY_TEMPLATE() [4/6] . . . 252
9.3.2.5 BARRAY_TEMPLATE() [5/6] . . . 252
9.3.2.6 BARRAY_TEMPLATE() [6/6] . . . 253
9.3.2.7 BARRAY_TEMPLATE_ARGS() . . . 253
9.3.2.8 BARRAY_TYPE() . . . 253
9.3.2.9 for() . . . 253
9.3.2.10 operator()() . . . 253
9.3.3 Variable Documentation . . . 253
9.3.3.1 rhs . . . 254
9.3.3.2 this . . . 254
9.4 include/barry/barray-meat.hpp File Reference . . . 254
9.4.1 Macro Definition Documentation . . . 254
9.4.1.1 COL . . . 255
9.4.1.2 ROW . . . 255
9.5 include/barry/barraycell-bones.hpp File Reference . . . 255
9.6 include/barry/barraycell-meat.hpp File Reference . . . 256
9.7 include/barry/barraydense-bones.hpp File Reference . . . 256
9.8 include/barry/barraydense-meat-operators.hpp File Reference . . . 257
9.8.1 Macro Definition Documentation . . . 257
9.8.1.1 BDENSE_TEMPLATE . . . 257
9.8.1.2 BDENSE_TEMPLATE_ARGS . . . 258
9.8.1.3 BDENSE_TYPE . . . 258
9.8.1.4 COL . . . 258
9.8.1.5 POS . . . 258
9.8.1.6 POS_N . . . 258
9.8.1.7 ROW . . . 259
9.8.2 Function Documentation . . . 259
9.8.2.1 BDENSE_TEMPLATE() [1/4] . . . 259
9.8.2.2 BDENSE_TEMPLATE() [2/4] . . . 259
9.8.2.3 BDENSE_TEMPLATE() [3/4] . . . 259
9.8.2.4 BDENSE_TEMPLATE() [4/4] . . . 259
9.8.2.5 BDENSE_TEMPLATE_ARGS() . . . 260
9.8.2.6 BDENSE_TYPE() . . . 260
9.9 include/barry/barraydense-meat.hpp File Reference . . . 260
9.9.1 Macro Definition Documentation . . . 260
9.9.1.1 COL . . . 260
9.9.1.2 POS . . . 261
9.9.1.3 POS_N . . . 261
9.9.1.4 ROW . . . 261
9.9.1.5 ZERO_CELL . . . 261
9.10 include/barry/barraydensecell-bones.hpp File Reference . . . 261

9.10.1 Macro Definition Documentation . . . 262
9.10.1.1 POS . . . 262
9.11 include/barry/barraydensecell-meat.hpp File Reference . . . 262
9.11.1 Macro Definition Documentation . . . 262
9.11.1.1 POS . . . 263
9.12 include/barry/barraydensecol-bones.hpp File Reference . . . 263
9.12.1 Macro Definition Documentation . . . 263
9.12.1.1 POS . . . 263
9.12.1.2 POS_N . . . 264
9.12.1.3 ZERO_CELL . . . 264
9.13 include/barry/barraydenserow-bones.hpp File Reference . . . 264
9.13.1 Macro Definition Documentation . . . 265
9.13.1.1 POS . . . 265
9.13.1.2 POS_N . . . 265
9.13.1.3 ZERO_CELL . . . 265
9.14 include/barry/barrayrow-bones.hpp File Reference . . . 265
9.15 include/barry/barrayrow-meat.hpp File Reference . . . 265
9.15.1 Macro Definition Documentation . . . 266
9.15.1.1 BROW_TEMPLATE . . . 266
9.15.1.2 BROW_TEMPLATE_ARGS . . . 266
9.15.1.3 BROW_TYPE . . . 266
9.15.2 Function Documentation . . . 266
9.15.2.1 BROW_TEMPLATE() [1/5] . . . 266
9.15.2.2 BROW_TEMPLATE() [2/5] . . . 267
9.15.2.3 BROW_TEMPLATE() [3/5] . . . 267
9.15.2.4 BROW_TEMPLATE() [4/5] . . . 267
9.15.2.5 BROW_TEMPLATE() [5/5] . . . 267
9.16 include/barry/barrayvector-bones.hpp File Reference . . . 267
9.17 include/barry/barrayvector-meat.hpp File Reference . . . 268
9.18 include/barry/barry-configuration.hpp File Reference . . . 268
9.18.1 Macro Definition Documentation . . . 269
9.18.1.1 BARRY_CHECK_SUPPORT . . . 269
9.18.1.2 BARRY_ISFINITE . . . 269
9.18.1.3 BARRY_MAX_NUM_ELEMENTS . . . 269
9.18.1.4 BARRY_SAFE_EXP . . . 269
9.18.1.5 printf_barry . . . 269
9.18.2 Typedef Documentation . . . 270
9.18.2.1 Map . . . 270
9.19 include/barry/barry-debug.hpp File Reference . . . 270
9.19.1 Macro Definition Documentation . . . 270
9.19.1.1 BARRY_DEBUG_LEVEL . . . 270
9.20 include/barry/barry-macros.hpp File Reference . . . 271

9.20.1 Macro Definition Documentation . . . 271
9.20.1.1 BARRY_NCORES_ARG . . . 271
9.20.1.2 BARRY_ONE . . . 271
9.20.1.3 BARRY_ONE_DENSE . . . 272
9.20.1.4 BARRY_UNUSED . . . 272
9.20.1.5 BARRY_ZERO . . . 272
9.20.1.6 BARRY_ZERO_DENSE . . . 272
9.21 include/barry/barry.hpp File Reference . . . 272
9.21.1 Macro Definition Documentation . . . 274
9.21.1.1 BARRY_HPP . . . 274
9.21.1.2 BARRY_VERSION . . . 274
9.21.1.3 BARRY_VERSION_MAYOR . . . 274
9.21.1.4 BARRY_VERSION_MINOR . . . 274
9.21.1.5 COUNTER_FUNCTION . . . 274
9.21.1.6 COUNTER_LAMBDA . . . 275
9.21.1.7 RULE_FUNCTION . . . 275
9.21.1.8 RULE_LAMBDA . . . 275
9.22 include/barry/cell-bones.hpp File Reference . . . 275
9.23 include/barry/cell-meat.hpp File Reference . . . 276
9.24 include/barry/col-bones.hpp File Reference . . . 276
9.25 include/barry/counters-bones.hpp File Reference . . . 276
9.26 include/barry/counters-meat.hpp File Reference . . . 277
9.26.1 Macro Definition Documentation . . . 278
9.26.1.1 COUNTER_TEMPLATE . . . 278
9.26.1.2 COUNTER_TEMPLATE_ARGS . . . 279
9.26.1.3 COUNTER_TYPE . . . 279
9.26.1.4 COUNTERS_TEMPLATE . . . 279
9.26.1.5 COUNTERS_TEMPLATE_ARGS . . . 279
9.26.1.6 COUNTERS_TYPE . . . 279
9.26.1.7 TMP_HASHER_CALL . . . 279
9.26.2 Function Documentation . . . 280
9.26.2.1 count_fun() . . . 280
9.26.2.2 COUNTER_TEMPLATE() [1/9] . . . 280
9.26.2.3 COUNTER_TEMPLATE() [2/9] . . . 280
9.26.2.4 COUNTER_TEMPLATE() [3/9] . . . 280
9.26.2.5 COUNTER_TEMPLATE() [4/9] . . . 280
9.26.2.6 COUNTER_TEMPLATE() [5/9] . . . 281
9.26.2.7 COUNTER_TEMPLATE() [6/9] . . . 281
9.26.2.8 COUNTER_TEMPLATE() [7/9] . . . 281
9.26.2.9 COUNTER_TEMPLATE() [8/9] . . . 281
9.26.2.10 COUNTER_TEMPLATE() [9/9] . . . 281
9.26.2.11 COUNTERS_TEMPLATE() [1/9] . . . 282

9.26.2.12 COUNTERS_TEMPLATE() [2/9] . . . 282
9.26.2.13 COUNTERS_TEMPLATE() [3/9] . . . 282
9.26.2.14 COUNTERS_TEMPLATE() [4/9] . . . 282
9.26.2.15 COUNTERS_TEMPLATE() [5/9] . . . 282
9.26.2.16 COUNTERS_TEMPLATE() [6/9] . . . 282
9.26.2.17 COUNTERS_TEMPLATE() [7/9] . . . 283
9.26.2.18 COUNTERS_TEMPLATE() [8/9] . . . 283
9.26.2.19 COUNTERS_TEMPLATE() [9/9] . . . 283
9.26.2.20 data() . . . 283
9.26.2.21 desc() . . . 283
9.26.2.22 for() . . . 283
9.26.2.23 hasher() [1/2] . . . 284
9.26.2.24 hasher() [2/2] . . . 284
9.26.2.25 hasher_fun() [1/2] . . . 284
9.26.2.26 hasher_fun() [2/2] . . . 284
9.26.2.27 if() [1/3] . . . 284
9.26.2.28 if() [2/3] . . . 284
9.26.2.29 if() [3/3] . . . 285
9.26.2.30 init_fun() [1/3] . . . 285
9.26.2.31 init_fun() [2/3] . . . 285
9.26.2.32 init_fun() [3/3] . . . 285
9.26.2.33 name() . . . 285
9.26.3 Variable Documentation . . . 285
9.26.3.1 add_dims . . . 285
9.26.3.2 count_fun_ . . . 286
9.26.3.3 counter . . . 286
9.26.3.4 counter_ . . . 286
9.26.3.5 data_ . . . 286
9.26.3.6 desc_ . . . 287
9.26.3.7 fun . . . 287
9.26.3.8 fun_ . . . 287
9.26.3.9 hasher_fun_ . . . 287
9.26.3.10 i . . . 288
9.26.3.11 init_fun_ . . . 288
9.26.3.12 j . . . 288
9.26.3.13 name_ . . . 288
9.26.3.14 noexcept . . . 288
9.26.3.15 res . . . 289
9.26.3.16 return . . . 289
9.27 include/barry/counters/network-css.hpp File Reference . . . 289
9.27.1 Macro Definition Documentation . . . 290
9.27.1.1 CSS_APPEND . . . 291

9.27.1.2 CSS_CASE_ELSE . . . 291
9.27.1.3 CSS_CASE_PERCEIVED . . . 291
9.27.1.4 CSS_CASE_TRUTH . . . 291
9.27.1.5 CSS_CHECK_SIZE . . . 291
9.27.1.6 CSS_CHECK_SIZE_INIT . . . 292
9.27.1.7 CSS_MATCH_TYPE . . . 292
9.27.1.8 CSS_NET_COUNTER_LAMBDA_INIT . . . 292
9.27.1.9 CSS_PERCEIVED_CELLS . . . 292
9.27.1.10 CSS_SIZE . . . 293
9.27.1.11 CSS_TRUE_CELLS . . . 293
9.27.2 Function Documentation . . . 293
9.27.2.1 counter_css_census01() . . . 293
9.27.2.2 counter_css_census02() . . . 293
9.27.2.3 counter_css_census03() . . . 294
9.27.2.4 counter_css_census04() . . . 294
9.27.2.5 counter_css_census05() . . . 294
9.27.2.6 counter_css_census06() . . . 294
9.27.2.7 counter_css_census07() . . . 295
9.27.2.8 counter_css_census08() . . . 295
9.27.2.9 counter_css_census09() . . . 295
9.27.2.10 counter_css_census10() . . . 295
9.27.2.11 counter_css_completely_false_recip_comiss() . . . 296
9.27.2.12 counter_css_completely_false_recip_omiss() . . . 296
9.27.2.13 counter_css_mixed_recip() . . . 296
9.27.2.14 counter_css_partially_false_recip_commi() . . . 296
9.27.2.15 counter_css_partially_false_recip_omiss() . . . 297
9.28 include/barry/counters/network.hpp File Reference . . . 297
9.28.1 Macro Definition Documentation . . . 300
9.28.1.1 BARRY_ZERO_NETWORK . . . 300
9.28.1.2 BARRY_ZERO_NETWORK_DENSE . . . 301
9.28.1.3 NET_C_DATA_IDX . . . 301
9.28.1.4 NET_C_DATA_NUM . . . 301
9.28.1.5 NETWORK_COUNTER . . . 301
9.28.1.6 NETWORK_COUNTER_LAMBDA . . . 301
9.28.1.7 NETWORK_RULE . . . 302
9.28.1.8 NETWORK_RULE_LAMBDA . . . 302
9.28.1.9 NETWORKDENSE_COUNTER_LAMBDA . . . 302
9.28.2 Typedef Documentation . . . 302
9.28.2.1 NetCounter . . . 302
9.28.2.2 NetCounters . . . 303
9.28.2.3 NetModel . . . 303
9.28.2.4 NetRule . . . 303

9.28.2.5 NetRules . . . 303
9.28.2.6 NetStatsCounter . . . 303
9.28.2.7 NetSupport . . . 303
9.28.2.8 Network . . . 304
9.28.2.9 NetworkDense . . . 304
9.28.3 Function Documentation . . . 304
9.28.3.1 rules_zerodiag() . . . 304
9.29 include/barry/freqtable.hpp File Reference . . . 304
9.30 include/barry/model-bones.hpp File Reference . . . 305
9.31 include/barry/model-meat.hpp File Reference . . . 305
9.31.1 Function Documentation . . . 306
9.31.1.1 likelihood_() . . . 306
9.31.1.2 update_normalizing_constant() . . . 306
9.32 include/barry/models/defm.hpp File Reference . . . 306
9.33 include/barry/models/defm/counters.hpp File Reference . . . 307
9.33.1 Macro Definition Documentation . . . 308
9.33.1.1 DEFM_COUNTER . . . 308
9.33.1.2 DEFM_COUNTER_LAMBDA . . . 309
9.33.1.3 DEFM_RULE . . . 309
9.33.1.4 DEFM_RULE_LAMBDA . . . 309
9.33.1.5 DEFM_RULEDYN_LAMBDA . . . 310
9.33.1.6 UNI_SUB . . . 310
9.34 include/barry/models/geese/counters.hpp File Reference . . . 310
9.35 include/barry/models/defm/defm-bones.hpp File Reference . . . 312
9.36 include/barry/models/defm/defm-meat.hpp File Reference . . . 313
9.36.1 Macro Definition Documentation . . . 313
9.36.1.1 DEFM_LOOP_ARRAYS . . . 313
9.36.1.2 DEFM_RANGES . . . 313
9.36.2 Function Documentation . . . 314
9.36.2.1 keygen_defm() . . . 314
9.37 include/barry/models/defm/defm-types.hpp File Reference . . . 314
9.37.1 Typedef Documentation . . . 315
9.37.1.1 DEFMArray . . . 315
9.38 include/barry/models/defm/formula.hpp File Reference . . . 315
9.38.1 Function Documentation . . . 315
9.38.1.1 defm_motif_parser() . . . 316
9.39 include/barry/models/geese.hpp File Reference . . . 317
9.40 include/barry/models/geese/flock-bones.hpp File Reference . . . 317
9.41 include/barry/models/geese/flock-meat.hpp File Reference . . . 318
9.42 include/barry/models/geese/geese-bones.hpp File Reference . . . 318
9.42.1 Macro Definition Documentation . . . 319
9.42.1.1 INITIALIZED . . . 319

9.42.2 Function Documentation . . . 319
9.42.2.1 keygen_full() . . . 319
9.42.2.2 RULE_FUNCTION() . . . 319
9.42.2.3 vec_diff() . . . 320
9.42.2.4 vector_caster() . . . 320
9.43 include/barry/models/geese/geese-meat-constructors.hpp File Reference . . . 320
9.44 include/barry/models/geese/geese-meat-likelihood.hpp File Reference . . . 321
9.44.1 Function Documentation . . . 321
9.44.1.1 pset_loop() . . . 322
9.45 include/barry/models/geese/geese-meat-likelihood_exhaust.hpp File Reference . . . 322
9.46 include/barry/models/geese/geese-meat-predict.hpp File Reference . . . 323
9.47 include/barry/models/geese/geese-meat-predict_exhaust.hpp File Reference . . . 323
9.48 include/barry/models/geese/geese-meat-predict_sim.hpp File Reference . . . 324
9.49 include/barry/models/geese/geese-meat-simulate.hpp File Reference . . . 324
9.50 include/barry/models/geese/geese-meat.hpp File Reference . . . 325
9.51 include/barry/models/geese/geese-node-bones.hpp File Reference . . . 325
9.52 include/barry/models/geese/geese-types.hpp File Reference . . . 326
9.52.1 Macro Definition Documentation . . . 326
9.52.1.1 POS . . . 327
9.52.2 Typedef Documentation . . . 327
9.52.2.1 PhyloArray . . . 327
9.52.2.2 PhyloCounter . . . 327
9.52.2.3 PhyloCounters . . . 327
9.52.2.4 PhyloModel . . . 327
9.52.2.5 PhyloPowerSet . . . 328
9.52.2.6 PhyloRule . . . 328
9.52.2.7 PhyloRuleData . . . 328
9.52.2.8 PhyloRuleDyn . . . 328
9.52.2.9 PhyloRules . . . 328
9.52.2.10 PhyloRulesDyn . . . 328
9.52.2.11 PhyloStatsCounter . . . 329
9.52.2.12 PhyloSupport . . . 329
9.53 include/barry/powerset-bones.hpp File Reference . . . 329
9.54 include/barry/powerset-meat.hpp File Reference . . . 330
9.55 include/barry/progress.hpp File Reference . . . 330
9.55.1 Macro Definition Documentation . . . 331
9.55.1.1 BARRY_PROGRESS_BAR_WIDTH . . . 331
9.56 include/barry/rules-bones.hpp File Reference . . . 331
9.56.1 Function Documentation . . . 331
9.56.1.1 rule_fun_default() . . . 332
9.57 include/barry/rules-meat.hpp File Reference . . . 332
9.58 include/barry/statscounter-bones.hpp File Reference . . . 332

9.59 include/barry/statscounter-meat.hpp File Reference . . . 333
9.59.1 Macro Definition Documentation . . . 334
9.59.1.1 STATSCOUNTER_TEMPLATE . . . 334
9.59.1.2 STATSCOUNTER_TEMPLATE_ARGS . . . 334
9.59.1.3 STATSCOUNTER_TYPE . . . 334
9.59.2 Function Documentation . . . 334
9.59.2.1 clear() . . . 334
9.59.2.2 for() . . . 335
9.59.2.3 resize() . . . 335
9.59.2.4 STATSCOUNTER_TEMPLATE() [1/9] . . . 335
9.59.2.5 STATSCOUNTER_TEMPLATE() [2/9] . . . 335
9.59.2.6 STATSCOUNTER_TEMPLATE() [3/9] . . . 335
9.59.2.7 STATSCOUNTER_TEMPLATE() [4/9] . . . 335
9.59.2.8 STATSCOUNTER_TEMPLATE() [5/9] . . . 336
9.59.2.9 STATSCOUNTER_TEMPLATE() [6/9] . . . 336
9.59.2.10 STATSCOUNTER_TEMPLATE() [7/9] . . . 336
9.59.2.11 STATSCOUNTER_TEMPLATE() [8/9] . . . 336
9.59.2.12 STATSCOUNTER_TEMPLATE() [9/9] . . . 336
9.59.3 Variable Documentation . . . 336
9.59.3.1 counter . . . 337
9.59.3.2 counter_deleted . . . 337
9.59.3.3 counters . . . 337
9.59.3.4 counters_ . . . 337
9.59.3.5 current_stats . . . 337
9.59.3.6 EmptyArray . . . 338
9.59.3.7 f_ . . . 338
9.59.3.8 j . . . 338
9.59.3.9 return . . . 338
9.60 include/barry/support-bones.hpp File Reference . . . 338
9.61 include/barry/support-meat.hpp File Reference . . . 339
9.61.1 Macro Definition Documentation . . . 339
9.61.1.1 BARRY_SUPPORT_MEAT_HPP . . . 339
9.62 include/barry/typedefs.hpp File Reference . . . 340
9.62.1 Typedef Documentation . . . 342
9.62.1.1 Col_type . . . 342
9.62.1.2 Counter_fun_type . . . 342
9.62.1.3 Counts_type . . . 342
9.62.1.4 Hasher_fun_type . . . 342
9.62.1.5 MapVec_type . . . 343
9.62.1.6 Row_type . . . 343
9.62.1.7 Rule_fun_type . . . 343
9.62.2 Function Documentation . . . 343

9.62.2.1 sort_array() . . . 343
9.62.2.2 vec_equal() . . . 344
9.62.2.3 vec_equal_approx() . . . 344
9.62.2.4 vec_inner_prod() [1/2] . . . 345
9.62.2.5 vec_inner_prod() [2/2] . . . 345
9.63 README.md File Reference . . . 345

Index **347**

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. Its primary goal is to provide a general framework for building discrete exponential-family models. A particular example is Exponential Random Graph Models (ERGMs), but we can use barry to deal with non-square arrays.

Among the key features included in barry, we have:

- Sparse arrays.
- User-defined count statistics.
- User-defined constrain of the support set.
- Powerset generation of binary arrays.
- Discrete Exponential Family Models module (DEFMs).
- Pooled DEFMs.

To use barry, you can either download the entire repository or, since it is header-only, the single header version barry.hpp.

This library was created and maintained by Dr. George G. Vega Yon as part of his doctoral dissertation "Essays on Bioinformatics and Social Network Analysis: Statistical and Computational Methods for Complex Systems."

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?
  net.print("Current view");

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

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

  // 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
```

Features

Efficient memory usage

One of the key features of barry is that it will handle memory efficiently. In the case of pooled-data models, the module for statistical models avoids double-counting support when possible by keeping track of what datasets (networks, for instance) share the same.

Documentation

More information can be found in the Doxygen website here and in the PDF version of the documentation here.

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 13
Statistical Models 24
Network counters 24
Phylo counters 47
    Phylo rules 35

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

BArray< Cell_Type, Data_Type > . . . 59
BArray< bool, bool > . . . 59
BArray< Cell_Type, Data_Type > . . . 59
BArrayCell< Cell_Type, Data_Type > . . . 74
BArrayCell_const< Cell_Type, Data_Type > . . . 76
BArrayDense< Cell_Type, Data_Type > . . . 79
BArrayDense< bool, bool > . . . 79
BArrayDenseCell< Cell_Type, Data_Type > . . . 96
BArrayDenseCell_const< Cell_Type, Data_Type > . . . 100
BArrayDenseCol< Cell_Type, Data_Type > . . . 100
BArrayDenseCol_const< Cell_Type, Data_Type > . . . 102
BArrayDenseRow< Cell_Type, Data_Type > . . . 104
BArrayDenseRow_const< Cell_Type, Data_Type > . . . 107
BArrayRow< Cell_Type, Data_Type > . . . 109
BArrayRow_const< Cell_Type, Data_Type > . . . 111
BArrayVector< Cell_Type, Data_Type > . . . 113
BArrayVector_const< Cell_Type, Data_Type > . . . 117
Cell< Cell_Type > . . . 121
Cell< bool > . . . 121
Cell_const< Cell_Type > . . . 126
ConstBArrayRowIter< Cell_Type, Data_Type > . . . 126
Counter< Array_Type, Data_Type > . . . 128
Counters< Array_Type, Data_Type > . . . 134
Counters< Array_Type, Data_Type > . . . 134
Counters< BArray< bool, bool >, bool > . . . 134
Counters< BArray<>, bool > . . . 134
DEFMCounterData . . . 144
DEFMData . . . 144
DEFMModel
  DEFM . . . 139
DEFMRuleData . . . 146
  DEFMRuleDynData . . . 147
Entries< Cell_Type > . . . 148
Flock . . . 150
FreqTable< T > . . . 157

Geese 160
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type > 175
NetCounterData 200
NetworkData 202
Node 204
NodeData 210
PhyloCounterData 211
PhyloRuleDynData 214
PowerSet< Array_Type, Data_Rule_Type > 216
Progress 223
Rule< Array_Type, Data_Type > 224
Rules< Array_Type, Data_Type > 227
Rules< BArray< bool, bool >, bool > 227
Rules< BArray<>, bool > 227
StatsCounter< Array_Type, Data_Type > 231
StatsCounter< BArray<>, bool > 231
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type > 235
Support< BArray<>, bool, bool, bool > 235
vecHasher< T > 246

Chapter 4

Class Index

4.1 Class List

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

BArray< Cell_Type, Data_Type >
: Baseline class for binary arrays . . . 59
BArrayCell< Cell_Type, Data_Type > . . . 74
BArrayCell_const< Cell_Type, Data_Type > . . . 76
BArrayDense< Cell_Type, Data_Type >
: Baseline class for binary arrays . . . 79
BArrayDenseCell< Cell_Type, Data_Type > . . . 96
BArrayDenseCell_const< Cell_Type, Data_Type > . . . 100
BArrayDenseCol< Cell_Type, Data_Type > . . . 100
BArrayDenseCol_const< Cell_Type, Data_Type > . . . 102
BArrayDenseRow< Cell_Type, Data_Type > . . . 104
BArrayDenseRow_const< Cell_Type, Data_Type > . . . 107
BArrayRow< Cell_Type, Data_Type > . . . 109
BArrayRow_const< Cell_Type, Data_Type > . . . 111
BArrayVector< Cell_Type, Data_Type >
: Row or column of a `BArray` . . . 113
BArrayVector_const< Cell_Type, Data_Type > . . . 117
Cell< Cell_Type >
: Entries in BArray. For now, it only has two members: . . . 121
Cell_const< Cell_Type > . . . 126
ConstBArrayRowIter< Cell_Type, Data_Type > . . . 126
Counter< Array_Type, Data_Type >
: A counter function based on change statistics . . . 128
Counters< Array_Type, Data_Type >
: Vector of counters . . . 134
DEFM . . . 139
DEFMCounterData
: Data class used to store arbitrary size_t or double vectors . . . 144
DEFMData
: Data class for DEFM arrays . . . 144
DEFMRuleData . . . 146
DEFMRuleDynData . . . 147
Entries< Cell_Type >
: A wrapper class to store `source`, `target`, `val` from a `BArray` object . . . 148
Flock
: A Flock is a group of Geese . . . 150

FreqTable< T >
Frequency table of vectors . . . 157
Geese
Annotated Phylo Model . . . 160
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: . . . 175
NetCounterData
Data class used to store arbitrary size_t or double vectors . . . 200
NetworkData
Data class for Networks . . . 202
Node
A single node for the model . . . 204
NodeData
Data definition for the PhyloArray class . . . 210
PhyloCounterData . . . 211
PhyloRuleDynData . . . 214
PowerSet< Array_Type, Data_Rule_Type >
Powerset of a binary array . . . 216
Progress
A simple progress bar . . . 223
Rule< Array_Type, Data_Type >
Rule for determining if a cell should be included in a sequence . . . 224
Rules< Array_Type, Data_Type >
Vector of objects of class Rule . . . 227
StatsCounter< Array_Type, Data_Type >
Count stats for a single Array . . . 231
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >
Compute the support of sufficient statistics . . . 235
vecHasher< T > . . . 246

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

include/barry/barray-bones.hpp . . . 249
include/barry/barray-iterator.hpp . . . 249
include/barry/barray-meat-operators.hpp . . . 250
include/barry/barray-meat.hpp . . . 254
include/barry/barraycell-bones.hpp . . . 255
include/barry/barraycell-meat.hpp . . . 256
include/barry/barraydense-bones.hpp . . . 256
include/barry/barraydense-meat-operators.hpp . . . 257
include/barry/barraydense-meat.hpp . . . 260
include/barry/barraydensecell-bones.hpp . . . 261
include/barry/barraydensecell-meat.hpp . . . 262
include/barry/barraydensecol-bones.hpp . . . 263
include/barry/barraydenserow-bones.hpp . . . 264
include/barry/barrayrow-bones.hpp . . . 265
include/barry/barrayrow-meat.hpp . . . 265
include/barry/barrayvector-bones.hpp . . . 267
include/barry/barrayvector-meat.hpp . . . 268
include/barry/barry-configuration.hpp . . . 268
include/barry/barry-debug.hpp . . . 270
include/barry/barry-macros.hpp . . . 271
include/barry/barry.hpp . . . 272
include/barry/cell-bones.hpp . . . 275
include/barry/cell-meat.hpp . . . 276
include/barry/col-bones.hpp . . . 276
include/barry/counters-bones.hpp . . . 276
include/barry/counters-meat.hpp . . . 277
include/barry/freqtable.hpp . . . 304
include/barry/model-bones.hpp . . . 305
include/barry/model-meat.hpp . . . 305
include/barry/powerset-bones.hpp . . . 329
include/barry/powerset-meat.hpp . . . 330
include/barry/progress.hpp . . . 330
include/barry/rules-bones.hpp . . . 331
include/barry/rules-meat.hpp . . . 332
include/barry/statscounter-bones.hpp . . . 332

include/barry/statscounter-meat.hpp . . . 333
include/barry/support-bones.hpp . . . 338
include/barry/support-meat.hpp . . . 339
include/barry/typedefs.hpp . . . 340
include/barry/counters/network-css.hpp . . . 289
include/barry/counters/network.hpp . . . 297
include/barry/models/defm.hpp . . . 306
include/barry/models/geese.hpp . . . 317
include/barry/models/defm/counters.hpp . . . 307
include/barry/models/defm/defm-bones.hpp . . . 312
include/barry/models/defm/defm-meat.hpp . . . 313
include/barry/models/defm/defm-types.hpp . . . 314
include/barry/models/defm/formula.hpp . . . 315
include/barry/models/geese/counters.hpp . . . 310
include/barry/models/geese/flock-bones.hpp . . . 317
include/barry/models/geese/flock-meat.hpp . . . 318
include/barry/models/geese/geese-bones.hpp . . . 318
include/barry/models/geese/geese-meat-constructors.hpp . . . 320
include/barry/models/geese/geese-meat-likelihood.hpp . . . 321
include/barry/models/geese/geese-meat-likelihood_exhaust.hpp . . . 322
include/barry/models/geese/geese-meat-predict.hpp . . . 323
include/barry/models/geese/geese-meat-predict_exhaust.hpp . . . 323
include/barry/models/geese/geese-meat-predict_sim.hpp . . . 324
include/barry/models/geese/geese-meat-simulate.hpp . . . 324
include/barry/models/geese/geese-meat.hpp . . . 325
include/barry/models/geese/geese-node-bones.hpp . . . 325
include/barry/models/geese/geese-types.hpp . . . 326

Chapter 6

Module Documentation

6.1 Counting

Classes

- class NetworkData

  Data class for Networks.
- class Counter< Array_Type, Data_Type >

  A counter function based on change statistics.

Macros

- #define MAKE_DEFM_HASHER(hasher, a, cov)

- #define MAKE_DUPL_VARS()
- #define IS_EITHER() (DATA_AT == Geese::etype_either)
- #define IS_DUPLICATION() ((DATA_AT == Geese::etype_duplication) & (DPL))
- #define IS_SPECIATION() ((DATA_AT == Geese::etype_speciation) & (!DPL))
- #define IF_MATCHES()
- #define IF_NOTMATCHES()
- #define PHYLO_COUNTER_LAMBDA(a)

  Extension of a simple counter.
- #define PHYLO_RULE_DYN_LAMBDA(a)
- #define PHYLO_CHECK_MISSING()
- std::string get_last_name (size_t d)
- void counter_overall_gains (PhyloCounters *counters, size_t duplication=Geese::etype_default)

  Overall functional gains.
- void counter_gains (PhyloCounters *counters, std::vector< size_t > nfun, size_t duplication=Geese::etype_default)

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

  k genes gain function nfun
- void counter_genes_changing (PhyloCounters *counters, size_t duplication=Geese::etype_default)

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

- void counter_preserve_pseudogene (PhyloCounters *counters, size_t nfunA, size_t nfunB, size_t duplication=Geese::etype_default)

  Keeps track of how many pairs of genes preserve pseudostate.
- void counter_prop_genes_changing (PhyloCounters *counters, size_t duplication=Geese::etype_default)

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

  Overall functional loss.
- void counter_maxfuns (PhyloCounters *counters, size_t lb, size_t ub, size_t duplication=Geese::etype_default)

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

  Total count of losses for an specific function.
- void counter_overall_changes (PhyloCounters *counters, size_t duplication=Geese::etype_default)

  Total number of changes. Use this statistic to account for "preservation".
- void counter_subfun (PhyloCounters *counters, size_t nfunA, size_t nfunB, size_t duplication=Geese::etype_default)

  Total count of Sub-functionalization events.
- void counter_cogain (PhyloCounters *counters, size_t nfunA, size_t nfunB, size_t duplication=Geese::etype_default)

  Co-evolution (joint gain or loss)
- void counter_longest (PhyloCounters *counters, size_t duplication=Geese::etype_default)

  Longest branch mutates (either by gain or by loss)
- void counter_neofun (PhyloCounters *counters, size_t nfunA, size_t nfunB, size_t duplication=Geese::etype_default)

  Total number of neofunctionalization events.
- void counter_pairwise_neofun_singlefun (PhyloCounters *counters, size_t nfunA, size_t duplication=Geese::etype_default)

  Total number of neofunctionalization events sum_u sum_{w < u} [x(u,a)(1 - x(w,a)) + (1 - x(u,a)) * x(w,a)] change stat: delta{x(u,a): 0->1} = 1 - 2 * x(w,a)*
- void counter_neofun_a2b (PhyloCounters *counters, size_t nfunA, size_t nfunB, size_t duplication=Geese::etype_default)

  Total number of neofunctionalization events.
- void counter_co_opt (PhyloCounters *counters, size_t nfunA, size_t nfunB, size_t duplication=Geese::etype_default)

  Function co-opting.
- void counter_k_genes_changing (PhyloCounters *counters, size_t k, size_t duplication=Geese::etype_default)

  Indicator function. Equals to one if k genes changed and zero otherwise.
- void counter_less_than_p_prop_genes_changing (PhyloCounters *counters, double p, size_t duplication=Geese::etype_default

  Indicator function. Equals to one if k genes changed and zero otherwise.
- void counter_gains_from_0 (PhyloCounters *counters, std::vector< size_t > nfun, size_t duplication=Geese::etype_default)

  Used when all the functions are in 0 (like the root node prob.)
- void counter_overall_gains_from_0 (PhyloCounters *counters, size_t duplication=Geese::etype_default)

  Used when all the functions are in 0 (like the root node prob.)
- void counter_pairwise_overall_change (PhyloCounters *counters, size_t duplication=Geese::etype_default)

  Used when all the functions are in 0 (like the root node prob.)
- void counter_pairwise_preserving (PhyloCounters *counters, size_t nfunA, size_t nfunB, size_t duplication=Geese::etype_default)

  Used when all the functions are in 0 (like the root node prob.)
- void counter_pairwise_first_gain (PhyloCounters *counters, size_t nfunA, size_t nfunB, size_t duplication=Geese::etype_default)

  Used when all the functions are in 0 (like the root node prob.)

6.1.1 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.

6.1.2 Macro Definition Documentation

6.1.2.1 IF_MATCHES

```
#define IF_MATCHES( )
```

Value:

```
    MAKE_DUPL_VARS() \
    if (IS_EITHER() || IS_DUPLICATION() || IS_SPECIATION())
```

Definition at line 20 of file counters.hpp.

6.1.2.2 IF_NOTMATCHES

```
#define IF_NOTMATCHES( )
```

Value:

```
    MAKE_DUPL_VARS() \
    if (!IS_EITHER() && !IS_DUPLICATION() && !IS_SPECIATION())
```

Definition at line 22 of file counters.hpp.

6.1.2.3 IS_DUPLICATION

```
#define IS_DUPLICATION( ) ((DATA_AT == Geese::etype_duplication) & (DPL))
```

Definition at line 17 of file counters.hpp.

6.1.2.4 IS_EITHER

```
#define IS_EITHER( ) (DATA_AT == Geese::etype_either)
```

Definition at line 16 of file counters.hpp.

6.1.2.5 IS_SPECIATION

```
#define IS_SPECIATION( ) ((DATA_AT == Geese::etype_speciation) & (!DPL))
```

Definition at line 18 of file counters.hpp.

6.1.2.6 MAKE_DEFM_HASHER

```
#define MAKE_DEFM_HASHER(
                hasher,
                a,
                cov )
```

Value:

```
    barry::Hasher_fun_type<DEFMArray,DEFMCounterData> \
    hasher = [cov](const DEFMArray & array, DEFMCounterData * d) { \
        std::vector< double > res; \
        /* Adding the column feature */ \
        for (size_t i = 0u; i < array.nrow(); ++i) \
            res.push_back(array.D()(i, cov)); \
        /* Adding the fixed dims */ \
        for (size_t i = 0u; i < (array.nrow() - 1); ++i) \
            for (size_t j = 0u; j < array.ncol(); ++j) \
                res.push_back(array(i, j)); \
        return res;\
    };
```

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

Definition at line 21 of file counters.hpp.

6.1.2.7 MAKE_DUPL_VARS

```
#define MAKE_DUPL_VARS( )
```

Value:

```
    bool DPL = Array.D_ptr()->duplication; \
    size_t DATA_AT = data[0u];
```

Details about the available counters for `PhyloArray` objects can be found in the Phylo counters section.

Definition at line 12 of file counters.hpp.

6.1.2.8 PHYLO_CHECK_MISSING

```
#define PHYLO_CHECK_MISSING( )
```

Value:

```
    if (Array.D_ptr() == nullptr) \
    throw std::logic_error("The array data is nullptr."); \
```

Definition at line 42 of file counters.hpp.

6.1.2.9 PHYLO_COUNTER_LAMBDA

```
#define PHYLO_COUNTER_LAMBDA(
                a )
```

Value:

```
    barry::Counter_fun_type<PhyloArray, PhyloCounterData> a = \
    [](const PhyloArray & Array, size_t i, size_t 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 36 of file counters.hpp.

6.1.2.10 PHYLO_RULE_DYN_LAMBDA

```
#define PHYLO_RULE_DYN_LAMBDA(
                a )
```

Value:

```
    barry::Rule_fun_type<PhyloArray, PhyloRuleDynData> a = \
    [](const PhyloArray & Array, size_t i, size_t j, PhyloRuleDynData & data)
```

Definition at line 39 of file counters.hpp.

6.1.3 Function Documentation

6.1.3.1 counter_co_opt()

```
void counter_co_opt (
            PhyloCounters * counters,
            size_t nfunA,
            size_t nfunB,
            size_t duplication = Geese::etype_default )  [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 1296 of file counters.hpp.

6.1.3.2 counter_cogain()

```
void counter_cogain (
            PhyloCounters * counters,
            size_t nfunA,
            size_t nfunB,
            size_t duplication = Geese::etype_default )  [inline]
```

Co-evolution (joint gain or loss)

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

Definition at line 791 of file counters.hpp.

6.1.3.3 counter_gains()

```
void counter_gains (
            PhyloCounters * counters,
            std::vector< size_t > nfun,
            size_t duplication = Geese::etype_default )  [inline]
```

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

Definition at line 96 of file counters.hpp.

6.1.3.4 counter_gains_from_0()

```
void counter_gains_from_0 (
            PhyloCounters * counters,
            std::vector< size_t > nfun,
            size_t duplication = Geese::etype_default )  [inline]
```

Used when all the functions are in 0 (like the root node prob.)

Needs to specify function a.

Definition at line 1630 of file counters.hpp.

6.1.3.5 counter_gains_k_offspring()

```
void counter_gains_k_offspring (
            PhyloCounters * counters,
            std::vector< size_t > nfun,
            size_t k = 1u,
            size_t duplication = Geese::etype_default )  [inline]
```

k genes gain function nfun

Definition at line 156 of file counters.hpp.

6.1.3.6 counter_genes_changing()

```
void counter_genes_changing (
            PhyloCounters * counters,
            size_t duplication = Geese::etype_default )  [inline]
```

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

Definition at line 228 of file counters.hpp.

6.1.3.7 counter_k_genes_changing()

```
void counter_k_genes_changing (
            PhyloCounters * counters,
            size_t k,
            size_t duplication = Geese::etype_default )  [inline]
```

Indicator function. Equals to one if k genes changed and zero otherwise.

Definition at line 1394 of file counters.hpp.

6.1.3.8 counter_less_than_p_prop_genes_changing()

```
void counter_less_than_p_prop_genes_changing (
            PhyloCounters * counters,
            double p,
            size_t duplication = Geese::etype_default )  [inline]
```

Indicator function. Equals to one if k genes changed and zero otherwise.

$<$ How many genes diverge the parent

Definition at line 1514 of file counters.hpp.

6.1.3.9 counter_longest()

```
void counter_longest (
            PhyloCounters * counters,
            size_t duplication = Geese::etype_default )  [inline]
```

Longest branch mutates (either by gain or by loss)

Definition at line 848 of file counters.hpp.

6.1.3.10 counter_loss()

```
void counter_loss (
            PhyloCounters * counters,
            std::vector< size_t > nfun,
            size_t duplication = Geese::etype_default )  [inline]
```

Total count of losses for an specific function.

Definition at line 591 of file counters.hpp.

6.1.3.11 counter_maxfuns()

```
void counter_maxfuns (
            PhyloCounters * counters,
            size_t lb,
            size_t ub,
            size_t duplication = Geese::etype_default )  [inline]
```

Cap the number of functions per gene.

Definition at line 529 of file counters.hpp.

6.1.3.12 counter_neofun()

```
void counter_neofun (
            PhyloCounters * counters,
            size_t nfunA,
            size_t nfunB,
            size_t duplication = Geese::etype_default )  [inline]
```

Total number of neofunctionalization events.

Needs to specify pairs of function.

Definition at line 1018 of file counters.hpp.

6.1.3.13 counter_neofun_a2b()

```
void counter_neofun_a2b (
            PhyloCounters * counters,
            size_t nfunA,
            size_t nfunB,
            size_t duplication = Geese::etype_default )  [inline]
```

Total number of neofunctionalization events.

Needs to specify pairs of function.

Definition at line 1163 of file counters.hpp.

6.1.3.14 counter_overall_changes()

```
void counter_overall_changes (
            PhyloCounters * counters,
            size_t duplication = Geese::etype_default )  [inline]
```

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

Definition at line 643 of file counters.hpp.

6.1.3.15 counter_overall_gains()

```
void counter_overall_gains (
            PhyloCounters * counters,
            size_t duplication = Geese::etype_default )  [inline]
```

Overall functional gains.

Total number of gains (irrespective of the function).

Definition at line 58 of file counters.hpp.

6.1.3.16 counter_overall_gains_from_0()

```
void counter_overall_gains_from_0 (
            PhyloCounters * counters,
            size_t duplication = Geese::etype_default )  [inline]
```

Used when all the functions are in 0 (like the root node prob.)

Needs to specify function a.

Definition at line 1696 of file counters.hpp.

6.1.3.17 counter_overall_loss()

```
void counter_overall_loss (
            PhyloCounters * counters,
            size_t duplication = Geese::etype_default )  [inline]
```

Overall functional loss.

Definition at line 481 of file counters.hpp.

6.1.3.18 counter_pairwise_first_gain()

```
void counter_pairwise_first_gain (
            PhyloCounters * counters,
            size_t nfunA,
            size_t nfunB,
            size_t duplication = Geese::etype_default )  [inline]
```

Used when all the functions are in 0 (like the root node prob.)

Needs to specify function a. sum x(a)^3(1-x(b))^3 + x(b)^3(1-x(a))^3 + x(a)^3 * x(b)^3 + (1 - x(a))^3 * (1-x(b))^3

Definition at line 1948 of file counters.hpp.

6.1.3.19 counter_pairwise_neofun_singlefun()

```
void counter_pairwise_neofun_singlefun (
            PhyloCounters * counters,
            size_t nfunA,
            size_t duplication = Geese::etype_default )  [inline]
```

Total number of neofunctionalization events sum_u sum_{w < u} [x(u,a)*(1 - x(w,a)) + (1 - x(u,a)) * x(w,a)] change stat: delta{x(u,a): 0->1} = 1 - 2 * x(w,a)

Definition at line 1099 of file counters.hpp.

6.1.3.20 counter_pairwise_overall_change()

```
void counter_pairwise_overall_change (
            PhyloCounters * counters,
            size_t duplication = Geese::etype_default )  [inline]
```

Used when all the functions are in 0 (like the root node prob.)

Needs to specify function a.

Definition at line 1744 of file counters.hpp.

6.1.3.21 counter_pairwise_preserving()

```
void counter_pairwise_preserving (
            PhyloCounters * counters,
            size_t nfunA,
            size_t nfunB,
            size_t duplication = Geese::etype_default )  [inline]
```

Used when all the functions are in 0 (like the root node prob.)

Needs to specify function a. sum x(a)^3(1-x(b))^3 + x(b)^3(1-x(a))^3 + x(a)^3 * x(b)^3 + (1 - x(a))^3 * (1-x(b))^3

Definition at line 1809 of file counters.hpp.

6.1.3.22 counter_preserve_pseudogene()

```
void counter_preserve_pseudogene (
            PhyloCounters * counters,
            size_t nfunA,
            size_t nfunB,
            size_t duplication = Geese::etype_default )  [inline]
```

Keeps track of how many pairs of genes preserve pseudostate.

Definition at line 297 of file counters.hpp.

6.1.3.23 counter_prop_genes_changing()

```
void counter_prop_genes_changing (
            PhyloCounters * counters,
            size_t duplication = Geese::etype_default )  [inline]
```

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

Definition at line 379 of file counters.hpp.

6.1.3.24 counter_subfun()

```
void counter_subfun (
            PhyloCounters * counters,
            size_t nfunA,
            size_t nfunB,
            size_t duplication = Geese::etype_default )  [inline]
```

Total count of Sub-functionalization events.

It requires to specify data = {funA, funB}

Definition at line 702 of file counters.hpp.

6.1.3.25 get_last_name()

```
std::string get_last_name (
            size_t d )  [inline]
```

Definition at line 45 of file counters.hpp.

6.2 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.

6.2.1 Detailed Description

Statistical models available in `barry`.

6.3 Network counters

Counters for network models.

Functions

- template<typename Tnet = Network>
  void counter_edges (NetCounters< Tnet > *counters)

  Number of edges.
- template<typename Tnet = Network>
  void counter_isolates (NetCounters< Tnet > *counters)

  Number of isolated vertices.
- template<> void counter_isolates (NetCounters< NetworkDense > *counters)
- template<typename Tnet = Network>
  void counter_mutual (NetCounters< Tnet > *counters)

  Number of mutual ties.
- template<typename Tnet = Network>
  void counter_istar2 (NetCounters< Tnet > *counters)
- template<> void counter_istar2 (NetCounters< NetworkDense > *counters)
- template<typename Tnet = Network>
  void counter_ostar2 (NetCounters< Tnet > *counters)
- template<> void counter_ostar2 (NetCounters< NetworkDense > *counters)
- template<typename Tnet = Network>
  void counter_ttriads (NetCounters< Tnet > *counters)
- template<> void counter_ttriads (NetCounters< NetworkDense > *counters)
- template<typename Tnet = Network>
  void counter_ctriads (NetCounters< Tnet > *counters)
- template<> void counter_ctriads (NetCounters< NetworkDense > *counters)
- template<typename Tnet = Network>
  void counter_density (NetCounters< Tnet > *counters)
- template<typename Tnet = Network>
  void counter_idegree15 (NetCounters< Tnet > *counters)
- template<> void counter_idegree15 (NetCounters< NetworkDense > *counters)
- template<typename Tnet = Network>
  void counter_odegree15 (NetCounters< Tnet > *counters)
- template<> void counter_odegree15 (NetCounters< NetworkDense > *counters)
- template<typename Tnet = Network>
  void counter_absdiff (NetCounters< Tnet > *counters, size_t attr_id, double alpha=1.0)

  Sum of absolute attribute difference between ego and alter.
- template<typename Tnet = Network>
  void counter_diff (NetCounters< Tnet > *counters, size_t 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)
- template<typename Tnet = Network>
  void counter_nodeicov (NetCounters< Tnet > *counters, size_t attr_id)
- template<typename Tnet = Network>
  void counter_nodeocov (NetCounters< Tnet > *counters, size_t attr_id)
- template<typename Tnet = Network>
  void counter_nodecov (NetCounters< Tnet > *counters, size_t attr_id)
- template<typename Tnet = Network>
  void counter_nodematch (NetCounters< Tnet > *counters, size_t attr_id)
- template<typename Tnet = Network>
  void counter_idegree (NetCounters< Tnet > *counters, std::vector< size_t > d)

  Counts number of vertices with a given in-degree.
- template<> void counter_idegree (NetCounters< NetworkDense > *counters, std::vector< size_t > d)
- template<typename Tnet = Network>
  void counter_odegree (NetCounters< Tnet > *counters, std::vector< size_t > d)

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

- template<typename Tnet = Network>
  void counter_degree (NetCounters< Tnet > *counters, std::vector< size_t > d)

  Counts number of vertices with a given out-degree.
- void counter_ones (DEFMCounters *counters, int covar_index=-1, std::string vname="", const std::vector< std::string > *x_names=nullptr)

  Prevalence of ones.
- void counter_logit_intercept (DEFMCounters *counters, size_t n_y, std::vector< size_t > which={}, int covar_index=-1, std::string vname="", const std::vector< std::string > *x_names=nullptr, const std::vector< std::string > *y_names=nullptr)
- void counter_transition (DEFMCounters *counters, std::vector< size_t > coords, std::vector< bool > signs, size_t m_order, size_t n_y, int covar_index=-1, std::string vname="", const std::vector< std::string > *x_↩ names=nullptr, const std::vector< std::string > *y_names=nullptr)

  Prevalence of ones.
- void counter_transition_formula (DEFMCounters *counters, std::string formula, size_t m_order, size_t n_y, int covar_index=-1, std::string vname="", const std::vector< std::string > *x_names=nullptr, const std::vector< std::string > *y_names=nullptr)

  Prevalence of ones.
- void counter_fixed_effect (DEFMCounters *counters, int covar_index, double k, std::string vname="", const std::vector< std::string > *x_names=nullptr)

  Prevalence of ones.

Returns true if the cell is free

Parameters

rules	A pointer to a `DEFMRules` object (`Rules<DEFMArray, bool>`).

- void rules_markov_fixed (DEFMRules *rules, size_t markov_order)

  Number of edges.
- void rules_dont_become_zero (DEFMSupport *support, std::vector< size_t > ids)

  Blocks switching a one to zero.

6.3.1 Detailed Description

Counters for network models.

Parameters

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

6.3.2 Function Documentation

6.3.2.1 counter_absdiff()

```
template<typename Tnet = Network>
void counter_absdiff (
            NetCounters< Tnet > * counters,
            size_t attr_id,
            double alpha = 1.0 )  [inline]
```

Sum of absolute attribute difference between ego and alter.

Definition at line 908 of file network.hpp.

6.3.2.2 counter_ctriads() [1/2]

```
template<>
void counter_ctriads (
            NetCounters< NetworkDense > * counters )  [inline]
```

Definition at line 665 of file network.hpp.

6.3.2.3 counter_ctriads() [2/2]

```
template<typename Tnet = Network>
void counter_ctriads (
            NetCounters< Tnet > * counters )  [inline]
```

Definition at line 610 of file network.hpp.

6.3.2.4 counter_degree()

```
template<typename Tnet = Network>
void counter_degree (
            NetCounters< Tnet > * counters,
            std::vector< size_t > d )  [inline]
```

Counts number of vertices with a given out-degree.

Definition at line 1326 of file network.hpp.

6.3.2.5 counter_density()

```
template<typename Tnet = Network>
void counter_density (
            NetCounters< Tnet > * counters )  [inline]
```

Definition at line 729 of file network.hpp.

6.3.2.6 counter_diff()

```
template<typename Tnet = Network>
void counter_diff (
            NetCounters< Tnet > * counters,
            size_t 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 953 of file network.hpp.

6.3.2.7 counter_edges()

```
template<typename Tnet = Network>
void counter_edges (
            NetCounters< Tnet > * counters )  [inline]
```

Number of edges.

Definition at line 152 of file network.hpp.

6.3.2.8 counter_fixed_effect()

```
void counter_fixed_effect (
            DEFMCounters * counters,
            int covar_index,
            double k,
            std::string vname = "",
            const std::vector< std::string > * x_names = nullptr )  [inline]
```

Prevalence of ones.

Parameters

counters	Pointer ot a vector of counters
covar_index	If >= than 0, then the interaction

Definition at line 610 of file counters.hpp.

6.3.2.9 counter_idegree() [1/2]

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

Definition at line 1170 of file network.hpp.

6.3.2.10 counter_idegree() [2/2]

```
template<typename Tnet = Network>
void counter_idegree (
            NetCounters< Tnet > * counters,
            std::vector< size_t > d )  [inline]
```

Counts number of vertices with a given in-degree.

Definition at line 1123 of file network.hpp.

6.3.2.11 counter_idegree15() [1/2]

```
template<>
void counter_idegree15 (
            NetCounters< NetworkDense > * counters )  [inline]
```

Definition at line 785 of file network.hpp.

6.3.2.12 counter_idegree15() [2/2]

```
template<typename Tnet = Network>
void counter_idegree15 (
            NetCounters< Tnet > * counters )  [inline]
```

Definition at line 757 of file network.hpp.

6.3.2.13 counter_isolates() [1/2]

```
template<>
void counter_isolates (
            NetCounters< NetworkDense > * counters )  [inline]
```

Definition at line 215 of file network.hpp.

6.3.2.14 counter_isolates() [2/2]

```
template<typename Tnet = Network>
void counter_isolates (
            NetCounters< Tnet > * counters )  [inline]
```

Number of isolated vertices.

Definition at line 175 of file network.hpp.

6.3.2.15 counter_istar2() [1/2]

```
template<>
void counter_istar2 (
            NetCounters< NetworkDense > * counters )  [inline]
```

Definition at line 338 of file network.hpp.

6.3.2.16 counter_istar2() [2/2]

```
template<typename Tnet = Network>
void counter_istar2 (
            NetCounters< Tnet > * counters )  [inline]
```

Definition at line 312 of file network.hpp.

6.3.2.17 counter_logit_intercept()

```
void counter_logit_intercept (
            DEFMCounters * counters,
            size_t n_y,
            std::vector< size_t > which = {},
            int covar_index = -1,
            std::string vname = "",
            const std::vector< std::string > * x_names = nullptr,
            const std::vector< std::string > * y_names = nullptr )  [inline]
```

Definition at line 151 of file counters.hpp.

6.3.2.18 counter_mutual()

```
template<typename Tnet = Network>
void counter_mutual (
            NetCounters< Tnet > * counters )  [inline]
```

Number of mutual ties.

Definition at line 256 of file network.hpp.

6.3.2.19 counter_nodecov()

```
template<typename Tnet = Network>
void counter_nodecov (
            NetCounters< Tnet > * counters,
            size_t attr_id )  [inline]
```

Definition at line 1066 of file network.hpp.

6.3.2.20 counter_nodeicov()

```
template<typename Tnet = Network>
void counter_nodeicov (
            NetCounters< Tnet > * counters,
            size_t attr_id )  [inline]
```

Definition at line 1016 of file network.hpp.

6.3.2.21 counter_nodematch()

```
template<typename Tnet = Network>
void counter_nodematch (
            NetCounters< Tnet > * counters,
            size_t attr_id )  [inline]
```

Definition at line 1091 of file network.hpp.

6.3.2.22 counter_nodeocov()

```
template<typename Tnet = Network>
void counter_nodeocov (
            NetCounters< Tnet > * counters,
            size_t attr_id )  [inline]
```

Definition at line 1041 of file network.hpp.

6.3.2.23 counter_odegree() [1/2]

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

Definition at line 1271 of file network.hpp.

6.3.2.24 counter_odegree() [2/2]

```
template<typename Tnet = Network>
void counter_odegree (
            NetCounters< Tnet > * counters,
            std::vector< size_t > d )  [inline]
```

Counts number of vertices with a given out-degree.

Definition at line 1223 of file network.hpp.

6.3.2.25 counter_odegree15() [1/2]

```
template<>
void counter_odegree15 (
            NetCounters< NetworkDense > * counters )  [inline]
```

Definition at line 862 of file network.hpp.

6.3.2.26 counter_odegree15() [2/2]

```
template<typename Tnet = Network>
void counter_odegree15 (
            NetCounters< Tnet > * counters )  [inline]
```

Definition at line 834 of file network.hpp.

6.3.2.27 counter_ones()

```
void counter_ones (
            DEFMCounters * counters,
            int covar_index = -1,
            std::string vname = "",
            const std::vector< std::string > * x_names = nullptr )  [inline]
```

Prevalence of ones.

Parameters

counters	Pointer ot a vector of counters
covar_index	If >= than 0, then the interaction

Definition at line 81 of file counters.hpp.

6.3.2.28 counter_ostar2() [1/2]

```
template<>
void counter_ostar2 (
            NetCounters< NetworkDense > * counters )  [inline]
```

Definition at line 404 of file network.hpp.

6.3.2.29 counter_ostar2() [2/2]

```
template<typename Tnet = Network>
void counter_ostar2 (
            NetCounters< Tnet > * counters )  [inline]
```

Definition at line 376 of file network.hpp.

6.3.2.30 counter_transition()

```
void counter_transition (
            DEFMCounters * counters,
            std::vector< size_t > coords,
            std::vector< bool > signs,
            size_t m_order,
            size_t n_y,
            int covar_index = -1,
            std::string vname = "",
            const std::vector< std::string > * x_names = nullptr,
            const std::vector< std::string > * y_names = nullptr )  [inline]
```

Prevalence of ones.

Parameters

counters	Pointer ot a vector of counters
covar_index	If >= than 0, then the interaction

Definition at line 270 of file counters.hpp.

6.3.2.31 counter_transition_formula()

```
void counter_transition_formula (
            DEFMCounters * counters,
            std::string formula,
            size_t m_order,
            size_t n_y,
            int covar_index = -1,
            std::string vname = "",
            const std::vector< std::string > * x_names = nullptr,
            const std::vector< std::string > * y_names = nullptr )  [inline]
```

Prevalence of ones.

Parameters

counters	Pointer ot a vector of counters
covar_index	If >= than 0, then the interaction

Definition at line 579 of file counters.hpp.

6.3.2.32 counter_ttriads() [1/2]

```
template<>
void counter_ttriads (
            NetCounters< NetworkDense > * counters )  [inline]
```

Definition at line 531 of file network.hpp.

6.3.2.33 counter_ttriads() [2/2]

```
template<typename Tnet = Network>
void counter_ttriads (
            NetCounters< Tnet > * counters )  [inline]
```

Definition at line 441 of file network.hpp.

6.3.2.34 NETWORK_COUNTER()

```
NETWORK_COUNTER (
            init_single_attr  )
```

Definition at line 997 of file network.hpp.

6.3.2.35 rules_dont_become_zero()

```
void rules_dont_become_zero (
            DEFMSupport * support,
            std::vector< size_t > ids )  [inline]
```

Blocks switching a one to zero.

Parameters

rules	
ids	Ids of the variables that will follow this rule.

Definition at line 678 of file counters.hpp.

6.3.2.36 rules_markov_fixed()

```
void rules_markov_fixed (
            DEFMRules * rules,
            size_t markov_order )  [inline]
```

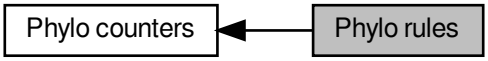
Number of edges.

Definition at line 653 of file counters.hpp.

6.4 Phylo rules

Rules for phylogenetic modeling.

Collaboration diagram for Phylo rules:



Classes

- class DEFMRuleDynData

Functions

- DEFMData::DEFMData ()

  Vector indicating which covariates are included in the model.
- DEFMData::DEFMData (DEFMArray *array_, const double *covariates_, size_t obs_start_, size_t X_ncol_, size_t X_nrow_)

  Constructor.
- double DEFMData::operator() (size_t i, size_t j) const

  Access to the row (i) colum (j) data.
- double DEFMData::at (size_t i, size_t j) const
- size_t DEFMData::ncol () const
- size_t DEFMData::nrow () const
- void DEFMData::print () const
- DEFMData::~DEFMData ()
- DEFMCounterData::DEFMCounterData ()
- DEFMCounterData::DEFMCounterData (const std::vector< size_t > indices_, const std::vector< double > numbers_, const std::vector< bool > logical_, bool is_motif_=true)
- size_t DEFMCounterData::idx (size_t i) const
- double DEFMCounterData::num (size_t i) const
- bool DEFMCounterData::is_true (size_t i) const
- DEFMCounterData::~DEFMCounterData ()
- double DEFMRuleData::num (size_t i) const
- size_t DEFMRuleData::idx (size_t i) const
- bool DEFMRuleData::is_true (size_t i) const
- DEFMRuleData::DEFMRuleData ()
- DEFMRuleData::DEFMRuleData (std::vector< double > numbers_, std::vector< size_t > indices_, std::vector< bool > logical_)
- DEFMRuleData::DEFMRuleData (std::vector< double > numbers_, std::vector< size_t > indices_)
- DEFMRuleDynData::DEFMRuleDynData (const std::vector< double > *counts_, std::vector< double > numbers_={}, std::vector< size_t > indices_={}, std::vector< bool > logical_={})
- DEFMRuleDynData::~DEFMRuleDynData ()
- void rule_dyn_limit_changes (PhyloSupport *support, size_t pos, size_t lb, size_t ub, size_t duplication=Geese::etype_default)

  Overall functional gains.

Variables

- DEFMArray * DEFMData::array
- const double * DEFMData::covariates

  Vector of covariates (complete vector)
- size_t DEFMData::obs_start

  Index of the observation in the data.
- size_t DEFMData::X_ncol

  Number of columns in the array of covariates.
- size_t DEFMData::X_nrow

  Number of rows in the array of covariates.
- std::vector< size_t > DEFMData::covar_sort
- std::vector< size_t > DEFMData::covar_used

  Value where the sorting of the covariates is stored.
- std::vector< size_t > DEFMCounterData::indices
- std::vector< double > DEFMCounterData::numbers
- std::vector< bool > DEFMCounterData::logical
- bool DEFMCounterData::is_motif

If false, then is a logit intercept.

- std::vector< double > DEFMRuleData::numbers
- std::vector< size_t > DEFMRuleData::indices
- std::vector< bool > DEFMRuleData::logical
- bool DEFMRuleData::init = false
- const std::vector< double > * DEFMRuleDynData::counts

Convenient typedefs for network objects.

- typedef barry::Counter< DEFMArray, DEFMCounterData > DEFMCounter
- typedef barry::Counters< DEFMArray, DEFMCounterData > DEFMCounters
- typedef barry::Support< DEFMArray, DEFMCounterData, DEFMRuleData, DEFMRuleDynData > DEFMSupport
- typedef barry::StatsCounter< DEFMArray, DEFMCounterData > DEFMStatsCounter
- typedef barry::Model< DEFMArray, DEFMCounterData, DEFMRuleData, DEFMRuleDynData > DEFMModel
- typedef barry::Rule< DEFMArray, DEFMRuleData > DEFMRule
- typedef barry::Rules< DEFMArray, DEFMRuleData > DEFMRules
- typedef barry::Rule< DEFMArray, DEFMRuleDynData > DEFMRuleDyn
- typedef barry::Rules< DEFMArray, DEFMRuleDynData > DEFMRulesDyn

6.4.1 Detailed Description

Rules for phylogenetic modeling.

Parameters

rules	A pointer to a `PhyloRules` object (`Rules<PhyloArray, PhyloRuleData>`).

6.4.2 Typedef Documentation

6.4.2.1 DEFMCounter

```
typedef barry::Counter<DEFMArray, DEFMCounterData > DEFMCounter
```

Definition at line 171 of file defm-types.hpp.

6.4.2.2 DEFMCounters

```
typedef barry::Counters<DEFMArray, DEFMCounterData> DEFMCounters
```

Definition at line 172 of file defm-types.hpp.

6.4.2.3 DEFMModel

```
typedef barry::Model<DEFMArray, DEFMCounterData,DEFMRuleData,DEFMRuleDynData> DEFMModel
```

Definition at line 175 of file defm-types.hpp.

6.4.2.4 DEFMRule

```
typedef barry::Rule<DEFMArray, DEFMRuleData> DEFMRule
```

Definition at line 178 of file defm-types.hpp.

6.4.2.5 DEFMRuleDyn

```
typedef barry::Rule<DEFMArray, DEFMRuleDynData> DEFMRuleDyn
```

Definition at line 180 of file defm-types.hpp.

6.4.2.6 DEFMRules

```
typedef barry::Rules<DEFMArray, DEFMRuleData> DEFMRules
```

Definition at line 179 of file defm-types.hpp.

6.4.2.7 DEFMRulesDyn

```
typedef barry::Rules<DEFMArray, DEFMRuleDynData> DEFMRulesDyn
```

Definition at line 181 of file defm-types.hpp.

6.4.2.8 DEFMStatsCounter

```
typedef barry::StatsCounter<DEFMArray, DEFMCounterData> DEFMStatsCounter
```

Definition at line 174 of file defm-types.hpp.

6.4.2.9 DEFMSupport

```
typedef barry::Support<DEFMArray, DEFMCounterData, DEFMRuleData,DEFMRuleDynData> DEFMSupport
```

Definition at line 173 of file defm-types.hpp.

6.4.3 Function Documentation

6.4.3.1 at()

```
double DEFMData::at (
            size_t i,
            size_t j ) const
```

6.4.3.2 DEFMCounterData() [1/2]

```
DEFMCounterData::DEFMCounterData ( )  [inline]
```

Definition at line 72 of file defm-types.hpp.

6.4.3.3 DEFMCounterData() [2/2]

```
DEFMCounterData::DEFMCounterData (
            const std::vector< size_t > indices_,
            const std::vector< double > numbers_,
            const std::vector< bool > logical_,
            bool is_motif_ = true )  [inline]
```

Definition at line 73 of file defm-types.hpp.

6.4.3.4 DEFMData() [1/2]

```
DEFMData::DEFMData ( )  [inline]
```

Vector indicating which covariates are included in the model.

Definition at line 27 of file defm-types.hpp.

6.4.3.5 DEFMData() [2/2]

```
DEFMData::DEFMData (
            DEFMArray * array_,
            const double * covariates_,
            size_t obs_start_,
            size_t X_ncol_,
            size_t X_nrow_ )  [inline]
```

Constructor.

Parameters

*covariates_*	Pointer to the attribute data.
*obs_start_*	Location of the current observation in the covariates vector
*X_ncol_*	Number of columns (covariates.)

Definition at line 36 of file defm-types.hpp.

6.4.3.6 DEFMRuleData() [1/3]

```
DEFMRuleData::DEFMRuleData ( )  [inline]
```

Definition at line 102 of file defm-types.hpp.

6.4.3.7 DEFMRuleData() [2/3]

```
DEFMRuleData::DEFMRuleData (
            std::vector< double > numbers_,
            std::vector< size_t > indices_ )  [inline]
```

Definition at line 110 of file defm-types.hpp.

6.4.3.8 DEFMRuleData() [3/3]

```
DEFMRuleData::DEFMRuleData (
            std::vector< double > numbers_,
            std::vector< size_t > indices_,
            std::vector< bool > logical_ )  [inline]
```

Definition at line 104 of file defm-types.hpp.

6.4.3.9 DEFMRuleDynData()

```
DEFMRuleDynData::DEFMRuleDynData (
            const std::vector< double > * counts_,
            std::vector< double > numbers_ = {},
            std::vector< size_t > indices_ = {},
            std::vector< bool > logical_ = {} )  [inline]
```

Definition at line 156 of file defm-types.hpp.

6.4.3.10 idx() [1/2]

```
size_t DEFMCounterData::idx (
            size_t i ) const  [inline]
```

Definition at line 81 of file defm-types.hpp.

6.4.3.11 idx() [2/2]

```
size_t DEFMRuleData::idx (
            size_t i ) const  [inline]
```

Definition at line 99 of file defm-types.hpp.

6.4.3.12 is_true() [1/2]

```
bool DEFMCounterData::is_true (
            size_t i ) const  [inline]
```

Definition at line 83 of file defm-types.hpp.

6.4.3.13 is_true() [2/2]

```
bool DEFMRuleData::is_true (
            size_t i ) const  [inline]
```

Definition at line 100 of file defm-types.hpp.

6.4.3.14 ncol()

```
size_t DEFMData::ncol ( ) const  [inline]
```

Definition at line 123 of file defm-types.hpp.

6.4.3.15 nrow()

```
size_t DEFMData::nrow ( ) const  [inline]
```

Definition at line 127 of file defm-types.hpp.

6.4.3.16 num() [1/2]

```
double DEFMCounterData::num (
            size_t i ) const  [inline]
```

Definition at line 82 of file defm-types.hpp.

6.4.3.17 num() [2/2]

```
double DEFMRuleData::num (
            size_t i ) const  [inline]
```

Definition at line 98 of file defm-types.hpp.

6.4.3.18 operator()()

```
double DEFMData::operator() (
            size_t i,
            size_t j ) const  [inline]
```

Access to the row (i) colum (j) data.

Parameters

i	
j	

Returns

double

Definition at line 118 of file defm-types.hpp.

6.4.3.19 print()

```
void DEFMData::print ( ) const  [inline]
```

Definition at line 131 of file defm-types.hpp.

6.4.3.20 rule_dyn_limit_changes()

```
void rule_dyn_limit_changes (
            PhyloSupport * support,
            size_t pos,
            size_t lb,
            size_t ub,
            size_t duplication = Geese::etype_default )  [inline]
```

Overall functional gains.

Parameters

support	Support of a model.
pos	Position of the focal statistic.
lb	Lower bound
ub	Upper bound

Returns

(void) adds a rule limiting the support of the model.

Definition at line 2058 of file counters.hpp.

6.4.3.21 ~DEFMCounterData()

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

Definition at line 85 of file defm-types.hpp.

6.4.3.22 ~DEFMData()

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

Definition at line 58 of file defm-types.hpp.

6.4.3.23 ~DEFMRuleDynData()

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

Definition at line 163 of file defm-types.hpp.

6.4.4 Variable Documentation

6.4.4.1 array

```
DEFMArray* DEFMData::array
```

Definition at line 19 of file defm-types.hpp.

6.4.4.2 counts

```
const std::vector< double >* DEFMRuleDynData::counts
```

Definition at line 154 of file defm-types.hpp.

6.4.4.3 covar_sort

```
std::vector< size_t > DEFMData::covar_sort
```

Definition at line 24 of file defm-types.hpp.

6.4.4.4 covar_used

```
std::vector< size_t > DEFMData::covar_used
```

Value where the sorting of the covariates is stored.

Definition at line 25 of file defm-types.hpp.

6.4.4.5 covariates

```
const double* DEFMData::covariates
```

Vector of covariates (complete vector)

Definition at line 20 of file defm-types.hpp.

6.4.4.6 indices [1/2]

```
std::vector< size_t > DEFMCounterData::indices
```

Definition at line 67 of file defm-types.hpp.

6.4.4.7 indices [2/2]

```
std::vector< size_t > DEFMRuleData::indices
```

Definition at line 93 of file defm-types.hpp.

6.4.4.8 init

```
bool DEFMRuleData::init = false
```

Definition at line 96 of file defm-types.hpp.

6.4.4.9 is_motif

```
bool DEFMCounterData::is_motif
```

If false, then is a logit intercept.

Definition at line 70 of file defm-types.hpp.

6.4.4.10 logical [1/2]

```
std::vector< bool > DEFMCounterData::logical
```

Definition at line 69 of file defm-types.hpp.

6.4.4.11 logical [2/2]

```
std::vector< bool > DEFMRuleData::logical
```

Definition at line 94 of file defm-types.hpp.

6.4.4.12 numbers [1/2]

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

Definition at line 68 of file defm-types.hpp.

6.4.4.13 numbers [2/2]

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

Definition at line 92 of file defm-types.hpp.

6.4.4.14 obs_start

```
size_t DEFMData::obs_start
```

Index of the observation in the data.

Definition at line 21 of file defm-types.hpp.

6.4.4.15 X_ncol

```
size_t DEFMData::X_ncol
```

Number of columns in the array of covariates.

Definition at line 22 of file defm-types.hpp.

6.4.4.16 X_nrow

```
size_t DEFMData::X_nrow
```

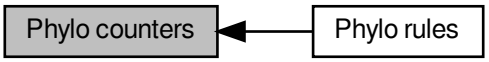
Number of rows in the array of covariates.

Definition at line 23 of file defm-types.hpp.

6.5 Phylo counters

Counters for phylogenetic modeling.

Collaboration diagram for Phylo counters:



Modules

- Phylo rules

  Rules for phylogenetic modeling.

- void counter_overall_gains (PhyloCounters *counters, size_t duplication=Geese::etype_default)

  Overall functional gains.
- void counter_gains (PhyloCounters *counters, std::vector< size_t > nfun, size_t duplication=Geese::etype_default)

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

  k genes gain function nfun
- void counter_genes_changing (PhyloCounters *counters, size_t duplication=Geese::etype_default)

  Keeps track of how many genes are changing (either 0, 1, or 2 if dealing with regular trees.)
- void counter_preserve_pseudogene (PhyloCounters *counters, size_t nfunA, size_t nfunB, size_t duplication=Geese::etype_default)

  Keeps track of how many pairs of genes preserve pseudostate.
- void counter_prop_genes_changing (PhyloCounters *counters, size_t duplication=Geese::etype_default)

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

  Overall functional loss.
- void counter_maxfuns (PhyloCounters *counters, size_t lb, size_t ub, size_t duplication=Geese::etype_default)

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

  Total count of losses for an specific function.
- void counter_overall_changes (PhyloCounters *counters, size_t duplication=Geese::etype_default)

  Total number of changes. Use this statistic to account for "preservation".
- void counter_subfun (PhyloCounters *counters, size_t nfunA, size_t nfunB, size_t duplication=Geese::etype_default)

  Total count of Sub-functionalization events.
- void counter_cogain (PhyloCounters *counters, size_t nfunA, size_t nfunB, size_t duplication=Geese::etype_default)

  Co-evolution (joint gain or loss)
- void counter_longest (PhyloCounters *counters, size_t duplication=Geese::etype_default)

  Longest branch mutates (either by gain or by loss)
- void counter_neofun (PhyloCounters *counters, size_t nfunA, size_t nfunB, size_t duplication=Geese::etype_default)

  Total number of neofunctionalization events.

- void counter_pairwise_neofun_singlefun (PhyloCounters *counters, size_t nfunA, size_t duplication=Geese::etype_default)

  Total number of neofunctionalization events sum_u sum_{w < u} [x(u,a)(1 - x(w,a)) + (1 - x(u,a)) * x(w,a)] change stat: delta{x(u,a): 0->1} = 1 - 2 * x(w,a)*
- void counter_neofun_a2b (PhyloCounters *counters, size_t nfunA, size_t nfunB, size_t duplication=Geese::etype_default)

  Total number of neofunctionalization events.
- void counter_co_opt (PhyloCounters *counters, size_t nfunA, size_t nfunB, size_t duplication=Geese::etype_default)

  Function co-opting.
- void counter_k_genes_changing (PhyloCounters *counters, size_t k, size_t duplication=Geese::etype_default)

  Indicator function. Equals to one if k genes changed and zero otherwise.
- void counter_less_than_p_prop_genes_changing (PhyloCounters *counters, double p, size_t duplication=Geese::etype_default

  Indicator function. Equals to one if k genes changed and zero otherwise.
- void counter_gains_from_0 (PhyloCounters *counters, std::vector< size_t > nfun, size_t duplication=Geese::etype_default)

  Used when all the functions are in 0 (like the root node prob.)
- void counter_overall_gains_from_0 (PhyloCounters *counters, size_t duplication=Geese::etype_default)

  Used when all the functions are in 0 (like the root node prob.)
- void counter_pairwise_overall_change (PhyloCounters *counters, size_t duplication=Geese::etype_default)

  Used when all the functions are in 0 (like the root node prob.)
- void counter_pairwise_preserving (PhyloCounters *counters, size_t nfunA, size_t nfunB, size_t duplication=Geese::etype_default)

  Used when all the functions are in 0 (like the root node prob.)
- void counter_pairwise_first_gain (PhyloCounters *counters, size_t nfunA, size_t nfunB, size_t duplication=Geese::etype_default)

  Used when all the functions are in 0 (like the root node prob.)

6.5.1 Detailed Description

Counters for phylogenetic modeling.

Parameters

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

6.5.2 Function Documentation

6.5.2.1 counter_co_opt()

```
void counter_co_opt (
            PhyloCounters * counters,
            size_t nfunA,
            size_t nfunB,
            size_t duplication = Geese::etype_default )  [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 1296 of file counters.hpp.

6.5.2.2 counter_cogain()

```
void counter_cogain (
            PhyloCounters * counters,
            size_t nfunA,
            size_t nfunB,
            size_t duplication = Geese::etype_default )  [inline]
```

Co-evolution (joint gain or loss)

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

Definition at line 791 of file counters.hpp.

6.5.2.3 counter_gains()

```
void counter_gains (
            PhyloCounters * counters,
            std::vector< size_t > nfun,
            size_t duplication = Geese::etype_default )  [inline]
```

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

Definition at line 96 of file counters.hpp.

6.5.2.4 counter_gains_from_0()

```
void counter_gains_from_0 (
            PhyloCounters * counters,
            std::vector< size_t > nfun,
            size_t duplication = Geese::etype_default )  [inline]
```

Used when all the functions are in 0 (like the root node prob.)

Needs to specify function a.

Definition at line 1630 of file counters.hpp.

6.5.2.5 counter_gains_k_offspring()

```
void counter_gains_k_offspring (
            PhyloCounters * counters,
            std::vector< size_t > nfun,
            size_t k = 1u,
            size_t duplication = Geese::etype_default )  [inline]
```

k genes gain function nfun

Definition at line 156 of file counters.hpp.

6.5.2.6 counter_genes_changing()

```
void counter_genes_changing (
            PhyloCounters * counters,
            size_t duplication = Geese::etype_default )  [inline]
```

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

Definition at line 228 of file counters.hpp.

6.5.2.7 counter_k_genes_changing()

```
void counter_k_genes_changing (
            PhyloCounters * counters,
            size_t k,
            size_t duplication = Geese::etype_default )  [inline]
```

Indicator function. Equals to one if k genes changed and zero otherwise.

Definition at line 1394 of file counters.hpp.

6.5.2.8 counter_less_than_p_prop_genes_changing()

```
void counter_less_than_p_prop_genes_changing (
            PhyloCounters * counters,
            double p,
            size_t duplication = Geese::etype_default )  [inline]
```

Indicator function. Equals to one if k genes changed and zero otherwise.

$<$ How many genes diverge the parent

Definition at line 1514 of file counters.hpp.

6.5.2.9 counter_longest()

```
void counter_longest (
            PhyloCounters * counters,
            size_t duplication = Geese::etype_default )  [inline]
```

Longest branch mutates (either by gain or by loss)

Definition at line 848 of file counters.hpp.

6.5.2.10 counter_loss()

```
void counter_loss (
            PhyloCounters * counters,
            std::vector< size_t > nfun,
            size_t duplication = Geese::etype_default )  [inline]
```

Total count of losses for an specific function.

Definition at line 591 of file counters.hpp.

6.5.2.11 counter_maxfuns()

```
void counter_maxfuns (
            PhyloCounters * counters,
            size_t lb,
            size_t ub,
            size_t duplication = Geese::etype_default )  [inline]
```

Cap the number of functions per gene.

Definition at line 529 of file counters.hpp.

6.5.2.12 counter_neofun()

```
void counter_neofun (
            PhyloCounters * counters,
            size_t nfunA,
            size_t nfunB,
            size_t duplication = Geese::etype_default )  [inline]
```

Total number of neofunctionalization events.

Needs to specify pairs of function.

Definition at line 1018 of file counters.hpp.

6.5.2.13 counter_neofun_a2b()

```
void counter_neofun_a2b (
            PhyloCounters * counters,
            size_t nfunA,
            size_t nfunB,
            size_t duplication = Geese::etype_default )  [inline]
```

Total number of neofunctionalization events.

Needs to specify pairs of function.

Definition at line 1163 of file counters.hpp.

6.5.2.14 counter_overall_changes()

```
void counter_overall_changes (
            PhyloCounters * counters,
            size_t duplication = Geese::etype_default )  [inline]
```

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

Definition at line 643 of file counters.hpp.

6.5.2.15 counter_overall_gains()

```
void counter_overall_gains (
            PhyloCounters * counters,
            size_t duplication = Geese::etype_default )  [inline]
```

Overall functional gains.

Total number of gains (irrespective of the function).

Definition at line 58 of file counters.hpp.

6.5.2.16 counter_overall_gains_from_0()

```
void counter_overall_gains_from_0 (
            PhyloCounters * counters,
            size_t duplication = Geese::etype_default )  [inline]
```

Used when all the functions are in 0 (like the root node prob.)

Needs to specify function a.

Definition at line 1696 of file counters.hpp.

6.5.2.17 counter_overall_loss()

```
void counter_overall_loss (
            PhyloCounters * counters,
            size_t duplication = Geese::etype_default )  [inline]
```

Overall functional loss.

Definition at line 481 of file counters.hpp.

6.5.2.18 counter_pairwise_first_gain()

```
void counter_pairwise_first_gain (
            PhyloCounters * counters,
            size_t nfunA,
            size_t nfunB,
            size_t duplication = Geese::etype_default )  [inline]
```

Used when all the functions are in 0 (like the root node prob.)

Needs to specify function a. sum x(a)^3(1-x(b))^3 + x(b)^3(1-x(a))^3 + x(a)^3 * x(b)^3 + (1 - x(a))^3 * (1-x(b))^3

Definition at line 1948 of file counters.hpp.

6.5.2.19 counter_pairwise_neofun_singlefun()

```
void counter_pairwise_neofun_singlefun (
            PhyloCounters * counters,
            size_t nfunA,
            size_t duplication = Geese::etype_default )  [inline]
```

Total number of neofunctionalization events sum_u sum_{w < u} [x(u,a)*(1 - x(w,a)) + (1 - x(u,a)) * x(w,a)] change stat: delta{x(u,a): 0->1} = 1 - 2 * x(w,a)

Definition at line 1099 of file counters.hpp.

6.5.2.20 counter_pairwise_overall_change()

```
void counter_pairwise_overall_change (
            PhyloCounters * counters,
            size_t duplication = Geese::etype_default )  [inline]
```

Used when all the functions are in 0 (like the root node prob.)

Needs to specify function a.

Definition at line 1744 of file counters.hpp.

6.5.2.21 counter_pairwise_preserving()

```
void counter_pairwise_preserving (
            PhyloCounters * counters,
            size_t nfunA,
            size_t nfunB,
            size_t duplication = Geese::etype_default )  [inline]
```

Used when all the functions are in 0 (like the root node prob.)

Needs to specify function a. sum x(a)^3(1-x(b))^3 + x(b)^3(1-x(a))^3 + x(a)^3 * x(b)^3 + (1 - x(a))^3 * (1-x(b))^3

Definition at line 1809 of file counters.hpp.

6.5.2.22 counter_preserve_pseudogene()

```
void counter_preserve_pseudogene (
            PhyloCounters * counters,
            size_t nfunA,
            size_t nfunB,
            size_t duplication = Geese::etype_default )  [inline]
```

Keeps track of how many pairs of genes preserve pseudostate.

Definition at line 297 of file counters.hpp.

6.5.2.23 counter_prop_genes_changing()

```
void counter_prop_genes_changing (
            PhyloCounters * counters,
            size_t duplication = Geese::etype_default )  [inline]
```

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

Definition at line 379 of file counters.hpp.

6.5.2.24 counter_subfun()

```
void counter_subfun (
            PhyloCounters * counters,
            size_t nfunA,
            size_t nfunB,
            size_t duplication = Geese::etype_default )  [inline]
```

Total count of Sub-functionalization events.

It requires to specify data = {funA, funB}

Definition at line 702 of file counters.hpp.

Chapter 7

Namespace Documentation

7.1 barry Namespace Reference

barry: Your go-to motif accountant

Namespaces

- counters

  Tree class and TreeIterator class.

7.1.1 Detailed Description

barry: Your go-to motif accountant

7.2 barry::counters Namespace Reference

Tree class and TreeIterator class.

Namespaces

- network

7.2.1 Detailed Description

Tree class and TreeIterator class.

7.3 barry::counters::network Namespace Reference

7.4 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

7.4.1 Detailed Description

Integer constants used to specify which cell should be check.

7.4.2 Variable Documentation

7.4.2.1 BOTH

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

Definition at line 27 of file typedefs.hpp.

7.4.2.2 NONE

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

Definition at line 28 of file typedefs.hpp.

7.4.2.3 ONE

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

Definition at line 29 of file typedefs.hpp.

7.4.2.4 TWO

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

Definition at line 30 of file typedefs.hpp.

7.5 defm Namespace Reference

7.6 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

7.6.1 Detailed Description

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

7.6.2 Variable Documentation

7.6.2.1 AS_ONE

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

Definition at line 45 of file typedefs.hpp.

7.6.2.2 AS_ZERO

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

Definition at line 44 of file typedefs.hpp.

7.6.2.3 BOTH

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

Definition at line 38 of file typedefs.hpp.

7.6.2.4 NONE

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

Definition at line 39 of file typedefs.hpp.

7.6.2.5 ONE

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

Definition at line 40 of file typedefs.hpp.

7.6.2.6 TWO

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

Definition at line 41 of file typedefs.hpp.

7.6.2.7 UKNOWN

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

Definition at line 43 of file typedefs.hpp.

7.7 geese Namespace Reference

Chapter 8

Class Documentation

8.1 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 (size_t i, size_t j) const
- Cell_Type get_cell (size_t i, size_t j, bool check_bounds=true) const
- std::vector< Cell_Type > get_col_vec (size_t i, bool check_bounds=true) const
- std::vector< Cell_Type > get_row_vec (size_t i, bool check_bounds=true) const
- void get_col_vec (std::vector< Cell_Type > *x, size_t i, bool check_bounds=true) const
- void get_row_vec (std::vector< Cell_Type > *x, size_t i, bool check_bounds=true) const
- const Row_type< Cell_Type > & row (size_t i, bool check_bounds=true) const
- const Col_type< Cell_Type > & col (size_t i, bool check_bounds=true) const
- Entries< Cell_Type > get_entries () const

  Get the edgelist.
- void transpose ()
- void clear (bool hard=true)
- void resize (size_t N_, size_t M_)
- void reserve ()
- void print (const char *fmt=nullptr,...) const
- void print_n (size_t nrow, size_t ncol, const char *fmt=nullptr,...) const
- bool is_dense () const noexcept

Constructors

Parameters

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

- BArray ()
  Zero-size array.
- BArray (size_t N_, size_t M_)
  Empty array.
- BArray (size_t N_, size_t M_, const std::vector< size_t > &source, const std::vector< size_t > &target, const std::vector< Cell_Type > &value, bool add=true)
  Edgelist with data.
- BArray (size_t N_, size_t M_, const std::vector< size_t > &source, const std::vector< size_t > &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_ptr ()
- const Data_Type * D_ptr () const
- Data_Type & D ()
- const Data_Type & D () const
- void flush_data ()

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

Column/row wise interchange

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

8.1.1 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<size_t,Cell_Type> >`.

Template Parameters

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

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

8.1.2 Constructor & Destructor Documentation

8.1.2.1 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 69 of file barray-bones.hpp.

8.1.2.2 BArray() [2/6]

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

Empty array.

Definition at line 72 of file barray-bones.hpp.

8.1.2.3 BArray() [3/6]

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

Edgelist with data.

8.1.2.4 BArray() [4/6]

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

Edgelist with no data (simpler)

8.1.2.5 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.

8.1.2.6 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.

8.1.2.7 ~BArray()

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

8.1.3 Member Function Documentation

8.1.3.1 clear()

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

8.1.3.2 col()

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

8.1.3.3 D() [1/2]

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

8.1.3.4 D() [2/2]

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

8.1.3.5 D_ptr() [1/2]

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

8.1.3.6 D_ptr() [2/2]

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

8.1.3.7 default_val()

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

8.1.3.8 flush_data()

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

8.1.3.9 get_cell()

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

8.1.3.10 get_col_vec() [1/2]

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

8.1.3.11 get_col_vec() [2/2]

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

8.1.3.12 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>

8.1.3.13 get_row_vec() [1/2]

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

8.1.3.14 get_row_vec() [2/2]

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

8.1.3.15 insert_cell() [1/3]

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

8.1.3.16 insert_cell() [2/3]

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

8.1.3.17 insert_cell() [3/3]

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

8.1.3.18 is_dense()

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

Definition at line 241 of file barray-bones.hpp.

8.1.3.19 is_empty()

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

8.1.3.20 ncol()

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

8.1.3.21 nnozero()

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

8.1.3.22 nrow()

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

8.1.3.23 operator()() [1/2]

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

8.1.3.24 operator()() [2/2]

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

8.1.3.25 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 )
```

8.1.3.26 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 )
```

8.1.3.27 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 )
```

8.1.3.28 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< size_t, size_t > & coords )
```

8.1.3.29 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 )
```

8.1.3.30 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 )
```

8.1.3.31 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< size_t, size_t > & coords )
```

8.1.3.32 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 )
```

8.1.3.33 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.

8.1.3.34 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.

8.1.3.35 operator==()

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

8.1.3.36 out_of_range()

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

8.1.3.37 print()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
void BArray< Cell_Type, Data_Type >::print (
            const char * fmt = nullptr,
             ... ) const
```

8.1.3.38 print_n()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
void BArray< Cell_Type, Data_Type >::print_n (
            size_t nrow,
            size_t ncol,
            const char * fmt = nullptr,
             ... ) const
```

8.1.3.39 reserve()

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

8.1.3.40 resize()

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

8.1.3.41 rm_cell()

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

8.1.3.42 row()

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

8.1.3.43 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_*	

8.1.3.44 swap_cells()

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

8.1.3.45 swap_cols()

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

8.1.3.46 swap_rows()

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

8.1.3.47 toggle_cell()

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

8.1.3.48 toggle_lock()

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

8.1.3.49 transpose()

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

8.1.3.50 zero_col()

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

8.1.3.51 zero_row()

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

8.1.4 Friends And Related Function Documentation

8.1.4.1 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.

8.1.4.2 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.

8.1.5 Member Data Documentation

8.1.5.1 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 54 of file barray-bones.hpp.

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

- include/barry/barray-bones.hpp

8.2 BArrayCell< Cell_Type, Data_Type > Class Template Reference

`#include <barraycell-bones.hpp>`

Public Member Functions

- BArrayCell (BArray< Cell_Type, Data_Type > *Array_, size_t i_, size_t 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

8.2.1 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.

8.2.2 Constructor & Destructor Documentation

8.2.2.1 BArrayCell()

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

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

8.2.2.2 ~BArrayCell()

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

Definition at line 31 of file barraycell-bones.hpp.

8.2.3 Member Function Documentation

8.2.3.1 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.

8.2.3.2 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.

8.2.3.3 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.

8.2.3.4 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.

8.2.3.5 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.

8.2.3.6 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.

8.2.3.7 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/barray-bones.hpp
- include/barry/barraycell-bones.hpp
- include/barry/barraycell-meat.hpp
- include/barry/barrayrow-meat.hpp

8.3 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_, size_t i_, size_t 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

8.3.1 Detailed Description

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

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

8.3.2 Constructor & Destructor Documentation

8.3.2.1 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_,
            size_t i_,
            size_t j_,
            bool check_bounds = true )  [inline]
```

Definition at line 55 of file barraycell-bones.hpp.

8.3.2.2 ~BArrayCell_const()

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

Definition at line 67 of file barraycell-bones.hpp.

8.3.3 Member Function Documentation

8.3.3.1 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.

8.3.3.2 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.

8.3.3.3 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.

8.3.3.4 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.

8.3.3.5 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.

8.3.3.6 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.

8.3.3.7 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/barray-bones.hpp
- include/barry/barraycell-bones.hpp
- include/barry/barraycell-meat.hpp
- include/barry/barrayrow-meat.hpp

8.4 BArrayDense< Cell_Type, Data_Type > Class Template Reference

Baseline class for binary arrays.

`#include <barraydense-bones.hpp>`

Public Member Functions

- bool operator== (const BArrayDense< Cell_Type, Data_Type > &Array_)
- ~BArrayDense ()
- void out_of_range (size_t i, size_t j) const
- Cell_Type get_cell (size_t i, size_t j, bool check_bounds=true) const
- std::vector< Cell_Type > get_col_vec (size_t i, bool check_bounds=true) const
- std::vector< Cell_Type > get_row_vec (size_t i, bool check_bounds=true) const
- void get_col_vec (std::vector< Cell_Type > *x, size_t i, bool check_bounds=true) const
- void get_row_vec (std::vector< Cell_Type > *x, size_t i, bool check_bounds=true) const
- BArrayDenseRow< Cell_Type, Data_Type > & row (size_t i, bool check_bounds=true)
- const BArrayDenseRow_const< Cell_Type, Data_Type > row (size_t i, bool check_bounds=true) const
- BArrayDenseCol< Cell_Type, Data_Type > & col (size_t j, bool check_bounds=true)
- const BArrayDenseCol_const< Cell_Type, Data_Type > col (size_t j, bool check_bounds=true) const
- Entries< Cell_Type > get_entries () const

  Get the edgelist.
- void transpose ()
- void clear (bool hard=true)
- void resize (size_t N_, size_t M_)
- void reserve ()
- void print (const char *fmt=nullptr,...) const
- bool is_dense () const noexcept
- const std::vector< Cell_Type > & get_data () const
- const Cell_Type rowsum (size_t i) const
- const Cell_Type colsum (size_t i) const

Constructors

Parameters

N_	*Number of rows*
M_	*Number of columns*
source	*An unsigned vector ranging from 0 to N_*
target	*An size_t vector ranging from 0 to M_*
target	*When `true` tries to add repeated observations.*
value	*Cell_Type defaul fill-in value (zero, by default.)*

- BArrayDense ()

  Zero-size array.
- BArrayDense (size_t N_, size_t M_, Cell_Type value=static_cast< Cell_Type >(0))

  Empty array.
- BArrayDense (size_t N_, size_t M_, const std::vector< size_t > &source, const std::vector< size_t > &target, const std::vector< Cell_Type > &value, bool add=true)

  Edgelist with data.
- BArrayDense (size_t N_, size_t M_, const std::vector< size_t > &source, const std::vector< size_t > &target, bool add=true)

Edgelist with no data (simpler)

- BArrayDense (const BArrayDense< Cell_Type, Data_Type > &Array_, bool copy_data=false)

  Copy constructor.
- BArrayDense< Cell_Type, Data_Type > & operator= (const BArrayDense< Cell_Type, Data_Type > &Array_)

  Assignment constructor.
- BArrayDense (BArrayDense< Cell_Type, Data_Type > &&x) noexcept

  Move operator.
- BArrayDense< Cell_Type, Data_Type > & operator= (BArrayDense< 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_ptr ()
- const Data_Type * D_ptr () const
- 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 (size_t i, size_t j, bool check_bounds=true) const
- size_t nrow () const noexcept
- size_t ncol () const noexcept
- size_t 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.*

- BArrayDense< Cell_Type, Data_Type > & operator+= (const std::pair< size_t, size_t > &coords)
- BArrayDense< Cell_Type, Data_Type > & operator-= (const std::pair< size_t, size_t > &coords)
- BArrayDenseCell< Cell_Type, Data_Type > operator() (size_t i, size_t j, bool check_bounds=true)
- const Cell_Type operator() (size_t i, size_t j, bool check_bounds=true) const
- void rm_cell (size_t i, size_t j, bool check_bounds=true, bool check_exists=true)
- void insert_cell (size_t i, size_t j, const Cell< Cell_Type > &v, bool check_bounds, bool)
- void insert_cell (size_t i, size_t j, Cell_Type v, bool check_bounds, bool)

- void swap_cells (size_t i0, size_t j0, size_t i1, size_t j1, bool check_bounds=true, int check_exists=CHECK::BOTH, int *report=nullptr)
- void toggle_cell (size_t i, size_t j, bool check_bounds=true, int check_exists=EXISTS::UKNOWN)
- void toggle_lock (size_t i, size_t j, bool check_bounds=true)

Column/row wise interchange

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

Arithmetic operators

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

Public Attributes

- bool visited = false

Friends

- class BArrayDenseCell< Cell_Type, Data_Type >
- class BArrayDenseCol< Cell_Type, Data_Type >
- class BArrayDenseCol_const< Cell_Type, Data_Type >
- class BArrayDenseRow< Cell_Type, Data_Type >
- class BArrayDenseRow_const< Cell_Type, Data_Type >

8.4.1 Detailed Description

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

Baseline class for binary arrays.

`BArrayDense` class objects are arbitrary dense-arrays. The data is stored internally in the `el` member, which can be accessed using the member function `get_data()`, by column.

Template Parameters

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

Definition at line 33 of file barraydense-bones.hpp.

8.4.2 Constructor & Destructor Documentation

8.4.2.1 BArrayDense() [1/6]

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

Zero-size array.

Definition at line 79 of file barraydense-bones.hpp.

8.4.2.2 BArrayDense() [2/6]

```
template<typename Cell_Type = bool, typename Data_Type = bool>
BArrayDense< Cell_Type, Data_Type >::BArrayDense (
            size_t N_,
            size_t M_,
            Cell_Type value = static_cast<Cell_Type>(0) )  [inline]
```

Empty array.

Definition at line 82 of file barraydense-bones.hpp.

8.4.2.3 BArrayDense() [3/6]

```
template<typename Cell_Type , typename Data_Type >
BArrayDense< Cell_Type, Data_Type >::BArrayDense (
            size_t N_,
            size_t M_,
            const std::vector< size_t > & source,
            const std::vector< size_t > & target,
            const std::vector< Cell_Type > & value,
            bool add = true )  [inline]
```

Edgelist with data.

Definition at line 35 of file barraydense-meat.hpp.

8.4.2.4 BArrayDense() [4/6]

```
template<typename Cell_Type , typename Data_Type >
BArrayDense< Cell_Type, Data_Type >::BArrayDense (
            size_t N_,
            size_t M_,
            const std::vector< size_t > & source,
            const std::vector< size_t > & target,
            bool add = true )  [inline]
```

Edgelist with no data (simpler)

Definition at line 86 of file barraydense-meat.hpp.

8.4.2.5 BArrayDense() [5/6]

```
template<typename Cell_Type , typename Data_Type >
BArrayDense< Cell_Type, Data_Type >::BArrayDense (
            const BArrayDense< Cell_Type, Data_Type > & Array_,
            bool copy_data = false )  [inline]
```

Copy constructor.

Definition at line 135 of file barraydense-meat.hpp.

8.4.2.6 BArrayDense() [6/6]

```
template<typename Cell_Type , typename Data_Type >
BArrayDense< Cell_Type, Data_Type >::BArrayDense (
            BArrayDense< Cell_Type, Data_Type > && x )  [inline], [noexcept]
```

Move operator.

Definition at line 229 of file barraydense-meat.hpp.

8.4.2.7 ~BArrayDense()

```
template<typename Cell_Type , typename Data_Type >
BArrayDense< Cell_Type, Data_Type >::~BArrayDense  [inline]
```

Definition at line 310 of file barraydense-meat.hpp.

8.4.3 Member Function Documentation

8.4.3.1 clear()

```
template<typename Cell_Type , typename Data_Type >
void BArrayDense< Cell_Type, Data_Type >::clear (
            bool hard = true )  [inline]
```

Definition at line 909 of file barraydense-meat.hpp.

8.4.3.2 col() [1/2]

```
template<typename Cell_Type , typename Data_Type >
BArrayDenseCol< Cell_Type, Data_Type > & BArrayDense< Cell_Type, Data_Type >::col (
            size_t j,
            bool check_bounds = true )  [inline]
```

Definition at line 502 of file barraydense-meat.hpp.

8.4.3.3 col() [2/2]

```
template<typename Cell_Type , typename Data_Type >
const BArrayDenseCol_const< Cell_Type, Data_Type > BArrayDense< Cell_Type, Data_Type >::col (
            size_t j,
            bool check_bounds = true ) const  [inline]
```

Definition at line 488 of file barraydense-meat.hpp.

8.4.3.4 colsum()

```
template<typename Cell_Type , typename Data_Type >
const Cell_Type BArrayDense< Cell_Type, Data_Type >::colsum (
            size_t i ) const  [inline]
```

Definition at line 1012 of file barraydense-meat.hpp.

8.4.3.5 D() [1/2]

```
template<typename Cell_Type , typename Data_Type >
Data_Type & BArrayDense< Cell_Type, Data_Type >::D  [inline]
```

Definition at line 345 of file barraydense-meat.hpp.

8.4.3.6 D() [2/2]

```
template<typename Cell_Type , typename Data_Type >
const Data_Type & BArrayDense< Cell_Type, Data_Type >::D  [inline]
```

Definition at line 350 of file barraydense-meat.hpp.

8.4.3.7 D_ptr() [1/2]

```
template<typename Cell_Type , typename Data_Type >
Data_Type * BArrayDense< Cell_Type, Data_Type >::D_ptr  [inline]
```

Definition at line 335 of file barraydense-meat.hpp.

8.4.3.8 D_ptr() [2/2]

```
template<typename Cell_Type , typename Data_Type >
const Data_Type * BArrayDense< Cell_Type, Data_Type >::D_ptr  [inline]
```

Definition at line 340 of file barraydense-meat.hpp.

8.4.3.9 default_val()

```
template<typename Cell_Type , typename Data_Type >
Cell< Cell_Type > BArrayDense< Cell_Type, Data_Type >::default_val  [inline]
```

Definition at line 575 of file barraydense-meat.hpp.

8.4.3.10 get_cell()

```
template<typename Cell_Type , typename Data_Type >
Cell_Type BArrayDense< Cell_Type, Data_Type >::get_cell (
             size_t i,
             size_t j,
             bool check_bounds = true ) const  [inline]
```

Definition at line 376 of file barraydense-meat.hpp.

8.4.3.11 get_col_vec() [1/2]

```
template<typename Cell_Type , typename Data_Type >
std::vector< Cell_Type > BArrayDense< Cell_Type, Data_Type >::get_col_vec (
            size_t i,
            bool check_bounds = true ) const  [inline]
```

Definition at line 424 of file barraydense-meat.hpp.

8.4.3.12 get_col_vec() [2/2]

```
template<typename Cell_Type , typename Data_Type >
void BArrayDense< Cell_Type, Data_Type >::get_col_vec (
            std::vector< Cell_Type > * x,
            size_t i,
            bool check_bounds = true ) const  [inline]
```

Definition at line 442 of file barraydense-meat.hpp.

8.4.3.13 get_data()

```
template<typename Cell_Type , typename Data_Type >
const std::vector< Cell_Type > & BArrayDense< Cell_Type, Data_Type >::get_data  [inline]
```

Definition at line 1002 of file barraydense-meat.hpp.

8.4.3.14 get_entries()

```
template<typename Cell_Type , typename Data_Type >
Entries< Cell_Type > BArrayDense< Cell_Type, Data_Type >::get_entries  [inline]
```

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>

Definition at line 514 of file barraydense-meat.hpp.

8.4.3.15 get_row_vec() [1/2]

```
template<typename Cell_Type , typename Data_Type >
std::vector< Cell_Type > BArrayDense< Cell_Type, Data_Type >::get_row_vec (
            size_t i,
            bool check_bounds = true ) const  [inline]
```

Definition at line 391 of file barraydense-meat.hpp.

8.4.3.16 get_row_vec() [2/2]

```
template<typename Cell_Type , typename Data_Type >
void BArrayDense< Cell_Type, Data_Type >::get_row_vec (
            std::vector< Cell_Type > * x,
            size_t i,
            bool check_bounds = true ) const  [inline]
```

Definition at line 409 of file barraydense-meat.hpp.

8.4.3.17 insert_cell() [1/2]

```
template<typename Cell_Type , typename Data_Type >
void BArrayDense< Cell_Type, Data_Type >::insert_cell (
            size_t i,
            size_t j,
            Cell_Type v,
            bool check_bounds,
            bool  )  [inline]
```

Definition at line 701 of file barraydense-meat.hpp.

8.4.3.18 insert_cell() [2/2]

```
template<typename Cell_Type , typename Data_Type >
void BArrayDense< Cell_Type, Data_Type >::insert_cell (
            size_t i,
            size_t j,
            const Cell< Cell_Type > & v,
            bool check_bounds,
            bool  )  [inline]
```

Definition at line 667 of file barraydense-meat.hpp.

8.4.3.19 is_dense()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
bool BArrayDense< Cell_Type, Data_Type >::is_dense ( ) const  [inline], [noexcept]
```

Definition at line 256 of file barraydense-bones.hpp.

8.4.3.20 is_empty()

```
template<typename Cell_Type , typename Data_Type >
bool BArrayDense< Cell_Type, Data_Type >::is_empty (
            size_t i,
            size_t j,
            bool check_bounds = true ) const  [inline]
```

Definition at line 543 of file barraydense-meat.hpp.

8.4.3.21 ncol()

```
template<typename Cell_Type , typename Data_Type >
size_t BArrayDense< Cell_Type, Data_Type >::ncol  [inline], [noexcept]
```

Definition at line 560 of file barraydense-meat.hpp.

8.4.3.22 nnozero()

```
template<typename Cell_Type , typename Data_Type >
size_t BArrayDense< Cell_Type, Data_Type >::nnozero  [inline], [noexcept]
```

Definition at line 564 of file barraydense-meat.hpp.

8.4.3.23 nrow()

```
template<typename Cell_Type , typename Data_Type >
size_t BArrayDense< Cell_Type, Data_Type >::nrow  [inline], [noexcept]
```

Definition at line 556 of file barraydense-meat.hpp.

8.4.3.24 operator()() [1/2]

```
template<typename Cell_Type , typename Data_Type >
BArrayDenseCell< Cell_Type, Data_Type > BArrayDense< Cell_Type, Data_Type >::operator() (
            size_t i,
            size_t j,
            bool check_bounds = true )  [inline]
```

Definition at line 619 of file barraydense-meat.hpp.

8.4.3.25 operator()() [2/2]

```
template<typename Cell_Type , typename Data_Type >
const Cell_Type BArrayDense< Cell_Type, Data_Type >::operator() (
            size_t i,
            size_t j,
            bool check_bounds = true ) const  [inline]
```

Definition at line 630 of file barraydense-meat.hpp.

8.4.3.26 operator*=()

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

8.4.3.27 operator+=() [1/3]

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

8.4.3.28 operator+=() [2/3]

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

8.4.3.29 operator+=() [3/3]

```
template<typename Cell_Type , typename Data_Type >
BArrayDense< Cell_Type, Data_Type > & BArrayDense< Cell_Type, Data_Type >::operator+= (
            const std::pair< size_t, size_t > & coords )  [inline]
```

Definition at line 580 of file barraydense-meat.hpp.

8.4.3.30 operator-=() [1/3]

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

8.4.3.31 operator-=() [2/3]

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

8.4.3.32 operator-=() [3/3]

```
template<typename Cell_Type , typename Data_Type >
BArrayDense< Cell_Type, Data_Type > & BArrayDense< Cell_Type, Data_Type >::operator-= (
            const std::pair< size_t, size_t > & coords )  [inline]
```

Definition at line 599 of file barraydense-meat.hpp.

8.4.3.33 operator/=()

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

8.4.3.34 operator=() [1/2]

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

Move assignment.

Definition at line 247 of file barraydense-meat.hpp.

8.4.3.35 operator=() [2/2]

```
template<typename Cell_Type , typename Data_Type >
BArrayDense< Cell_Type, Data_Type > & BArrayDense< Cell_Type, Data_Type >::operator= (
            const BArrayDense< Cell_Type, Data_Type > & Array_ )  [inline]
```

Assignment constructor.

Definition at line 179 of file barraydense-meat.hpp.

8.4.3.36 operator==()

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

Definition at line 291 of file barraydense-meat.hpp.

8.4.3.37 out_of_range()

```
template<typename Cell_Type , typename Data_Type >
void BArrayDense< Cell_Type, Data_Type >::out_of_range (
            size_t i,
            size_t j ) const  [inline]
```

Definition at line 355 of file barraydense-meat.hpp.

8.4.3.38 print()

```
template<typename Cell_Type , typename Data_Type >
void BArrayDense< Cell_Type, Data_Type >::print (
            const char * fmt = nullptr,
             ... )  const  [inline]
```

Definition at line 968 of file barraydense-meat.hpp.

8.4.3.39 reserve()

```
template<typename Cell_Type , typename Data_Type >
void BArrayDense< Cell_Type, Data_Type >::reserve  [inline]
```

Definition at line 959 of file barraydense-meat.hpp.

8.4.3.40 resize()

```
template<typename Cell_Type , typename Data_Type >
void BArrayDense< Cell_Type, Data_Type >::resize (
            size_t N_,
            size_t M_ )  [inline]
```

Definition at line 923 of file barraydense-meat.hpp.

8.4.3.41 rm_cell()

```
template<typename Cell_Type , typename Data_Type >
void BArrayDense< Cell_Type, Data_Type >::rm_cell (
            size_t i,
            size_t j,
            bool check_bounds = true,
            bool check_exists = true )  [inline]
```

Definition at line 644 of file barraydense-meat.hpp.

8.4.3.42 row() [1/2]

```
template<typename Cell_Type , typename Data_Type >
BArrayDenseRow< Cell_Type, Data_Type > & BArrayDense< Cell_Type, Data_Type >::row (
            size_t i,
            bool check_bounds = true )  [inline]
```

Definition at line 474 of file barraydense-meat.hpp.

8.4.3.43 row() [2/2]

```
template<typename Cell_Type , typename Data_Type >
const BArrayDenseRow_const< Cell_Type, Data_Type > BArrayDense< Cell_Type, Data_Type >::row (
            size_t i,
            bool check_bounds = true ) const  [inline]
```

Definition at line 461 of file barraydense-meat.hpp.

8.4.3.44 rowsum()

```
template<typename Cell_Type , typename Data_Type >
const Cell_Type BArrayDense< Cell_Type, Data_Type >::rowsum (
            size_t i ) const  [inline]
```

Definition at line 1007 of file barraydense-meat.hpp.

8.4.3.45 set_data()

```
template<typename Cell_Type , typename Data_Type >
void BArrayDense< Cell_Type, Data_Type >::set_data (
            Data_Type * data_,
            bool delete_data_ = false )  [inline]
```

Set the data object.

Parameters

*data_*	
*delete_↩ data_*	

Definition at line 319 of file barraydense-meat.hpp.

8.4.3.46 swap_cells()

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

Definition at line 732 of file barraydense-meat.hpp.

8.4.3.47 swap_cols()

```
template<typename Cell_Type , typename Data_Type >
void BArrayDense< Cell_Type, Data_Type >::swap_cols (
            size_t j0,
            size_t j1,
            bool check_bounds = true )  [inline]
```

Definition at line 816 of file barraydense-meat.hpp.

8.4.3.48 swap_rows()

```
template<typename Cell_Type , typename Data_Type >
void BArrayDense< Cell_Type, Data_Type >::swap_rows (
            size_t i0,
            size_t i1,
            bool check_bounds = true )  [inline]
```

Definition at line 788 of file barraydense-meat.hpp.

8.4.3.49 toggle_cell()

```
template<typename Cell_Type , typename Data_Type >
void BArrayDense< Cell_Type, Data_Type >::toggle_cell (
            size_t i,
            size_t j,
            bool check_bounds = true,
            int check_exists = EXISTS::UKNOWN )  [inline]
```

Definition at line 769 of file barraydense-meat.hpp.

8.4.3.50 toggle_lock()

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

8.4.3.51 transpose()

```
template<typename Cell_Type , typename Data_Type >
void BArrayDense< Cell_Type, Data_Type >::transpose  [inline]
```

Definition at line 881 of file barraydense-meat.hpp.

8.4.3.52 zero_col()

```
template<typename Cell_Type , typename Data_Type >
void BArrayDense< Cell_Type, Data_Type >::zero_col (
            size_t j,
            bool check_bounds = true )  [inline]
```

Definition at line 862 of file barraydense-meat.hpp.

8.4.3.53 zero_row()

```
template<typename Cell_Type , typename Data_Type >
void BArrayDense< Cell_Type, Data_Type >::zero_row (
            size_t i,
            bool check_bounds = true )  [inline]
```

Definition at line 843 of file barraydense-meat.hpp.

8.4.4 Friends And Related Function Documentation

8.4.4.1 BArrayDenseCell< Cell_Type, Data_Type >

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

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

8.4.4.2 BArrayDenseCol< Cell_Type, Data_Type >

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

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

8.4.4.3 BArrayDenseCol_const< Cell_Type, Data_Type >

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

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

8.4.4.4 BArrayDenseRow< Cell_Type, Data_Type >

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

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

8.4.4.5 BArrayDenseRow_const< Cell_Type, Data_Type >

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

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

8.4.5 Member Data Documentation

8.4.5.1 visited

```
template<typename Cell_Type = bool, typename Data_Type = bool>
bool BArrayDense< 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 63 of file barraydense-bones.hpp.

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

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

8.5 BArrayDenseCell< Cell_Type, Data_Type > Class Template Reference

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

Public Member Functions

- BArrayDenseCell (BArrayDense< Cell_Type, Data_Type > *Array_, size_t i_, size_t j_, bool check_↩ bounds=true)
- BArrayDenseCell< Cell_Type, Data_Type > & operator= (const BArrayDenseCell< Cell_Type, Data_Type > &other)
- ~BArrayDenseCell ()
- 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

Friends

- class BArrayDense< Cell_Type, Data_Type >
- class BArrayDenseCol< Cell_Type, Data_Type >
- class BArrayDenseCol_const< Cell_Type, Data_Type >

8.5.1 Detailed Description

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

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

8.5.2 Constructor & Destructor Documentation

8.5.2.1 BArrayDenseCell()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
BArrayDenseCell< Cell_Type, Data_Type >::BArrayDenseCell (
            BArrayDense< Cell_Type, Data_Type > * Array_,
            size_t i_,
            size_t j_,
            bool check_bounds = true )  [inline]
```

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

8.5.2.2 ~BArrayDenseCell()

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

Definition at line 56 of file barraydensecell-bones.hpp.

8.5.3 Member Function Documentation

8.5.3.1 operator Cell_Type()

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

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

8.5.3.2 operator*=()

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

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

8.5.3.3 operator+=()

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

Definition at line 48 of file barraydensecell-meat.hpp.

8.5.3.4 operator-=()

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

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

8.5.3.5 operator/=()

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

Definition at line 95 of file barraydensecell-meat.hpp.

8.5.3.6 operator=() [1/2]

```
template<typename Cell_Type , typename Data_Type >
BArrayDenseCell< Cell_Type, Data_Type > & BArrayDenseCell< Cell_Type, Data_Type >::operator=
(
            const BArrayDenseCell< Cell_Type, Data_Type > & other )  [inline]
```

Definition at line 9 of file barraydensecell-meat.hpp.

8.5.3.7 operator=() [2/2]

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

Definition at line 31 of file barraydensecell-meat.hpp.

8.5.3.8 operator==()

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

Definition at line 117 of file barraydensecell-meat.hpp.

8.5.4 Friends And Related Function Documentation

8.5.4.1 BArrayDense< Cell_Type, Data_Type >

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

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

8.5.4.2 BArrayDenseCol< Cell_Type, Data_Type >

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

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

8.5.4.3 BArrayDenseCol_const< Cell_Type, Data_Type >

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

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

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

- include/barry/barraydense-bones.hpp
- include/barry/barraydensecell-bones.hpp
- include/barry/barraydensecell-meat.hpp

8.6 BArrayDenseCell_const< Cell_Type, Data_Type > Class Template Reference

8.6.1 Detailed Description

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

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

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

- include/barry/barraydense-bones.hpp

8.7 BArrayDenseCol< Cell_Type, Data_Type > Class Template Reference

`#include <barraydensecol-bones.hpp>`

Public Member Functions

- BArrayDenseCol (BArrayDense< Cell_Type, Data_Type > &array_, size_t j)
- Col_type< Cell_Type >::iterator & begin ()
- Col_type< Cell_Type >::iterator & end ()
- size_t size () const noexcept
- std::pair< size_t, Cell_Type * > & operator() (size_t i)

Friends

- class BArrayDense< Cell_Type, Data_Type >
- class BArrayDenseCell< Cell_Type, Data_Type >
- class BArrayDenseCell_const< Cell_Type, Data_Type >

8.7.1 Detailed Description

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

Definition at line 9 of file barraydensecol-bones.hpp.

8.7.2 Constructor & Destructor Documentation

8.7.2.1 BArrayDenseCol()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
BArrayDenseCol< Cell_Type, Data_Type >::BArrayDenseCol (
            BArrayDense< Cell_Type, Data_Type > & array_,
            size_t j )  [inline]
```

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

8.7.3 Member Function Documentation

8.7.3.1 begin()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
Col_type<Cell_Type>::iterator& BArrayDenseCol< Cell_Type, Data_Type >::begin ( )  [inline]
```

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

8.7.3.2 end()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
Col_type<Cell_Type>::iterator& BArrayDenseCol< Cell_Type, Data_Type >::end ( )  [inline]
```

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

8.7.3.3 operator()()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
std::pair<size_t,Cell_Type*>& BArrayDenseCol< Cell_Type, Data_Type >::operator() (
            size_t i )  [inline]
```

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

8.7.3.4 size()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
size_t BArrayDenseCol< Cell_Type, Data_Type >::size ( ) const  [inline], [noexcept]
```

Definition at line 56 of file barraydensecol-bones.hpp.

8.7.4 Friends And Related Function Documentation

8.7.4.1 BArrayDense< Cell_Type, Data_Type >

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

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

8.7.4.2 BArrayDenseCell< Cell_Type, Data_Type >

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

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

8.7.4.3 BArrayDenseCell_const< Cell_Type, Data_Type >

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

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

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

- include/barry/barraydense-bones.hpp
- include/barry/barraydensecol-bones.hpp

8.8 BArrayDenseCol_const< Cell_Type, Data_Type > Class Template Reference

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

Public Member Functions

- BArrayDenseCol_const (const BArrayDense< Cell_Type, Data_Type > &array_, size_t j)
- Col_type< Cell_Type >::iterator begin ()
- Col_type< Cell_Type >::iterator end ()
- size_t size () const noexcept
- const std::pair< size_t, Cell_Type * > operator() (size_t i) const

Friends

- class BArrayDenseCell< Cell_Type, Data_Type >
- class BArrayDenseCell_const< Cell_Type, Data_Type >

8.8.1 Detailed Description

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

Definition at line 71 of file barraydensecol-bones.hpp.

8.8.2 Constructor & Destructor Documentation

8.8.2.1 BArrayDenseCol_const()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
BArrayDenseCol_const< Cell_Type, Data_Type >::BArrayDenseCol_const (
            const BArrayDense< Cell_Type, Data_Type > & array_,
            size_t j )  [inline]
```

Definition at line 80 of file barraydensecol-bones.hpp.

8.8.3 Member Function Documentation

8.8.3.1 begin()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
Col_type<Cell_Type>::iterator BArrayDenseCol_const< Cell_Type, Data_Type >::begin ( )  [inline]
```

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

8.8.3.2 end()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
Col_type<Cell_Type>::iterator BArrayDenseCol_const< Cell_Type, Data_Type >::end ( )  [inline]
```

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

8.8.3.3 operator()()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
const std::pair<size_t,Cell_Type*> BArrayDenseCol_const< Cell_Type, Data_Type >::operator() (
            size_t i ) const  [inline]
```

Definition at line 112 of file barraydensecol-bones.hpp.

8.8.3.4 size()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
size_t BArrayDenseCol_const< Cell_Type, Data_Type >::size ( ) const  [inline], [noexcept]
```

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

8.8.4 Friends And Related Function Documentation

8.8.4.1 BArrayDenseCell< Cell_Type, Data_Type >

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

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

8.8.4.2 BArrayDenseCell_const< Cell_Type, Data_Type >

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

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

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

- include/barry/barraydense-bones.hpp
- include/barry/barraydensecol-bones.hpp

8.9 BArrayDenseRow< Cell_Type, Data_Type > Class Template Reference

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

Public Member Functions

- BArrayDenseRow (BArrayDense< Cell_Type, Data_Type > &array_, size_t i)
- Row_type< Cell_Type >::iterator & begin ()
- Row_type< Cell_Type >::iterator & end ()
- size_t size () const noexcept
- std::pair< size_t, Cell< Cell_Type > > & operator() (size_t i)

Friends

- class BArrayDense< Cell_Type, Data_Type >
- class BArrayDenseCell< Cell_Type, Data_Type >
- class BArrayDenseCell_const< Cell_Type, Data_Type >

8.9.1 Detailed Description

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

Definition at line 9 of file barraydenserow-bones.hpp.

8.9.2 Constructor & Destructor Documentation

8.9.2.1 BArrayDenseRow()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
BArrayDenseRow< Cell_Type, Data_Type >::BArrayDenseRow (
            BArrayDense< Cell_Type, Data_Type > & array_,
            size_t i )  [inline]
```

Definition at line 40 of file barraydenserow-bones.hpp.

8.9.3 Member Function Documentation

8.9.3.1 begin()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
Row_type<Cell_Type>::iterator& BArrayDenseRow< Cell_Type, Data_Type >::begin ( )  [inline]
```

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

8.9.3.2 end()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
Row_type<Cell_Type>::iterator& BArrayDenseRow< Cell_Type, Data_Type >::end ( )  [inline]
```

Definition at line 53 of file barraydenserow-bones.hpp.

8.9.3.3 operator()()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
std::pair<size_t,Cell<Cell_Type> >& BArrayDenseRow< Cell_Type, Data_Type >::operator() (
            size_t i )  [inline]
```

Definition at line 69 of file barraydenserow-bones.hpp.

8.9.3.4 size()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
size_t BArrayDenseRow< Cell_Type, Data_Type >::size ( ) const  [inline], [noexcept]
```

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

8.9.4 Friends And Related Function Documentation

8.9.4.1 BArrayDense< Cell_Type, Data_Type >

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

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

8.9.4.2 BArrayDenseCell< Cell_Type, Data_Type >

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

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

8.9.4.3 BArrayDenseCell_const< Cell_Type, Data_Type >

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

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

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

- include/barry/barraydense-bones.hpp
- include/barry/barraydenserow-bones.hpp

8.10 BArrayDenseRow_const< Cell_Type, Data_Type > Class Template Reference

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

Public Member Functions

- BArrayDenseRow_const (const BArrayDense< Cell_Type, Data_Type > &array_, size_t i)
- Row_type< Cell_Type >::const_iterator begin () const
- Row_type< Cell_Type >::const_iterator end () const
- size_t size () const noexcept
- const std::pair< size_t, Cell< Cell_Type > > operator() (size_t i) const

Friends

- class BArrayDenseCell< Cell_Type, Data_Type >
- class BArrayDenseCell_const< Cell_Type, Data_Type >

8.10.1 Detailed Description

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

Definition at line 80 of file barraydenserow-bones.hpp.

8.10.2 Constructor & Destructor Documentation

8.10.2.1 BArrayDenseRow_const()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
BArrayDenseRow_const< Cell_Type, Data_Type >::BArrayDenseRow_const (
            const BArrayDense< Cell_Type, Data_Type > & array_,
            size_t i )  [inline]
```

Definition at line 89 of file barraydenserow-bones.hpp.

8.10.3 Member Function Documentation

8.10.3.1 begin()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
Row_type< Cell_Type >::const_iterator BArrayDenseRow_const< Cell_Type, Data_Type >::begin ( )
const  [inline]
```

Definition at line 108 of file barraydenserow-bones.hpp.

8.10.3.2 end()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
Row_type< Cell_Type >::const_iterator BArrayDenseRow_const< Cell_Type, Data_Type >::end ( )
const  [inline]
```

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

8.10.3.3 operator()()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
const std::pair<size_t,Cell<Cell_Type> > BArrayDenseRow_const< Cell_Type, Data_Type >↩
::operator() (
            size_t i ) const  [inline]
```

Definition at line 123 of file barraydenserow-bones.hpp.

8.10.3.4 size()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
size_t BArrayDenseRow_const< Cell_Type, Data_Type >::size ( ) const  [inline], [noexcept]
```

Definition at line 118 of file barraydenserow-bones.hpp.

8.10.4 Friends And Related Function Documentation

8.10.4.1 BArrayDenseCell< Cell_Type, Data_Type >

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

Definition at line 69 of file barraydenserow-bones.hpp.

8.10.4.2 BArrayDenseCell_const< Cell_Type, Data_Type >

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

Definition at line 69 of file barraydenserow-bones.hpp.

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

- include/barry/barraydense-bones.hpp
- include/barry/barraydenserow-bones.hpp

8.11 BArrayRow< Cell_Type, Data_Type > Class Template Reference

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

Public Member Functions

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

8.11.1 Detailed Description

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

Definition at line 5 of file barrayrow-bones.hpp.

8.11.2 Constructor & Destructor Documentation

8.11.2.1 BArrayRow()

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

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

8.11.2.2 ~BArrayRow()

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

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

8.11.3 Member Function Documentation

8.11.3.1 operator BArrayRow< Cell_Type, Data_Type >()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
BArrayRow< Cell_Type, Data_Type >::operator BArrayRow< Cell_Type, Data_Type > ( ) const
```

8.11.3.2 operator*=()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
void BArrayRow< Cell_Type, Data_Type >::operator*= (
            const BArrayRow< Cell_Type, Data_Type > & val )
```

8.11.3.3 operator+=()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
void BArrayRow< Cell_Type, Data_Type >::operator+= (
            const BArrayRow< Cell_Type, Data_Type > & val )
```

8.11.3.4 operator-=()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
void BArrayRow< Cell_Type, Data_Type >::operator-= (
            const BArrayRow< Cell_Type, Data_Type > & val )
```

8.11.3.5 operator/=()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
void BArrayRow< Cell_Type, Data_Type >::operator/= (
            const BArrayRow< Cell_Type, Data_Type > & val )
```

8.11.3.6 operator=()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
void BArrayRow< Cell_Type, Data_Type >::operator= (
            const BArrayRow< Cell_Type, Data_Type > & val )
```

8.11.3.7 operator==()

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

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

- include/barry/barrayrow-bones.hpp

8.12 BArrayRow_const< Cell_Type, Data_Type > Class Template Reference

`#include <barrayrow-bones.hpp>`

Public Member Functions

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

8.12.1 Detailed Description

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

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

8.12.2 Constructor & Destructor Documentation

8.12.2.1 BArrayRow_const()

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

Definition at line 49 of file barrayrow-bones.hpp.

8.12.2.2 ~BArrayRow_const()

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

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

8.12.3 Member Function Documentation

8.12.3.1 operator BArrayRow_const< Cell_Type, Data_Type >()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
BArrayRow_const< Cell_Type, Data_Type >::operator BArrayRow_const< Cell_Type, Data_Type > ( )
const
```

8.12.3.2 operator"!=()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
bool BArrayRow_const< Cell_Type, Data_Type >::operator!= (
            const BArrayRow_const< Cell_Type, Data_Type > & val ) const
```

8.12.3.3 operator<()

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

8.12.3.4 operator<=()

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

8.12.3.5 operator==()

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

8.12.3.6 operator>()

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

8.12.3.7 operator>=()

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

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

- include/barry/barrayrow-bones.hpp

8.13 BArrayVector< Cell_Type, Data_Type > Class Template Reference

Row or column of a BArray

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

Public Member Functions

- BArrayVector (BArray< Cell_Type, Data_Type > *Array_, size_t &dim_ size_t &i_, bool check_bounds=true)
  Construct a new BArrayVector object.
- ~BArrayVector ()
- bool is_row () const noexcept
- bool is_col () const noexcept
- size_t size () const noexcept
- std::vector< Cell_Type >::const_iterator begin () noexcept
- std::vector< Cell_Type >::const_iterator end () noexcept
- 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 std::vector< Cell_Type > () const
- bool operator== (const Cell_Type &val) const

8.13.1 Detailed Description

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

Row or column of a `BArray`

Template Parameters

Cell_Type	
Data_Type	

Definition at line 11 of file barrayvector-bones.hpp.

8.13.2 Constructor & Destructor Documentation

8.13.2.1 BArrayVector()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
BArrayVector< Cell_Type, Data_Type >::BArrayVector (
            BArray< Cell_Type, Data_Type > * Array_,
            size_t &dim_ size_t & i_,
            bool check_bounds = true )  [inline]
```

Construct a new BArrayVector object.

Parameters

*Array_*	Pointer to a `BArray` object
*dim_*	Dimension. 0 means row and 1 means column.
*i_*	Element to point.
check_bounds	When `true`, check boundaries.

Definition at line 32 of file barrayvector-bones.hpp.

8.13.2.2 ~BArrayVector()

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

Definition at line 53 of file barrayvector-bones.hpp.

8.13.3 Member Function Documentation

8.13.3.1 begin()

```
template<typename Cell_Type , typename Data_Type >
std::vector< Cell_Type >::const_iterator BArrayVector< Cell_Type, Data_Type >::begin  [inline],
[noexcept]
```

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

8.13.3.2 end()

```
template<typename Cell_Type , typename Data_Type >
std::vector< Cell_Type >::const_iterator BArrayVector< Cell_Type, Data_Type >::end  [inline],
[noexcept]
```

Definition at line 64 of file barrayvector-meat.hpp.

8.13.3.3 is_col()

```
template<typename Cell_Type , typename Data_Type >
bool BArrayVector< Cell_Type, Data_Type >::is_col  [inline], [noexcept]
```

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

8.13.3.4 is_row()

```
template<typename Cell_Type , typename Data_Type >
bool BArrayVector< Cell_Type, Data_Type >::is_row  [inline], [noexcept]
```

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

8.13.3.5 operator std::vector< Cell_Type >()

```
template<typename Cell_Type , typename Data_Type >
BArrayVector< Cell_Type, Data_Type >::operator std::vector< Cell_Type >  [inline]
```

Definition at line 175 of file barrayvector-meat.hpp.

8.13.3.6 operator*=()

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

Definition at line 133 of file barrayvector-meat.hpp.

8.13.3.7 operator+=()

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

Definition at line 91 of file barrayvector-meat.hpp.

8.13.3.8 operator-=()

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

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

8.13.3.9 operator/=()

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

Definition at line 154 of file barrayvector-meat.hpp.

8.13.3.10 operator=()

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

Definition at line 69 of file barrayvector-meat.hpp.

8.13.3.11 operator==()

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

Definition at line 185 of file barrayvector-meat.hpp.

8.13.3.12 size()

```
template<typename Cell_Type , typename Data_Type >
size_t BArrayVector< Cell_Type, Data_Type >::size  [inline], [noexcept]
```

Definition at line 39 of file barrayvector-meat.hpp.

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

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

8.14 BArrayVector_const< Cell_Type, Data_Type > Class Template Reference

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

Public Member Functions

- BArrayVector_const (const BArray< Cell_Type, Data_Type > *Array_, size_t &dim_ size_t &i_, bool check↩ _bounds=true)
- ~BArrayVector_const ()
- bool is_row () const noexcept
- bool is_col () const noexcept
- size_t size () const noexcept
- std::vector< Cell_Type >::const_iterator begin () noexcept
- std::vector< Cell_Type >::const_iterator end () noexcept
- operator std::vector< 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

8.14.1 Detailed Description

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

Definition at line 73 of file barrayvector-bones.hpp.

8.14.2 Constructor & Destructor Documentation

8.14.2.1 BArrayVector_const()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
BArrayVector_const< Cell_Type, Data_Type >::BArrayVector_const (
            const BArray< Cell_Type, Data_Type > * Array_,
            size_t &dim_ size_t & i_,
            bool check_bounds = true )  [inline]
```

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

8.14.2.2 ~BArrayVector_const()

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

Definition at line 108 of file barrayvector-bones.hpp.

8.14.3 Member Function Documentation

8.14.3.1 begin()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
std::vector< Cell_Type >::const_iterator BArrayVector_const< Cell_Type, Data_Type >::begin (
)  [noexcept]
```

8.14.3.2 end()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
std::vector< Cell_Type >::const_iterator BArrayVector_const< Cell_Type, Data_Type >::end ( )
[noexcept]
```

8.14.3.3 is_col()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
bool BArrayVector_const< Cell_Type, Data_Type >::is_col ( ) const  [noexcept]
```

8.14.3.4 is_row()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
bool BArrayVector_const< Cell_Type, Data_Type >::is_row ( ) const  [noexcept]
```

8.14.3.5 operator std::vector< Cell_Type >()

```
template<typename Cell_Type , typename Data_Type >
BArrayVector_const< Cell_Type, Data_Type >::operator std::vector< Cell_Type >  [inline]
```

Definition at line 212 of file barrayvector-meat.hpp.

8.14.3.6 operator"!=()

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

Definition at line 249 of file barrayvector-meat.hpp.

8.14.3.7 operator<()

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

Definition at line 254 of file barrayvector-meat.hpp.

8.14.3.8 operator<=()

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

Definition at line 281 of file barrayvector-meat.hpp.

8.14.3.9 operator==()

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

Definition at line 222 of file barrayvector-meat.hpp.

8.14.3.10 operator>()

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

Definition at line 308 of file barrayvector-meat.hpp.

8.14.3.11 operator>=()

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

Definition at line 315 of file barrayvector-meat.hpp.

8.14.3.12 size()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
size_t BArrayVector_const< Cell_Type, Data_Type >::size ( ) const  [noexcept]
```

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

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

8.15 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, bool active_=true)
- ~Cell ()
- Cell (const Cell< Cell_Type > &arg)
- Cell< Cell_Type > & operator= (const 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
- bool operator== (const Cell< Cell_Type > &rhs) const
- bool operator!= (const Cell< Cell_Type > &rhs) const
- void add (double x)
- void add (size_t x)
- void add (int x)
- Cell ()
- Cell ()
- Cell ()

Public Attributes

- Cell_Type value
- bool visited
- bool active

8.15.1 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 10 of file cell-bones.hpp.

8.15.2 Constructor & Destructor Documentation

8.15.2.1 Cell() [1/7]

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

8.15.2.2 Cell() [2/7]

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

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

8.15.2.3 ~Cell()

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

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

8.15.2.4 Cell() [3/7]

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

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

8.15.2.5 Cell() [4/7]

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

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

8.15.2.6 Cell() [5/7]

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

Definition at line 62 of file cell-meat.hpp.

8.15.2.7 Cell() [6/7]

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

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

8.15.2.8 Cell() [7/7]

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

Definition at line 64 of file cell-meat.hpp.

8.15.3 Member Function Documentation

8.15.3.1 add() [1/4]

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

8.15.3.2 add() [2/4]

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

Definition at line 42 of file cell-meat.hpp.

8.15.3.3 add() [3/4]

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

Definition at line 52 of file cell-meat.hpp.

8.15.3.4 add() [4/4]

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

Definition at line 47 of file cell-meat.hpp.

8.15.3.5 operator Cell_Type()

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

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

8.15.3.6 operator"!=()

```
template<typename Cell_Type >
bool Cell< Cell_Type >::operator!= (
            const Cell< Cell_Type > & rhs ) const
```

Definition at line 31 of file cell-meat.hpp.

8.15.3.7 operator=() [1/2]

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

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

8.15.3.8 operator=() [2/2]

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

Definition at line 5 of file cell-meat.hpp.

8.15.3.9 operator==()

```
template<typename Cell_Type >
bool Cell< Cell_Type >::operator== (
            const Cell< Cell_Type > & rhs ) const
```

Definition at line 21 of file cell-meat.hpp.

8.15.4 Member Data Documentation

8.15.4.1 active

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

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

8.15.4.2 value

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

Definition at line 12 of file cell-bones.hpp.

8.15.4.3 visited

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

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

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

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

8.16 Cell_const< Cell_Type > Class Template Reference

8.16.1 Detailed Description

template<typename Cell_Type>
class Cell_const< Cell_Type >

Definition at line 8 of file barray-meat.hpp.

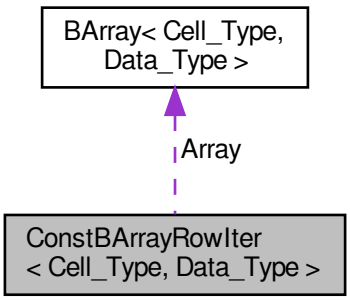
The documentation for this class was generated from the following file:

- include/barry/barray-meat.hpp

8.17 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

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

8.17.1 Detailed Description

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

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

8.17.2 Constructor & Destructor Documentation

8.17.2.1 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.

8.17.2.2 ~ConstBArrayRowIter()

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

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

8.17.3 Member Data Documentation

8.17.3.1 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.

8.17.3.2 current_col

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

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

8.17.3.3 current_row

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

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

8.17.3.4 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

8.18 Counter< Array_Type, Data_Type > Class Template Reference

A counter function based on change statistics.

`#include <counters-bones.hpp>`

Public Member Functions

- ~Counter ()
- double count (Array_Type &Array, size_t i, size_t j)
- double init (Array_Type &Array, size_t i, size_t j)
- std::string get_name () const
- std::string get_description () const

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_, Hasher_fun_type< Array_Type, Data_Type > hasher_fun_, Data_Type data_, std::string name_="", std::string desc_="")
- Counter (const Counter< Array_Type, Data_Type > &counter_)

  Copy constructor.
- Counter (Counter< Array_Type, Data_Type > &&counter_) noexcept

  Move constructor.
- Counter< Array_Type, Data_Type > operator= (const Counter< Array_Type, Data_Type > &counter_)

  Copy assignment.
- Counter< Array_Type, Data_Type > & operator= (Counter< Array_Type, Data_Type > &&counter_) noexcept

  Move assignment.

- void set_hasher (Hasher_fun_type< Array_Type, Data_Type > fun)

  Get and set the hasher function.
- Hasher_fun_type< Array_Type, Data_Type > get_hasher ()

Public Attributes

- Counter_fun_type< Array_Type, Data_Type > count_fun
- Counter_fun_type< Array_Type, Data_Type > init_fun
- Hasher_fun_type< Array_Type, Data_Type > hasher_fun
- Data_Type data
- std::string name = ""
- std::string desc = ""

8.18.1 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 35 of file counters-bones.hpp.

8.18.2 Constructor & Destructor Documentation

8.18.2.1 Counter() [1/4]

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

Definition at line 57 of file counters-bones.hpp.

8.18.2.2 Counter() [2/4]

```
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_,
            Hasher_fun_type< Array_Type, Data_Type > hasher_fun_,
            Data_Type data_,
            std::string name_ = "",
            std::string desc_ = "" )  [inline]
```

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

8.18.2.3 Counter() [3/4]

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

Copy constructor.

8.18.2.4 Counter() [4/4]

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

Move constructor.

8.18.2.5 ~Counter()

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

Definition at line 75 of file counters-bones.hpp.

8.18.3 Member Function Documentation

8.18.3.1 count()

```
template<typename Array_Type = BArray<>, typename Data_Type = bool>
double Counter< Array_Type, Data_Type >::count (
            Array_Type & Array,
            size_t i,
            size_t j )
```

8.18.3.2 get_description()

```
template<typename Array_Type = BArray<>, typename Data_Type = bool>
std::string Counter< Array_Type, Data_Type >::get_description ( ) const
```

8.18.3.3 get_hasher()

```
template<typename Array_Type = BArray<>, typename Data_Type = bool>
Hasher_fun_type<Array_Type,Data_Type> Counter< Array_Type, Data_Type >::get_hasher ( )
```

8.18.3.4 get_name()

```
template<typename Array_Type = BArray<>, typename Data_Type = bool>
std::string Counter< Array_Type, Data_Type >::get_name ( ) const
```

8.18.3.5 init()

```
template<typename Array_Type = BArray<>, typename Data_Type = bool>
double Counter< Array_Type, Data_Type >::init (
            Array_Type & Array,
            size_t i,
            size_t j )
```

8.18.3.6 operator=() [1/2]

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

Copy assignment.

8.18.3.7 operator=() [2/2]

```
template<typename Array_Type = BArray<>, typename Data_Type = bool>
Counter<Array_Type,Data_Type>& Counter< Array_Type, Data_Type >::operator= (
            Counter< Array_Type, Data_Type > && counter_ )  [noexcept]
```

Move assignment.

8.18.3.8 set_hasher()

```
template<typename Array_Type = BArray<>, typename Data_Type = bool>
void Counter< Array_Type, Data_Type >::set_hasher (
            Hasher_fun_type< Array_Type, Data_Type > fun )
```

Get and set the hasher function.

The hasher function is used to characterize the support of the array. This way, if possible, the support enumeration is recycled.

Parameters

fun	

8.18.4 Member Data Documentation

8.18.4.1 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 38 of file counters-bones.hpp.

8.18.4.2 data

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

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

8.18.4.3 desc

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

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

8.18.4.4 hasher_fun

```
template<typename Array_Type = BArray<>, typename Data_Type = bool>
Hasher_fun_type<Array_Type,Data_Type> Counter< Array_Type, Data_Type >::hasher_fun
```

Definition at line 40 of file counters-bones.hpp.

8.18.4.5 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 39 of file counters-bones.hpp.

8.18.4.6 name

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

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

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

- include/barry/counters-bones.hpp

8.19 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_)

  Copy constructor.
- Counters (Counters< Array_Type, Data_Type > &&counters_) noexcept

  Move constructor.
- Counters< Array_Type, Data_Type > operator= (const Counters< Array_Type, Data_Type > &counter_)

  Copy assignment constructor.
- Counters< Array_Type, Data_Type > & operator= (Counters< Array_Type, Data_Type > &&counter_) noexcept

  Move assignment constructor.
- Counter< Array_Type, Data_Type > & operator[ ] (size_t 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_fun_type< Array_Type, Data_Type > count_fun_, Counter_fun_type< Array_Type, Data_Type > init_fun_, Hasher_fun_type< Array_Type, Data_Type > hasher_fun_, Data_Type data_, std::string name_="", std::string desc_="")
- std::vector< std::string > get_names () const
- std::vector< std::string > get_descriptions () const
- std::vector< double > gen_hash (const Array_Type &array, bool add_dims=true)

  Generates a hash for the given array according to the counters.
- void add_hash (Hasher_fun_type< Array_Type, Data_Type > fun_)

8.19.1 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 108 of file counters-bones.hpp.

8.19.2 Constructor & Destructor Documentation

8.19.2.1 Counters() [1/3]

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

8.19.2.2 ~Counters()

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

Definition at line 120 of file counters-bones.hpp.

8.19.2.3 Counters() [2/3]

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

Copy constructor.

Parameters

*counter_*	

8.19.2.4 Counters() [3/3]

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

Move constructor.

Parameters

*counters_*	

8.19.3 Member Function Documentation

8.19.3.1 add_counter() [1/2]

```
template<typename Array_Type = BArray<>, typename Data_Type = bool>
void Counters< Array_Type, Data_Type >::add_counter (
            Counter< Array_Type, Data_Type > counter )
```

8.19.3.2 add_counter() [2/2]

```
template<typename Array_Type = BArray<>, typename Data_Type = bool>
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_,
            Hasher_fun_type< Array_Type, Data_Type > hasher_fun_,
            Data_Type data_,
            std::string name_ = "",
            std::string desc_ = "" )
```

8.19.3.3 add_hash()

```
template<typename Array_Type = BArray<>, typename Data_Type = bool>
void Counters< Array_Type, Data_Type >::add_hash (
            Hasher_fun_type< Array_Type, Data_Type > fun_ )
```

8.19.3.4 gen_hash()

```
template<typename Array_Type = BArray<>, typename Data_Type = bool>
std::vector< double > Counters< Array_Type, Data_Type >::gen_hash (
            const Array_Type & array,
            bool add_dims = true )
```

Generates a hash for the given array according to the counters.

Parameters

array	
add_dims	When `true` (default) the dimmension of the array will be added to the hash.

Returns

std::vector< double > That can be hashed later.

8.19.3.5 get_descriptions()

```
template<typename Array_Type = BArray<>, typename Data_Type = bool>
std::vector< std::string > Counters< Array_Type, Data_Type >::get_descriptions ( ) const
```

8.19.3.6 get_names()

```
template<typename Array_Type = BArray<>, typename Data_Type = bool>
std::vector< std::string > Counters< Array_Type, Data_Type >::get_names ( ) const
```

8.19.3.7 operator=() [1/2]

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

Copy assignment constructor.

Parameters

*counter_*	

Returns

Counters<Array_Type,Data_Type>

8.19.3.8 operator=() [2/2]

```
template<typename Array_Type = BArray<>, typename Data_Type = bool>
Counters<Array_Type,Data_Type>& Counters< Array_Type, Data_Type >::operator= (
            Counters< Array_Type, Data_Type > && counter_ )  [noexcept]
```

Move assignment constructor.

Parameters

*counter_*	

Returns

Counters<Array_Type,Data_Type>&

8.19.3.9 operator[]()

```
template<typename Array_Type = BArray<>, typename Data_Type = bool>
Counter<Array_Type,Data_Type>& Counters< Array_Type, Data_Type >::operator[] (
            size_t idx )
```

Returns a pointer to a particular counter.

Parameters

idx	Id of the counter

Returns

Counter<Array_Type,Data_Type>*

8.19.3.10 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

size_t

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

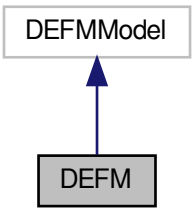
The documentation for this class was generated from the following file:

- include/barry/counters-bones.hpp

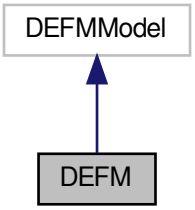
8.20 DEFM Class Reference

`#include <defm-bones.hpp>`

Inheritance diagram for DEFM:



Collaboration diagram for DEFM:



Public Member Functions

- DEFM (int *id, int *y, double *x, size_t id_length, size_t y_ncol, size_t x_ncol, size_t m_order, bool copy_↩ data=true, bool column_major=true)
- DEFMModel & get_model ()
- void init ()
- void simulate (std::vector< double > par, int *y_out)
- size_t get_n_y () const
- size_t get_n_obs () const
- size_t get_n_covars () const
- size_t get_m_order () const
- size_t get_n_rows () const
- const int * get_Y () const
- const int * get_ID () const
- const double * get_X () const
- barry::FreqTable< int > motif_census (std::vector< size_t > idx)

- std::vector< double > logodds (const std::vector< double > &par, size_t i, size_t j)
- void set_names (std::vector< std::string > Y_names_, std::vector< std::string > X_names_)
- const std::vector< std::string > & get_Y_names () const
- const std::vector< std::string > & get_X_names () const
- void print () const
- std::vector< bool > is_motif ()
- bool get_column_major () const noexcept

8.20.1 Detailed Description

Definition at line 4 of file defm-bones.hpp.

8.20.2 Constructor & Destructor Documentation

8.20.2.1 DEFM()

```
DEFM::DEFM (
            int * id,
            int * y,
            double * x,
            size_t id_length,
            size_t y_ncol,
            size_t x_ncol,
            size_t m_order,
            bool copy_data = true,
            bool column_major = true )  [inline]
```

Definition at line 105 of file defm-meat.hpp.

8.20.3 Member Function Documentation

8.20.3.1 get_column_major()

```
bool DEFM::get_column_major ( ) const  [inline], [noexcept]
```

Definition at line 449 of file defm-meat.hpp.

8.20.3.2 get_ID()

```
const int * DEFM::get_ID ( ) const  [inline]
```

Definition at line 310 of file defm-meat.hpp.

8.20.3.3 get_m_order()

```
size_t DEFM::get_m_order ( ) const  [inline]
```

Definition at line 295 of file defm-meat.hpp.

8.20.3.4 get_model()

```
DEFMModel& DEFM::get_model ( )  [inline]
```

Definition at line 49 of file defm-bones.hpp.

8.20.3.5 get_n_covars()

```
size_t DEFM::get_n_covars ( ) const  [inline]
```

Definition at line 290 of file defm-meat.hpp.

8.20.3.6 get_n_obs()

```
size_t DEFM::get_n_obs ( ) const  [inline]
```

Definition at line 285 of file defm-meat.hpp.

8.20.3.7 get_n_rows()

```
size_t DEFM::get_n_rows ( ) const  [inline]
```

Definition at line 300 of file defm-meat.hpp.

8.20.3.8 get_n_y()

```
size_t DEFM::get_n_y ( ) const  [inline]
```

Definition at line 280 of file defm-meat.hpp.

8.20.3.9 get_X()

```
const double * DEFM::get_X ( ) const  [inline]
```

Definition at line 315 of file defm-meat.hpp.

8.20.3.10 get_X_names()

```
const std::vector< std::string > & DEFM::get_X_names ( ) const  [inline]
```

Definition at line 422 of file defm-meat.hpp.

8.20.3.11 get_Y()

```
const int * DEFM::get_Y ( ) const  [inline]
```

Definition at line 305 of file defm-meat.hpp.

8.20.3.12 get_Y_names()

```
const std::vector< std::string > & DEFM::get_Y_names ( ) const  [inline]
```

Definition at line 418 of file defm-meat.hpp.

8.20.3.13 init()

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

Definition at line 215 of file defm-meat.hpp.

8.20.3.14 is_motif()

```
std::vector< bool > DEFM::is_motif ( )  [inline]
```

Definition at line 439 of file defm-meat.hpp.

8.20.3.15 logodds()

```
std::vector< double > DEFM::logodds (
            const std::vector< double > & par,
            size_t i,
            size_t j )  [inline]
```

Definition at line 359 of file defm-meat.hpp.

8.20.3.16 motif_census()

```
barry::FreqTable< int > DEFM::motif_census (
            std::vector< size_t > idx )  [inline]
```

Definition at line 321 of file defm-meat.hpp.

8.20.3.17 print()

```
void DEFM::print ( ) const  [inline]
```

Definition at line 426 of file defm-meat.hpp.

8.20.3.18 set_names()

```
void DEFM::set_names (
            std::vector< std::string > Y_names_,
            std::vector< std::string > X_names_ )  [inline]
```

Definition at line 401 of file defm-meat.hpp.

8.20.3.19 simulate()

```
void DEFM::simulate (
            std::vector< double > par,
            int * y_out )  [inline]
```

Definition at line 39 of file defm-meat.hpp.

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

- include/barry/models/defm/defm-bones.hpp
- include/barry/models/defm/defm-meat.hpp

8.21 DEFMCounterData Class Reference

Data class used to store arbitrary size_t or double vectors.

`#include <defm-types.hpp>`

Public Member Functions

- DEFMCounterData ()
- DEFMCounterData (const std::vector< size_t > indices_, const std::vector< double > numbers_, const std::vector< bool > logical_, bool is_motif_=true)
- size_t idx (size_t i) const
- double num (size_t i) const
- bool is_true (size_t i) const
- ~DEFMCounterData ()

Public Attributes

- std::vector< size_t > indices
- std::vector< double > numbers
- std::vector< bool > logical
- bool is_motif

  If false, then is a logit intercept.

8.21.1 Detailed Description

Data class used to store arbitrary size_t or double vectors.

Definition at line 64 of file defm-types.hpp.

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

- include/barry/models/defm/defm-types.hpp

8.22 DEFMData Class Reference

Data class for DEFM arrays.

`#include <defm-types.hpp>`

Public Member Functions

- DEFMData ()

  Vector indicating which covariates are included in the model.
- DEFMData (DEFMArray *array_, const double *covariates_, size_t obs_start_, size_t X_ncol_, size_t X_↩ nrow_)

  Constructor.
- double operator() (size_t i, size_t j) const

  Access to the row (i) colum (j) data.
- double at (size_t i, size_t j) const
- size_t ncol () const
- size_t nrow () const
- void print () const
- ~DEFMData ()

Public Attributes

- DEFMArray * array
- const double * covariates

  Vector of covariates (complete vector)
- size_t obs_start

  Index of the observation in the data.
- size_t X_ncol

  Number of columns in the array of covariates.
- size_t X_nrow

  Number of rows in the array of covariates.
- std::vector< size_t > covar_sort
- std::vector< size_t > covar_used

  Value where the sorting of the covariates is stored.

8.22.1 Detailed Description

Data class for DEFM arrays.

This holds information pointing to the data array, including information regarding the number of observations, the time slices of the observation, and the number of covariates in the data.

Definition at line 16 of file defm-types.hpp.

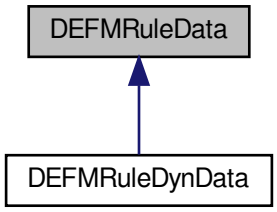
The documentation for this class was generated from the following file:

- include/barry/models/defm/defm-types.hpp

8.23 DEFMRuleData Class Reference

`#include <defm-types.hpp>`

Inheritance diagram for DEFMRuleData:



Public Member Functions

- double num (size_t i) const
- size_t idx (size_t i) const
- bool is_true (size_t i) const
- DEFMRuleData ()
- DEFMRuleData (std::vector< double > numbers_, std::vector< size_t > indices_, std::vector< bool > logical_)
- DEFMRuleData (std::vector< double > numbers_, std::vector< size_t > indices_)

Public Attributes

- std::vector< double > numbers
- std::vector< size_t > indices
- std::vector< bool > logical
- bool init = false

8.23.1 Detailed Description

Definition at line 89 of file defm-types.hpp.

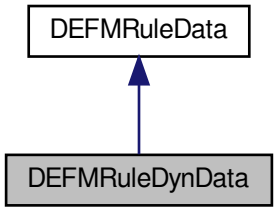
The documentation for this class was generated from the following file:

- include/barry/models/defm/defm-types.hpp

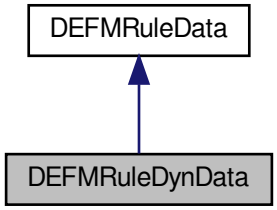
8.24 DEFMRuleDynData Class Reference

`#include <defm-types.hpp>`

Inheritance diagram for DEFMRuleDynData:



Collaboration diagram for DEFMRuleDynData:



Public Member Functions

- DEFMRuleDynData (const std::vector< double > *counts_, std::vector< double > numbers_={}, std::vector< size_t > indices_={}, std::vector< bool > logical_={})
- ~DEFMRuleDynData ()

Public Attributes

- const std::vector< double > * counts

8.24.1 Detailed Description

Definition at line 152 of file defm-types.hpp.

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

- include/barry/models/defm/defm-types.hpp

8.25 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 (size_t n)
- ~Entries ()
- void resize (size_t n)

Public Attributes

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

8.25.1 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 78 of file typedefs.hpp.

8.25.2 Constructor & Destructor Documentation

8.25.2.1 Entries() [1/2]

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

Definition at line 84 of file typedefs.hpp.

8.25.2.2 Entries() [2/2]

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

Definition at line 85 of file typedefs.hpp.

8.25.2.3 ~Entries()

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

Definition at line 92 of file typedefs.hpp.

8.25.3 Member Function Documentation

8.25.3.1 resize()

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

Definition at line 94 of file typedefs.hpp.

8.25.4 Member Data Documentation

8.25.4.1 source

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

Definition at line 80 of file typedefs.hpp.

8.25.4.2 target

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

Definition at line 81 of file typedefs.hpp.

8.25.4.3 val

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

Definition at line 82 of file typedefs.hpp.

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

- include/barry/typedefs.hpp

8.26 Flock Class Reference

A Flock is a group of Geese.

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

Public Member Functions

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

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

  Set the seed of the model.
- void init (size_t bar_width=BARRY_PROGRESS_BAR_WIDTH)
- PhyloCounters * get_counters ()
- PhyloSupport * get_support_fun ()
- std::vector< double > * get_stats_support ()
- std::vector< std::vector< double > > * get_stats_target ()
- PhyloModel * get_model ()
- double likelihood_joint (const std::vector< double > &par, bool as_log=false, bool use_reduced_sequence=true, size_t ncores=1u)

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

  Access the i-th geese element.

Information about the model

- size_t nfuns () const noexcept
- size_t ntrees () const noexcept
- std::vector< size_t > nnodes () const noexcept
- std::vector< size_t > nleafs () const noexcept
- size_t nterms () const
- size_t support_size () const noexcept
- std::vector< std::string > colnames () const
- size_t parse_polytomies (bool verb=true, std::vector< size_t > *dist=nullptr) const noexcept

  Check polytomies and return the largest.
- void print () const

Public Attributes

- std::vector< Geese > dat
- size_t nfunctions = 0u
- bool initialized = false
- std::mt19937 rengine
- PhyloModel model = PhyloModel()

8.26.1 Detailed Description

A Flock is a group of Geese.

This object builds 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.

8.26.2 Constructor & Destructor Documentation

8.26.2.1 Flock()

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

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

8.26.2.2 ~Flock()

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

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

8.26.3 Member Function Documentation

8.26.3.1 add_data()

```
size_t Flock::add_data (
            std::vector< std::vector< size_t > > & annotations,
            std::vector< size_t > & 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

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

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

8.26.3.2 colnames()

```
std::vector< std::string > Flock::colnames ( ) const  [inline]
```

Definition at line 246 of file flock-meat.hpp.

8.26.3.3 get_counters()

```
PhyloCounters * Flock::get_counters ( )  [inline]
```

Definition at line 100 of file flock-meat.hpp.

8.26.3.4 get_model()

```
PhyloModel * Flock::get_model ( )  [inline]
```

Definition at line 131 of file flock-meat.hpp.

8.26.3.5 get_stats_support()

```
std::vector< double > * Flock::get_stats_support ( )  [inline]
```

Definition at line 117 of file flock-meat.hpp.

8.26.3.6 get_stats_target()

```
std::vector< std::vector< double > > * Flock::get_stats_target ( )  [inline]
```

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

8.26.3.7 get_support_fun()

```
PhyloSupport * Flock::get_support_fun ( )  [inline]
```

Definition at line 110 of file flock-meat.hpp.

8.26.3.8 init()

```
void Flock::init (
            size_t bar_width = BARRY_PROGRESS_BAR_WIDTH )  [inline]
```

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

8.26.3.9 likelihood_joint()

```
double Flock::likelihood_joint (
            const std::vector< double > & par,
            bool as_log = false,
            bool use_reduced_sequence = true,
            size_t ncores = 1u )  [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 138 of file flock-meat.hpp.

8.26.3.10 nfuns()

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

Definition at line 189 of file flock-meat.hpp.

8.26.3.11 nleafs()

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

Definition at line 217 of file flock-meat.hpp.

8.26.3.12 nnodes()

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

Definition at line 203 of file flock-meat.hpp.

8.26.3.13 nterms()

```
size_t Flock::nterms ( ) const  [inline]
```

Definition at line 231 of file flock-meat.hpp.

8.26.3.14 ntrees()

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

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

8.26.3.15 operator()()

```
Geese * Flock::operator() (
            size_t 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 324 of file flock-meat.hpp.

8.26.3.16 parse_polytomies()

```
size_t Flock::parse_polytomies (
            bool verb = true,
            std::vector< size_t > * dist = nullptr ) const  [inline], [noexcept]
```

Check polytomies and return the largest.

Definition at line 253 of file flock-meat.hpp.

8.26.3.17 print()

```
void Flock::print ( ) const  [inline]
```

Definition at line 280 of file flock-meat.hpp.

8.26.3.18 set_seed()

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

Set the seed of the model.

Parameters

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

Definition at line 42 of file flock-meat.hpp.

8.26.3.19 support_size()

```
size_t Flock::support_size ( ) const  [inline], [noexcept]
```

Definition at line 239 of file flock-meat.hpp.

8.26.4 Member Data Documentation

8.26.4.1 dat

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

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

8.26.4.2 initialized

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

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

8.26.4.3 model

```
PhyloModel Flock::model = PhyloModel()
```

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

8.26.4.4 nfunctions

```
size_t Flock::nfunctions = 0u
```

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

8.26.4.5 rengine

`std::mt19937 Flock::rengine`

Definition at line 22 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-meat.hpp

8.27 FreqTable< T > Class Template Reference

Frequency table of vectors.

`#include <freqtable.hpp>`

Public Member Functions

- FreqTable ()
- ~FreqTable ()
- size_t add (const std::vector< T > &x, size_t *h_precomp)
- Counts_type as_vector () const
- const std::vector< double > & get_data () const
- const std::unordered_map< size_t, size_t > & get_index () const
- void clear ()
- void reserve (size_t n, size_t k)
- void print () const
- size_t size () const noexcept

  Number of unique elements in the table. (.
- size_t make_hash (const std::vector< T > &x) const

8.27.1 Detailed Description

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

Frequency table of vectors.

This is mostly used in `Support`. The main data is contained in the `data` double vector. The matrix is stored in a row-wise fashion, where the first element is the frequency with which the vector is observed.

For example, in a model with `k` terms the first k + 1 elements of `data` would be:

- weights
- term 1
- term 2
- ...
- term k

Definition at line 22 of file freqtable.hpp.

8.27.2 Constructor & Destructor Documentation

8.27.2.1 FreqTable()

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

Definition at line 34 of file freqtable.hpp.

8.27.2.2 ~FreqTable()

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

Definition at line 35 of file freqtable.hpp.

8.27.3 Member Function Documentation

8.27.3.1 add()

```
template<typename T >
size_t FreqTable< T >::add (
            const std::vector< T > & x,
            size_t * h_precomp )  [inline]
```

Definition at line 59 of file freqtable.hpp.

8.27.3.2 as_vector()

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

Definition at line 139 of file freqtable.hpp.

8.27.3.3 clear()

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

Definition at line 168 of file freqtable.hpp.

8.27.3.4 get_data()

```
template<typename T = double>
const std::vector< double >& FreqTable< T >::get_data ( ) const  [inline]
```

Definition at line 40 of file freqtable.hpp.

8.27.3.5 get_index()

```
template<typename T = double>
const std::unordered_map<size_t,size_t>& FreqTable< T >::get_index ( ) const  [inline]
```

Definition at line 41 of file freqtable.hpp.

8.27.3.6 make_hash()

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

Definition at line 239 of file freqtable.hpp.

8.27.3.7 print()

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

Definition at line 204 of file freqtable.hpp.

8.27.3.8 reserve()

```
template<typename T >
void FreqTable< T >::reserve (
            size_t n,
            size_t k )  [inline]
```

Definition at line 182 of file freqtable.hpp.

8.27.3.9 size()

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

Number of unique elements in the table. (.

Returns

size_t

Definition at line 231 of file freqtable.hpp.

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

- include/barry/freqtable.hpp

8.28 Geese Class Reference

Annotated Phylo Model.

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

Public Member Functions

- ~Geese ()
- void init (size_t bar_width=BARRY_PROGRESS_BAR_WIDTH)
- 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, size_t ncores=1u, bool no_update_pset_probs=false)
- double likelihood_exhaust (const std::vector< double > &par)
- std::vector< double > get_probabilities () const
- void set_seed (const size_t &s)
- std::vector< std::vector< size_t > > simulate (const std::vector< double > &par)
- std::vector< std::vector< double > > observed_counts ()
- void print_observed_counts ()
- void print () const

Prints information about the GEESE.

- void print_nodes () const
- void init_node (Node &n)
- void update_annotations (size_t nodeid, std::vector< size_t > newann)
- std::vector< std::vector< bool > > get_states () const

  Powerset of a gene's possible states.
- std::vector< size_t > get_annotated_nodes () const

  Returns the ids of the nodes with at least one annotation.
- std::vector< size_t > get_annotations () const

  Returns the annotations of the nodes with at least one annotation.

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`*
duplication	*Logical scalar indicating the type of event (true: duplication, false: speciation.)*

The ordering of the entries does not matter. Passing the nodes in post order or not makes no difference to the constructor.

- Geese ()
- Geese (std::vector< std::vector< size_t > > &annotations, std::vector< size_t > &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

Parameters

verb	*When `true` it will print out information about the encountered polytomies.*

- size_t nfuns () const noexcept

  Number of functions analyzed.
- size_t nnodes () const noexcept

  Number of nodes (interior + leaf)
- size_t nleafs () const noexcept

  Number of leaf.
- size_t nterms () const

  Number of terms included.
- size_t support_size () const noexcept

  Number of unique sets of sufficient stats.
- std::vector< size_t > nannotations () const noexcept

  Number of annotations.
- std::vector< std::string > colnames () const

  Names of the terms in the model.

- size_t parse_polytomies (bool verb=true, std::vector< size_t > *dist=nullptr) const noexcept
  Check polytomies and return the largest.

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< size_t > &preorder)
- std::vector< std::vector< double > > predict_exhaust_backend (const std::vector< double > &par, const std::vector< size_t > &preorder)
- std::vector< std::vector< double > > predict_exhaust (const std::vector< double > &par)
- std::vector< std::vector< double > > predict_sim (const std::vector< double > &par, bool only_↩ annotated=false, size_t nsims=10000u)

Non-const pointers to shared objects in <tt>Geese</tt>

These functions provide direct access to some member objects that are shared by the nodes within `Geese`.

Returns

`get_rengine()` returns the Pseudo-RNG engine used.
`get_counters()` returns the vector of counters used.
`get_model()` returns the `Model` object used.
`get_support_fun()` returns the computed support of the model.

- std::mt19937 * get_rengine ()
- PhyloCounters * get_counters ()
- PhyloModel * get_model ()
- PhyloSupport * get_support_fun ()

Public Attributes

- size_t nfunctions
- std::map< size_t, Node > nodes
- barry::MapVec_type< size_t > map_to_state_id

- std::vector< std::vector< std::vector< size_t > > > pset_loc

  Locations of columns.
- std::vector< size_t > sequence
- std::vector< size_t > reduced_sequence
- bool initialized = false
- bool delete_rengine = false
- bool delete_support = false

Static Public Attributes

- static const size_t etype_default = 1ul
- static const size_t etype_speciation = 0ul
- static const size_t etype_duplication = 1ul
- static const size_t etype_either = 2ul

8.28.1 Detailed Description

Annotated Phylo Model.

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

Class representing a phylogenetic tree model with annotations.

The `Geese` class represents a phylogenetic tree model with annotations. It includes a total of `N + 1` nodes, the `+ 1` being the root node. The class provides methods for initializing the model, calculating the likelihood, simulating trees, and making predictions.

The class includes shared objects within a `Geese` object, such as `rengine`, `model`, `states`, `n_zeros`, `n_ones`, `n_dupl_events`, and `n_spec_events`. It also includes information about the type of event, such as `etype_default`, `etype_speciation`, `etype_duplication`, and `etype_either`.

The class provides constructors, a destructor, and methods for initializing the model, inheriting support, calculating the sequence, calculating the reduced sequence, calculating the likelihood, calculating the likelihood exhaustively, getting probabilities, setting the seed, simulating trees, parsing polytomies, getting observed counts, printing observed counts, printing information about the GEESE, and making predictions.

See also

Flock

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

8.28.2 Constructor & Destructor Documentation

8.28.2.1 Geese() [1/4]

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

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

8.28.2.2 Geese() [2/4]

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

Definition at line 20 of file geese-meat-constructors.hpp.

8.28.2.3 Geese() [3/4]

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

Definition at line 230 of file geese-meat-constructors.hpp.

8.28.2.4 Geese() [4/4]

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

Definition at line 309 of file geese-meat-constructors.hpp.

8.28.2.5 ~Geese()

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

Definition at line 119 of file geese-meat.hpp.

8.28.3 Member Function Documentation

8.28.3.1 calc_reduced_sequence()

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

Definition at line 358 of file geese-meat.hpp.

8.28.3.2 calc_sequence()

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

Definition at line 314 of file geese-meat.hpp.

8.28.3.3 colnames()

```
std::vector< std::string > Geese::colnames ( ) const  [inline]
```

Names of the terms in the model.

Definition at line 480 of file geese-meat.hpp.

8.28.3.4 get_annotated_nodes()

```
std::vector< size_t > Geese::get_annotated_nodes ( ) const  [inline]
```

Returns the ids of the nodes with at least one annotation.

Definition at line 769 of file geese-meat.hpp.

8.28.3.5 get_annotations()

```
std::vector< size_t > Geese::get_annotations ( ) const  [inline]
```

Returns the annotations of the nodes with at least one annotation.

Definition at line 792 of file geese-meat.hpp.

8.28.3.6 get_counters()

```
PhyloCounters * Geese::get_counters ( )  [inline]
```

Definition at line 752 of file geese-meat.hpp.

8.28.3.7 get_model()

```
PhyloModel * Geese::get_model ( )  [inline]
```

Definition at line 757 of file geese-meat.hpp.

8.28.3.8 get_probabilities()

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

Definition at line 406 of file geese-meat.hpp.

8.28.3.9 get_rengine()

```
std::mt19937 * Geese::get_rengine ( )  [inline]
```

Definition at line 747 of file geese-meat.hpp.

8.28.3.10 get_states()

```
std::vector< std::vector< bool > > Geese::get_states ( ) const  [inline]
```

Powerset of a gene's possible states.

This list of vectors is used throughout Geese. It lists all possible combinations of functional states for any gene. Thus, for P functions, there will be 2^P possible combinations.

Returns

std::vector< std::vector< bool > > of length 2^P.

Definition at line 765 of file geese-meat.hpp.

8.28.3.11 get_support_fun()

```
PhyloSupport * Geese::get_support_fun ( )  [inline]
```

Definition at line 761 of file geese-meat.hpp.

8.28.3.12 inherit_support()

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

Definition at line 257 of file geese-meat.hpp.

8.28.3.13 init()

```
void Geese::init (
            size_t bar_width = BARRY_PROGRESS_BAR_WIDTH )  [inline]
```

Definition at line 131 of file geese-meat.hpp.

8.28.3.14 init_node()

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

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

8.28.3.15 likelihood()

```
double Geese::likelihood (
            const std::vector< double > & par,
            bool as_log = false,
            bool use_reduced_sequence = true,
            size_t ncores = 1u,
            bool no_update_pset_probs = false )  [inline]
```

Definition at line 111 of file geese-meat-likelihood.hpp.

8.28.3.16 likelihood_exhaust()

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

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

8.28.3.17 nannotations()

```
std::vector< size_t > Geese::nannotations ( ) const  [inline], [noexcept]
```

Number of annotations.

Definition at line 471 of file geese-meat.hpp.

8.28.3.18 nfuns()

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

Number of functions analyzed.

Definition at line 427 of file geese-meat.hpp.

8.28.3.19 nleafs()

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

Number of leaf.

Definition at line 441 of file geese-meat.hpp.

8.28.3.20 nnodes()

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

Number of nodes (interior + leaf)

Definition at line 434 of file geese-meat.hpp.

8.28.3.21 nterms()

```
size_t Geese::nterms ( ) const  [inline]
```

Number of terms included.

Definition at line 453 of file geese-meat.hpp.

8.28.3.22 observed_counts()

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

Definition at line 522 of file geese-meat.hpp.

8.28.3.23 operator=() [1/2]

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

8.28.3.24 operator=() [2/2]

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

8.28.3.25 parse_polytomies()

```
size_t Geese::parse_polytomies (
            bool verb = true,
            std::vector< size_t > * dist = nullptr ) const  [inline], [noexcept]
```

Check polytomies and return the largest.

Definition at line 487 of file geese-meat.hpp.

8.28.3.26 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 287 of file geese-meat-predict.hpp.

8.28.3.27 predict_backend()

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

< True if the array belongs to the set

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

8.28.3.28 predict_exhaust()

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

Definition at line 5 of file geese-meat-predict_exhaust.hpp.

8.28.3.29 predict_exhaust_backend()

```
std::vector< std::vector< double > > Geese::predict_exhaust_backend (
            const std::vector< double > & par,
            const std::vector< size_t > & preorder )  [inline]
```

Definition at line 47 of file geese-meat-predict_exhaust.hpp.

8.28.3.30 predict_sim()

```
std::vector< std::vector< double > > Geese::predict_sim (
            const std::vector< double > & par,
            bool only_annotated = false,
            size_t nsims = 10000u )  [inline]
```

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

8.28.3.31 print()

```
void Geese::print ( ) const  [inline]
```

Prints information about the GEESE.

Definition at line 656 of file geese-meat.hpp.

8.28.3.32 print_nodes()

```
void Geese::print_nodes ( ) const  [inline]
```

Definition at line 674 of file geese-meat.hpp.

8.28.3.33 print_observed_counts()

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

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

8.28.3.34 set_seed()

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

Definition at line 4 of file geese-meat-simulate.hpp.

8.28.3.35 simulate()

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

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

8.28.3.36 support_size()

```
size_t Geese::support_size ( ) const  [inline], [noexcept]
```

Number of unique sets of sufficient stats.

Definition at line 461 of file geese-meat.hpp.

8.28.3.37 update_annotations()

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

Definition at line 285 of file geese-meat.hpp.

8.28.4 Member Data Documentation

8.28.4.1 delete_rengine

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

Definition at line 142 of file geese-bones.hpp.

8.28.4.2 delete_support

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

Definition at line 143 of file geese-bones.hpp.

8.28.4.3 etype_default

```
const size_t Geese::etype_default = 1ul  [static]
```

Definition at line 156 of file geese-bones.hpp.

8.28.4.4 etype_duplication

```
const size_t Geese::etype_duplication = 1ul  [static]
```

Definition at line 158 of file geese-bones.hpp.

8.28.4.5 etype_either

```
const size_t Geese::etype_either = 2ul  [static]
```

Definition at line 159 of file geese-bones.hpp.

8.28.4.6 etype_speciation

```
const size_t Geese::etype_speciation = 0ul  [static]
```

Definition at line 157 of file geese-bones.hpp.

8.28.4.7 initialized

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

Definition at line 141 of file geese-bones.hpp.

8.28.4.8 map_to_state_id

```
barry::MapVec_type< size_t > Geese::map_to_state_id
```

Definition at line 133 of file geese-bones.hpp.

8.28.4.9 nfunctions

```
size_t Geese::nfunctions
```

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

8.28.4.10 nodes

```
std::map< size_t, Node > Geese::nodes
```

Definition at line 131 of file geese-bones.hpp.

8.28.4.11 pset_loc

```
std::vector< std::vector< std::vector< size_t > > > Geese::pset_loc
```

Locations of columns.

Definition at line 134 of file geese-bones.hpp.

8.28.4.12 reduced_sequence

```
std::vector< size_t > Geese::reduced_sequence
```

Definition at line 138 of file geese-bones.hpp.

8.28.4.13 sequence

```
std::vector< size_t > Geese::sequence
```

Definition at line 137 of file geese-bones.hpp.

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

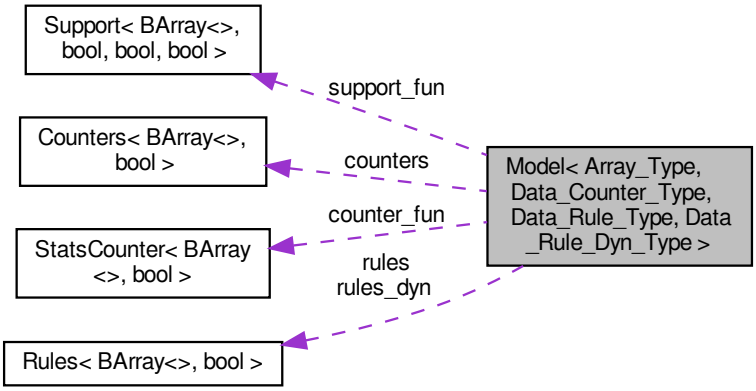
- include/barry/models/geese/geese-bones.hpp
- include/barry/models/geese/geese-meat-constructors.hpp
- include/barry/models/geese/geese-meat-likelihood.hpp
- include/barry/models/geese/geese-meat-likelihood_exhaust.hpp
- include/barry/models/geese/geese-meat-predict.hpp
- include/barry/models/geese/geese-meat-predict_exhaust.hpp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

8.29 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

- void update_normalizing_constants (const std::vector< double > &params,)

  Computes the normalizing constant for a given set of parameters.
- void update_likelihoods (const std::vector< double > &params,)
- void update_pset_probs (const std::vector< double > &params,)
- void set_rengine (std::mt19937 *rengine_, bool delete_=false)
- void set_seed (size_t s)
- Model ()
- Model (size_t 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_)
- virtual ~Model ()
- void store_psets () noexcept
- std::vector< double > gen_key (const Array_Type &Array_)
- size_t add_array (const Array_Type &Array_, bool force_new=false)

  Adds an array to the support of not already included.
- void print_stats (size_t i) const
- virtual void print () const

  Prints information about the model.
- Array_Type sample (const Array_Type &Array_, const std::vector< double > &params={})
- Array_Type sample (const size_t &i, const std::vector< double > &params)

- double conditional_prob (const Array_Type &Array_, const std::vector< double > &params, size_t i, size_t j)

  Conditional probability ("Gibbs sampler")
- const std::mt19937 * get_rengine () const
- Counters< Array_Type, Data_Counter_Type > * get_counters ()
- Rules< Array_Type, Data_Rule_Type > * get_rules ()
- Rules< Array_Type, Data_Rule_Dyn_Type > * get_rules_dyn ()
- Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type > * get_support_fun ()

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_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)
- void set_counters (Counters< Array_Type, Data_Counter_Type > *counters_)
- void add_hasher (Hasher_fun_type< Array_Type, Data_Counter_Type > fun_)

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_fun_type< Array_Type, Data_Rule_Type > count_fun_, Data_Rule_Type data_)
- 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_fun_type< Array_Type, Data_Rule_Dyn_Type > count_fun_, Data_Rule_Dyn↩ _Type data_)
- 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 size_t &i, bool as_log=false, bool no_↩ update_normconst=false)
- double likelihood (const std::vector< double > &params, const Array_Type &Array_, int i=-1, bool as_↩ log=false, bool no_update_normconst=false)
- double likelihood (const std::vector< double > &params, const std::vector< double > &target_, const size_t &i, bool as_log=false, bool no_update_normconst=false)
- double likelihood (const std::vector< double > &params, const double *target_, const size_t &i, bool as↩ _log=false, bool no_update_normconst=false)
- double likelihood_total (const std::vector< double > &params, bool as_log=false, BARRY_NCORES_ARG(=2), bool no_update_normconst=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.*

- const std::vector< double > & get_normalizing_constants () const
- const std::vector< double > & get_likelihoods () const
- const std::vector< Array_Type > * get_pset (const size_t &i)
- const std::vector< double > * get_pset_stats (const size_t &i)

Size of the model

Number of different supports included in the model

This will return the size of `stats_target`.

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.*

- size_t size () const noexcept
- size_t size_unique () const noexcept
- size_t nterms () const noexcept
- size_t nrules () const noexcept
- size_t nrules_dyn () const noexcept
- size_t support_size () const noexcept
- std::vector< std::string > colnames () const

- std::vector< std::vector< double > > * get_stats_target ()

  Raw pointers to the support and target statistics.
- std::vector< double > * get_stats_support ()

  Sufficient statistics of the support(s)
- std::vector< size_t > * get_stats_support_sizes ()

  Number of vectors included in the support.
- std::vector< size_t > * get_stats_support_sizes_acc ()

  Accumulated number of vectors included in the support.
- std::vector< size_t > * get_arrays2support ()
- std::vector< std::vector< Array_Type > > * get_pset_arrays ()
- std::vector< std::vector< double > > * get_pset_stats ()

  Statistics of the support(s)
- std::vector< std::vector< double > > * get_pset_probs ()

- void set_transform_model (std::function< std::vector< double >(double *, size_t)> fun, std::vector< std::string > names)

  Set the transform_model_fun object.
- std::vector< double > transform_model (double *data, size_t k)

Protected Attributes

- MapVec_type< double, size_t > 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
- bool delete_counters = false
- bool delete_rules = false
- bool delete_rules_dyn = false
- std::function< std::vector< double >double *, size_t k)> transform_model_fun = nullptr

  Transformation of the model.
- std::vector< std::string > transform_model_term_names

Random number generation

Random number generation

- std::mt19937 * rengine = nullptr
- bool delete_rengine = false

Information about the arrays used in the model

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

Each vector of `stats_support` *has the data stored in a row-wise order, with each row starting with the weights, e.g., in a model with* `k` *terms the first k + 1 elements of* `stats_support` *would be:*

- *weights*
- *term 1*
- *term 2*
- *...*
- *term k*

- std::vector< double > stats_support

  Sufficient statistics of the model (support)
- std::vector< size_t > stats_support_sizes

  Number of vectors included in the support.
- std::vector< size_t > stats_support_sizes_acc

  Accumulated number of vectors included in the support.
- std::vector< size_t > stats_support_n_arrays

  Number of arrays included per support.
- std::vector< std::vector< double > > stats_target

  Target statistics of the model.
- std::vector< double > stats_likelihood
- std::vector< size_t > arrays2support

Container space for the powerset (and its sufficient stats_target)

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

  Arrays of the support(s)

- std::vector< std::vector< double > > pset_stats
  Statistics of the support(s)
- std::vector< std::vector< double > > pset_probs
  Probabilities of the support(s)

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

8.29.1 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 34 of file model-bones.hpp.

8.29.2 Constructor & Destructor Documentation

8.29.2.1 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 269 of file model-meat.hpp.

8.29.2.2 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 (
            size_t size_ )  [inline]
```

Definition at line 307 of file model-meat.hpp.

8.29.2.3 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 349 of file model-meat.hpp.

8.29.2.4 ~Model()

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

Definition at line 183 of file model-bones.hpp.

8.29.3 Member Function Documentation

8.29.3.1 add_array()

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
size_t 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 587 of file model-meat.hpp.

8.29.3.2 add_counter() [1/2]

```
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 462 of file model-meat.hpp.

8.29.3.3 add_counter() [2/2]

```
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 )  [inline]
```

Definition at line 471 of file model-meat.hpp.

8.29.3.4 add_hasher()

```
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_hasher (
            Hasher_fun_type< Array_Type, Data_Counter_Type > fun_ )  [inline]
```

Definition at line 506 of file model-meat.hpp.

8.29.3.5 add_rule() [1/2]

```
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 517 of file model-meat.hpp.

8.29.3.6 add_rule() [2/2]

```
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 >::add_rule (
            Rule_fun_type< Array_Type, Data_Rule_Type > count_fun_,
            Data_Rule_Type data_ )
```

8.29.3.7 add_rule_dyn() [1/2]

```
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 545 of file model-meat.hpp.

8.29.3.8 add_rule_dyn() [2/2]

```
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_ )  [inline]
```

Definition at line 554 of file model-meat.hpp.

8.29.3.9 colnames()

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
std::vector< std::string > Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_↩
Dyn_Type >::colnames  [inline]
```

Definition at line 1310 of file model-meat.hpp.

8.29.3.10 conditional_prob()

```
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 >::conditional↩
_prob (
            const Array_Type & Array_,
            const std::vector< double > & params,
            size_t i,
            size_t j )  [inline]
```

Conditional probability ("Gibbs sampler")

Computes the conditional probability of observing P{Y(i,j) = | Y^C, theta}, i.e., the probability of observing the entry Y(i,j) equal to one given the rest of the array.

Parameters

*Array_*	Array to check
params	Vector of parameters
i	Row entry
j	Column entry

Returns

double The conditional probability

Definition at line 1602 of file model-meat.hpp.

8.29.3.11 gen_key()

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
std::vector< double > Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_↩
Type >::gen_key (
            const Array_Type & Array_ )  [inline]
```

Definition at line 455 of file model-meat.hpp.

8.29.3.12 get_arrays2support()

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
std::vector< size_t > * Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn↩
_Type >::get_arrays2support  [inline]
```

Definition at line 1691 of file model-meat.hpp.

8.29.3.13 get_counters()

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
Counters< Array_Type, Data_Counter_Type > * Model< Array_Type, Data_Counter_Type, Data_Rule↩
_Type, Data_Rule_Dyn_Type >::get_counters  [inline]
```

Definition at line 1640 of file model-meat.hpp.

8.29.3.14 get_likelihoods()

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
const std::vector< double > & Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_↩
Rule_Dyn_Type >::get_likelihoods  [inline]
```

Definition at line 1117 of file model-meat.hpp.

8.29.3.15 get_normalizing_constants()

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
const std::vector< double > & Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_↩
Rule_Dyn_Type >::get_normalizing_constants  [inline]
```

Definition at line 1104 of file model-meat.hpp.

8.29.3.16 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 size_t & i )  [inline]
```

Definition at line 1125 of file model-meat.hpp.

8.29.3.17 get_pset_arrays()

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
std::vector< std::vector< Array_Type > > * Model< Array_Type, Data_Counter_Type, Data_Rule_↩
Type, Data_Rule_Dyn_Type >::get_pset_arrays  [inline]
```

Definition at line 1698 of file model-meat.hpp.

8.29.3.18 get_pset_probs()

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
std::vector< std::vector< double > > * Model< Array_Type, Data_Counter_Type, Data_Rule_Type,
Data_Rule_Dyn_Type >::get_pset_probs  [inline]
```

Definition at line 1710 of file model-meat.hpp.

8.29.3.19 get_pset_stats() [1/2]

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
std::vector< std::vector< double > > * Model< Array_Type, Data_Counter_Type, Data_Rule_Type,
Data_Rule_Dyn_Type >::get_pset_stats  [inline]
```

Statistics of the support(s)

Definition at line 1704 of file model-meat.hpp.

8.29.3.20 get_pset_stats() [2/2]

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

Definition at line 1139 of file model-meat.hpp.

8.29.3.21 get_rengine()

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
const std::mt19937 * Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type
>::get_rengine  [inline]
```

Definition at line 1635 of file model-meat.hpp.

8.29.3.22 get_rules()

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

Definition at line 1645 of file model-meat.hpp.

8.29.3.23 get_rules_dyn()

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
Rules< Array_Type, Data_Rule_Dyn_Type > * Model< Array_Type, Data_Counter_Type, Data_Rule_↩
Type, Data_Rule_Dyn_Type >::get_rules_dyn  [inline]
```

Definition at line 1650 of file model-meat.hpp.

8.29.3.24 get_stats_support()

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
std::vector< double > * Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn↩
_Type >::get_stats_support  [inline]
```

Sufficient statistics of the support(s)

Definition at line 1668 of file model-meat.hpp.

8.29.3.25 get_stats_support_sizes()

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
std::vector< size_t > * Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn↩
_Type >::get_stats_support_sizes  [inline]
```

Number of vectors included in the support.

Definition at line 1676 of file model-meat.hpp.

8.29.3.26 get_stats_support_sizes_acc()

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
std::vector< size_t > * Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn↩
_Type >::get_stats_support_sizes_acc  [inline]
```

Accumulated number of vectors included in the support.

Definition at line 1684 of file model-meat.hpp.

8.29.3.27 get_stats_target()

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
std::vector< std::vector< double > > * Model< Array_Type, Data_Counter_Type, Data_Rule_Type,
Data_Rule_Dyn_Type >::get_stats_target  [inline]
```

Raw pointers to the support and target statistics.

The support of the model is stored as a vector of vector<double>. Each element of it contains the support for an specific type of array included. It represents an array of size `(k + 1) x n unique elements`, with the data stored by-row. The last element of each entry corresponds to the weights, i.e., the frequency with which such sufficient statistics are observed in the support.

Definition at line 1661 of file model-meat.hpp.

8.29.3.28 get_support_fun()

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

Definition at line 1656 of file model-meat.hpp.

8.29.3.29 likelihood() [1/4]

```
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,
            bool no_update_normconst = false )  [inline]
```

Definition at line 796 of file model-meat.hpp.

8.29.3.30 likelihood() [2/4]

```
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 double * target_,
            const size_t & i,
            bool as_log = false,
            bool no_update_normconst = false )  [inline]
```

Definition at line 946 of file model-meat.hpp.

8.29.3.31 likelihood() [3/4]

```
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 size_t & i,
            bool as_log = false,
            bool no_update_normconst = false )  [inline]
```

Definition at line 749 of file model-meat.hpp.

8.29.3.32 likelihood() [4/4]

```
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 size_t & i,
            bool as_log = false,
            bool no_update_normconst = false )  [inline]
```

Definition at line 880 of file model-meat.hpp.

8.29.3.33 likelihood_total()

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
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,
            BARRY_NCORES_ARG(=2) ,
            bool no_update_normconst = false )  [inline]
```

Definition at line 1018 of file model-meat.hpp.

8.29.3.34 nrules()

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

Definition at line 1280 of file model-meat.hpp.

8.29.3.35 nrules_dyn()

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

Definition at line 1288 of file model-meat.hpp.

8.29.3.36 nterms()

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

Definition at line 1269 of file model-meat.hpp.

8.29.3.37 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 396 of file model-meat.hpp.

8.29.3.38 print()

```
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  [inline],
[virtual]
```

Prints information about the model.

Definition at line 1191 of file model-meat.hpp.

8.29.3.39 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 (
            size_t i ) const  [inline]
```

Definition at line 1151 of file model-meat.hpp.

8.29.3.40 sample() [1/2]

```
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 Array_Type & Array_,
            const std::vector< double > & params = {} )  [inline]
```

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

Definition at line 1407 of file model-meat.hpp.

8.29.3.41 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 size_t & i,
            const std::vector< double > & params )  [inline]
```

Definition at line 1327 of file model-meat.hpp.

8.29.3.42 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 488 of file model-meat.hpp.

8.29.3.43 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 153 of file model-bones.hpp.

8.29.3.44 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 527 of file model-meat.hpp.

8.29.3.45 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 569 of file model-meat.hpp.

8.29.3.46 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 (
            size_t s )  [inline]
```

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

8.29.3.47 set_transform_model()

```
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_transform↩
_model (
            std::function< std::vector< double >(double *, size_t)> fun,
            std::vector< std::string > names )  [inline]
```

Set the transform_model_fun object.

The transform_model function is used to transform the data

Parameters

data	
target	
n_arrays	
arrays2support	

Definition at line 1716 of file model-meat.hpp.

8.29.3.48 size()

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

Definition at line 1252 of file model-meat.hpp.

8.29.3.49 size_unique()

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

Definition at line 1260 of file model-meat.hpp.

8.29.3.50 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 449 of file model-meat.hpp.

8.29.3.51 support_size()

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

Definition at line 1296 of file model-meat.hpp.

8.29.3.52 transform_model()

```
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
>::transform_model (
            double * data,
            size_t k )
```

8.29.3.53 update_likelihoods()

```
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 >::update_↩
likelihoods (
            const std::vector< double > & params )  [inline]
```

Definition at line 176 of file model-meat.hpp.

8.29.3.54 update_normalizing_constants()

```
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 >::update_↩
normalizing_constants (
            const std::vector< double > & params )  [inline]
```

Computes the normalizing constant for a given set of parameters.

This function will compute the normalizing constant for a given set of parameters. It will also update the `normalizing_constants` member variable.

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

8.29.3.55 update_pset_probs()

```
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 >::update_pset↩
_probs (
            const std::vector< double > & params )  [inline]
```

Definition at line 215 of file model-meat.hpp.

8.29.4 Member Data Documentation

8.29.4.1 arrays2support

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
std::vector< size_t > Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_↩
Type >::arrays2support  [protected]
```

Definition at line 68 of file model-bones.hpp.

8.29.4.2 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  [protected]
```

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

8.29.4.3 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  [protected]
```

Definition at line 94 of file model-bones.hpp.

8.29.4.4 delete_counters

```
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_↩
counters = false  [protected]
```

Definition at line 106 of file model-bones.hpp.

8.29.4.5 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  [protected]
```

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

8.29.4.6 delete_rules

```
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_rules
= false  [protected]
```

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

8.29.4.7 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 Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::delete_↩
rules_dyn = false  [protected]
```

Definition at line 108 of file model-bones.hpp.

8.29.4.8 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  [protected]
```

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

8.29.4.9 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, size_t > Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_↩
Rule_Dyn_Type >::keys2support  [protected]
```

Map of types of arrays to support sets.

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

Definition at line 75 of file model-bones.hpp.

8.29.4.10 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  [protected]
```

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

8.29.4.11 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  [protected]
```

Vector of the previously used parameters.

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

8.29.4.12 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  [protected]
```

Arrays of the support(s)

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

8.29.4.13 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  [protected]
```

Probabilities of the support(s)

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

8.29.4.14 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<double> > Model< Array_Type, Data_Counter_Type, Data_Rule_Type,
Data_Rule_Dyn_Type >::pset_stats  [protected]
```

Statistics of the support(s)

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

8.29.4.15 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  [protected]
```

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

8.29.4.16 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  [protected]
```

Definition at line 95 of file model-bones.hpp.

8.29.4.17 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  [protected]
```

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

8.29.4.18 stats_likelihood

```
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 >::stats_likelihood  [protected]
```

Definition at line 67 of file model-bones.hpp.

8.29.4.19 stats_support

```
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 >::stats_support  [protected]
```

Sufficient statistics of the model (support)

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

8.29.4.20 stats_support_n_arrays

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
std::vector< size_t > Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_↩
Type >::stats_support_n_arrays  [protected]
```

Number of arrays included per support.

Definition at line 65 of file model-bones.hpp.

8.29.4.21 stats_support_sizes

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
std::vector< size_t > Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_↩
Type >::stats_support_sizes  [protected]
```

Number of vectors included in the support.

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

8.29.4.22 stats_support_sizes_acc

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
std::vector< size_t > Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_↩
Type >::stats_support_sizes_acc  [protected]
```

Accumulated number of vectors included in the support.

Definition at line 64 of file model-bones.hpp.

8.29.4.23 stats_target

```
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 >::stats_target  [protected]
```

Target statistics of the model.

Definition at line 66 of file model-bones.hpp.

8.29.4.24 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  [protected]
```

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

8.29.4.25 transform_model_fun

```
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>double *, size_t k)> Model< Array_Type, Data_Counter_Type,
Data_Rule_Type, Data_Rule_Dyn_Type >::transform_model_fun = nullptr  [protected]
```

Transformation of the model.

When specified, this function will update the model by modifying the linear equation. For example, if the user wanted to add interaction terms, rescale, or apply other operations of the sorts, the user can do such through this function.

The function should return `void` and receive the following arguments:

- `data` Pointer to the first element of the set of sufficient statistics
- `k` size_t indicating the number of sufficient statistics

Returns

Nothing, but it will modify the model data.

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

8.29.4.26 transform_model_term_names

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
std::vector< std::string > Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_↩
Dyn_Type >::transform_model_term_names  [protected]
```

Definition at line 128 of file model-bones.hpp.

8.29.4.27 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  [protected]
```

Definition at line 83 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

8.30 NetCounterData Class Reference

Data class used to store arbitrary size_t or double vectors.

`#include <network.hpp>`

Public Member Functions

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

Public Attributes

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

8.30.1 Detailed Description

Data class used to store arbitrary size_t or double vectors.

Definition at line 56 of file network.hpp.

8.30.2 Constructor & Destructor Documentation

8.30.2.1 NetCounterData() [1/2]

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

Definition at line 62 of file network.hpp.

8.30.2.2 NetCounterData() [2/2]

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

Definition at line 63 of file network.hpp.

8.30.2.3 ~NetCounterData()

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

Definition at line 68 of file network.hpp.

8.30.3 Member Data Documentation

8.30.3.1 indices

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

Definition at line 59 of file network.hpp.

8.30.3.2 numbers

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

Definition at line 60 of file network.hpp.

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

- include/barry/counters/network.hpp

8.31 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

8.31.1 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 19 of file network.hpp.

8.31.2 Constructor & Destructor Documentation

8.31.2.1 NetworkData() [1/3]

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

Definition at line 25 of file network.hpp.

8.31.2.2 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 33 of file network.hpp.

8.31.2.3 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 45 of file network.hpp.

8.31.2.4 ~NetworkData()

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

Definition at line 51 of file network.hpp.

8.31.3 Member Data Documentation

8.31.3.1 directed

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

Definition at line 22 of file network.hpp.

8.31.3.2 vertex_attr

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

Definition at line 23 of file network.hpp.

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

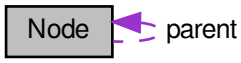
- include/barry/counters/network.hpp

8.32 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
- size_t noffspring () const noexcept
- bool is_leaf () const noexcept

Construct a new Node object

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

Public Attributes

- size_t id

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

  Order in which the node was created.
- PhyloArray array

  Array of the node.
- std::vector< size_t > annotations

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

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

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

  Offspring nodes.
- std::vector< size_t > 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.

8.32.1 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.

8.32.2 Constructor & Destructor Documentation

8.32.2.1 Node() [1/5]

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

Definition at line 37 of file geese-node-bones.hpp.

8.32.2.2 Node() [2/5]

```
Node::Node (
            size_t id_,
            size_t ord_,
            bool duplication_ )  [inline]
```

Definition at line 57 of file geese-node-bones.hpp.

8.32.2.3 Node() [3/5]

```
Node::Node (
            size_t id_,
            size_t ord_,
            std::vector< size_t > annotations_,
            bool duplication_ )  [inline]
```

Definition at line 63 of file geese-node-bones.hpp.

8.32.2.4 Node() [4/5]

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

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

8.32.2.5 Node() [5/5]

```
Node::Node (
            const Node & x )  [inline]
```

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

8.32.2.6 ~Node()

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

Definition at line 48 of file geese-node-bones.hpp.

8.32.3 Member Function Documentation

8.32.3.1 get_parent()

```
int Node::get_parent ( ) const  [inline]
```

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

8.32.3.2 is_leaf()

```
bool Node::is_leaf ( ) const  [inline], [noexcept]
```

Definition at line 112 of file geese-node-bones.hpp.

8.32.3.3 noffspring()

```
size_t Node::noffspring ( ) const  [inline], [noexcept]
```

Definition at line 106 of file geese-node-bones.hpp.

8.32.4 Member Data Documentation

8.32.4.1 annotations

```
std::vector< size_t > Node::annotations
```

Observed annotations (only defined for Geese)

Definition at line 18 of file geese-node-bones.hpp.

8.32.4.2 array

```
PhyloArray Node::array
```

Array of the node.

Definition at line 17 of file geese-node-bones.hpp.

8.32.4.3 arrays

```
std::vector< PhyloArray > Node::arrays = {}
```

Arrays given all possible states.

Definition at line 21 of file geese-node-bones.hpp.

8.32.4.4 duplication

```
bool Node::duplication
```

Definition at line 19 of file geese-node-bones.hpp.

8.32.4.5 id

```
size_t Node::id
```

Id of the node (as specified in the input)

Definition at line 14 of file geese-node-bones.hpp.

8.32.4.6 narray

```
std::vector< size_t > Node::narray = {}
```

ID of the array in the model.

Definition at line 25 of file geese-node-bones.hpp.

8.32.4.7 offspring

```
std::vector< Node* > Node::offspring = {}
```

Offspring nodes.

Definition at line 24 of file geese-node-bones.hpp.

8.32.4.8 ord

```
size_t Node::ord
```

Order in which the node was created.

Definition at line 15 of file geese-node-bones.hpp.

8.32.4.9 parent

```
Node* Node::parent = nullptr
```

Parent node.

Definition at line 23 of file geese-node-bones.hpp.

8.32.4.10 probability

```
std::vector< double > Node::probability
```

The probability of observing each state.

Definition at line 29 of file geese-node-bones.hpp.

8.32.4.11 subtree_prob

```
std::vector< double > Node::subtree_prob
```

Induced subtree probabilities.

Definition at line 28 of file geese-node-bones.hpp.

8.32.4.12 visited

```
bool Node::visited = false
```

Definition at line 26 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

8.33 NodeData Class Reference

Data definition for the `PhyloArray` class.

`#include <geese-types.hpp>`

Public Member Functions

- NodeData (const std::vector< double > &blengths_, const std::vector< bool > &states_, bool duplication_=true)

Public Attributes

- std::vector< double > blengths = {}
- std::vector< bool > states = {}
- bool duplication = true

8.33.1 Detailed Description

Data definition for the `PhyloArray` class.

This holds basic information about a given node.

Definition at line 15 of file geese-types.hpp.

8.33.2 Constructor & Destructor Documentation

8.33.2.1 NodeData()

```
NodeData::NodeData (
            const std::vector< double > & blengths_,
            const std::vector< bool > & states_,
            bool duplication_ = true )  [inline]
```

Definition at line 35 of file geese-types.hpp.

8.33.3 Member Data Documentation

8.33.3.1 blengths

```
std::vector< double > NodeData::blengths = {}
```

Branch length.

Definition at line 21 of file geese-types.hpp.

8.33.3.2 duplication

```
bool NodeData::duplication = true
```

Definition at line 31 of file geese-types.hpp.

8.33.3.3 states

```
std::vector< bool > NodeData::states = {}
```

State of the parent node.

Definition at line 26 of file geese-types.hpp.

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

- include/barry/models/geese/geese-types.hpp

8.34 PhyloCounterData Class Reference

```
#include <geese-types.hpp>
```

Public Member Functions

- PhyloCounterData (std::vector< size_t > data_, std::vector< double > *counters_=nullptr)
- PhyloCounterData ()
- size_t at (size_t d)
- size_t operator() (size_t d)
- size_t operator[ ] (size_t d)
- void reserve (size_t x)
- void push_back (size_t x)
- void shrink_to_fit ()
- size_t size ()
- std::vector< size_t >::iterator begin ()
- std::vector< size_t >::iterator end ()
- bool empty ()
- std::vector< double > * get_counters ()

8.34.1 Detailed Description

Definition at line 45 of file geese-types.hpp.

8.34.2 Constructor & Destructor Documentation

8.34.2.1 PhyloCounterData() [1/2]

```
PhyloCounterData::PhyloCounterData (
            std::vector< size_t > data_,
            std::vector< double > * counters_ = nullptr )  [inline]
```

Definition at line 51 of file geese-types.hpp.

8.34.2.2 PhyloCounterData() [2/2]

```
PhyloCounterData::PhyloCounterData ( )  [inline]
```

Definition at line 56 of file geese-types.hpp.

8.34.3 Member Function Documentation

8.34.3.1 at()

```
size_t PhyloCounterData::at (
            size_t d )  [inline]
```

Definition at line 58 of file geese-types.hpp.

8.34.3.2 begin()

```
std::vector< size_t >::iterator PhyloCounterData::begin ( )  [inline]
```

Definition at line 66 of file geese-types.hpp.

8.34.3.3 empty()

```
bool PhyloCounterData::empty ( )  [inline]
```

Definition at line 69 of file geese-types.hpp.

8.34.3.4 end()

```
std::vector< size_t >::iterator PhyloCounterData::end ( )  [inline]
```

Definition at line 67 of file geese-types.hpp.

8.34.3.5 get_counters()

```
std::vector< double >* PhyloCounterData::get_counters ( )  [inline]
```

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

8.34.3.6 operator()()

```
size_t PhyloCounterData::operator() (
            size_t d )  [inline]
```

Definition at line 59 of file geese-types.hpp.

8.34.3.7 operator[]()

```
size_t PhyloCounterData::operator[] (
            size_t d )  [inline]
```

Definition at line 60 of file geese-types.hpp.

8.34.3.8 push_back()

```
void PhyloCounterData::push_back (
            size_t x )  [inline]
```

Definition at line 62 of file geese-types.hpp.

8.34.3.9 reserve()

```
void PhyloCounterData::reserve (
            size_t x )  [inline]
```

Definition at line 61 of file geese-types.hpp.

8.34.3.10 shrink_to_fit()

```
void PhyloCounterData::shrink_to_fit ( )  [inline]
```

Definition at line 63 of file geese-types.hpp.

8.34.3.11 size()

```
size_t PhyloCounterData::size ( )  [inline]
```

Definition at line 64 of file geese-types.hpp.

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

- include/barry/models/geese/geese-types.hpp

8.35 PhyloRuleDynData Class Reference

`#include <geese-types.hpp>`

Public Member Functions

- PhyloRuleDynData (const std::vector< double > *counts_, size_t pos_, size_t lb_, size_t ub_, size_t duplication_)
- const double operator() () const
- ~PhyloRuleDynData ()

Public Attributes

- const std::vector< double > * counts
- size_t pos
- size_t lb
- size_t ub
- size_t duplication

8.35.1 Detailed Description

Definition at line 74 of file geese-types.hpp.

8.35.2 Constructor & Destructor Documentation

8.35.2.1 PhyloRuleDynData()

```
PhyloRuleDynData::PhyloRuleDynData (
            const std::vector< double > * counts_,
            size_t pos_,
            size_t lb_,
            size_t ub_,
            size_t duplication_ )  [inline]
```

Definition at line 82 of file geese-types.hpp.

8.35.2.2 ~PhyloRuleDynData()

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

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

8.35.3 Member Function Documentation

8.35.3.1 operator()()

```
const double PhyloRuleDynData::operator() ( ) const  [inline]
```

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

8.35.4 Member Data Documentation

8.35.4.1 counts

```
const std::vector< double >* PhyloRuleDynData::counts
```

Definition at line 76 of file geese-types.hpp.

8.35.4.2 duplication

```
size_t PhyloRuleDynData::duplication
```

Definition at line 80 of file geese-types.hpp.

8.35.4.3 lb

```
size_t PhyloRuleDynData::lb
```

Definition at line 78 of file geese-types.hpp.

8.35.4.4 pos

```
size_t PhyloRuleDynData::pos
```

Definition at line 77 of file geese-types.hpp.

8.35.4.5 ub

```
size_t PhyloRuleDynData::ub
```

Definition at line 79 of file geese-types.hpp.

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

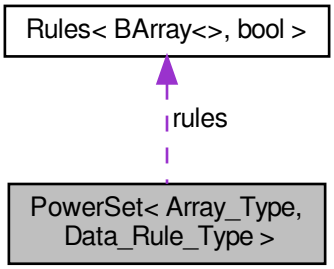
- include/barry/models/geese/geese-types.hpp

8.36 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 (size_t N_, size_t M_)

Construct and destroy a PowerSet object

- PowerSet ()
- PowerSet (size_t N_, size_t 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_fun_type< Array_Type, Data_Rule_Type > count_fun_, Data_Rule_Type data_)

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 size_t &i) const

Public Attributes

- Array_Type EmptyArray
- std::vector< Array_Type > data
- Rules< Array_Type, Data_Rule_Type > * rules
- size_t N
- size_t M
- bool rules_deleted = false
- std::vector< size_t > coordinates_free
- std::vector< size_t > coordinates_locked
- size_t n_free
- size_t n_locked

8.36.1 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 11 of file powerset-bones.hpp.

8.36.2 Constructor & Destructor Documentation

8.36.2.1 PowerSet() [1/3]

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

Definition at line 36 of file powerset-bones.hpp.

8.36.2.2 PowerSet() [2/3]

```
template<typename Array_Type = BArray<>, typename Data_Rule_Type = bool>
PowerSet< Array_Type, Data_Rule_Type >::PowerSet (
            size_t N_,
            size_t M_ )  [inline]
```

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

8.36.2.3 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 5 of file powerset-meat.hpp.

8.36.2.4 ~PowerSet()

```
template<typename Array_Type , typename Data_Rule_Type >
PowerSet< Array_Type, Data_Rule_Type >::~PowerSet  [inline]
```

Definition at line 13 of file powerset-meat.hpp.

8.36.3 Member Function Documentation

8.36.3.1 add_rule() [1/2]

```
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 180 of file powerset-meat.hpp.

8.36.3.2 add_rule() [2/2]

```
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_ )  [inline]
```

Definition at line 189 of file powerset-meat.hpp.

8.36.3.3 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 68 of file powerset-bones.hpp.

8.36.3.4 calc()

```
template<typename Array_Type , typename Data_Rule_Type >
void PowerSet< Array_Type, Data_Rule_Type >::calc  [inline]
```

Definition at line 151 of file powerset-meat.hpp.

8.36.3.5 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 69 of file powerset-bones.hpp.

8.36.3.6 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 67 of file powerset-bones.hpp.

8.36.3.7 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 66 of file powerset-bones.hpp.

8.36.3.8 init_support()

```
template<typename Array_Type , typename Data_Rule_Type >
void PowerSet< Array_Type, Data_Rule_Type >::init_support  [inline]
```

Definition at line 19 of file powerset-meat.hpp.

8.36.3.9 operator[]()

```
template<typename Array_Type = BArray<>, typename Data_Rule_Type = bool>
const Array_Type& PowerSet< Array_Type, Data_Rule_Type >::operator[] (
            const size_t & i ) const  [inline]
```

Definition at line 71 of file powerset-bones.hpp.

8.36.3.10 reset()

```
template<typename Array_Type , typename Data_Rule_Type >
void PowerSet< Array_Type, Data_Rule_Type >::reset (
            size_t N_,
            size_t M_ )  [inline]
```

Definition at line 167 of file powerset-meat.hpp.

8.36.3.11 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 70 of file powerset-bones.hpp.

8.36.4 Member Data Documentation

8.36.4.1 coordinates_free

```
template<typename Array_Type = BArray<>, typename Data_Rule_Type = bool>
std::vector< size_t > PowerSet< Array_Type, Data_Rule_Type >::coordinates_free
```

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

8.36.4.2 coordinates_locked

```
template<typename Array_Type = BArray<>, typename Data_Rule_Type = bool>
std::vector< size_t > PowerSet< Array_Type, Data_Rule_Type >::coordinates_locked
```

Definition at line 27 of file powerset-bones.hpp.

8.36.4.3 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 19 of file powerset-bones.hpp.

8.36.4.4 EmptyArray

```
template<typename Array_Type = BArray<>, typename Data_Rule_Type = bool>
Array_Type PowerSet< Array_Type, Data_Rule_Type >::EmptyArray
```

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

8.36.4.5 M

```
template<typename Array_Type = BArray<>, typename Data_Rule_Type = bool>
size_t PowerSet< Array_Type, Data_Rule_Type >::M
```

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

8.36.4.6 N

```
template<typename Array_Type = BArray<>, typename Data_Rule_Type = bool>
size_t PowerSet< Array_Type, Data_Rule_Type >::N
```

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

8.36.4.7 n_free

```
template<typename Array_Type = BArray<>, typename Data_Rule_Type = bool>
size_t PowerSet< Array_Type, Data_Rule_Type >::n_free
```

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

8.36.4.8 n_locked

```
template<typename Array_Type = BArray<>, typename Data_Rule_Type = bool>
size_t PowerSet< Array_Type, Data_Rule_Type >::n_locked
```

Definition at line 29 of file powerset-bones.hpp.

8.36.4.9 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 20 of file powerset-bones.hpp.

8.36.4.10 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 23 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

8.37 Progress Class Reference

A simple progress bar.

```
#include <progress.hpp>
```

Public Member Functions

- Progress (int n_, int width_)
- ~Progress ()
- void next ()
- void end ()

8.37.1 Detailed Description

A simple progress bar.

Definition at line 11 of file progress.hpp.

8.37.2 Constructor & Destructor Documentation

8.37.2.1 Progress()

```
Progress::Progress (
            int n_,
            int width_ )  [inline]
```

Definition at line 30 of file progress.hpp.

8.37.2.2 ~Progress()

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

Definition at line 23 of file progress.hpp.

8.37.3 Member Function Documentation

8.37.3.1 end()

```
void Progress::end ( )  [inline]
```

Definition at line 52 of file progress.hpp.

8.37.3.2 next()

```
void Progress::next ( )  [inline]
```

Definition at line 41 of file progress.hpp.

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

- include/barry/progress.hpp

8.38 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, size_t i, size_t j)
- std::string & get_name ()
- std::string & get_description ()
- std::string get_name () const
- std::string get_description () const

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_, std::string name_="", std::string desc_="")

8.38.1 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 20 of file rules-bones.hpp.

8.38.2 Constructor & Destructor Documentation

8.38.2.1 Rule() [1/2]

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

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

8.38.2.2 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_,
            std::string name_ = "",
            std::string desc_ = "" )  [inline]
```

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

8.38.2.3 ~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.

8.38.3 Member Function Documentation

8.38.3.1 D()

```
template<typename Array_Type , typename Data_Type >
Data_Type & Rule< Array_Type, Data_Type >::D  [inline]
```

Read/Write access to the data.

Definition at line 37 of file rules-meat.hpp.

8.38.3.2 get_description() [1/2]

```
template<typename Array_Type , typename Data_Type >
std::string & Rule< Array_Type, Data_Type >::get_description  [inline]
```

Definition at line 54 of file rules-meat.hpp.

8.38.3.3 get_description() [2/2]

```
template<typename Array_Type , typename Data_Type >
std::string Rule< Array_Type, Data_Type >::get_description  [inline]
```

Definition at line 66 of file rules-meat.hpp.

8.38.3.4 get_name() [1/2]

```
template<typename Array_Type , typename Data_Type >
std::string & Rule< Array_Type, Data_Type >::get_name  [inline]
```

Definition at line 48 of file rules-meat.hpp.

8.38.3.5 get_name() [2/2]

```
template<typename Array_Type , typename Data_Type >
std::string Rule< Array_Type, Data_Type >::get_name  [inline]
```

Definition at line 60 of file rules-meat.hpp.

8.38.3.6 operator()()

```
template<typename Array_Type , typename Data_Type >
bool Rule< Array_Type, Data_Type >::operator() (
            const Array_Type & a,
            size_t i,
            size_t j )  [inline]
```

Definition at line 43 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

8.39 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 ()
- size_t size () const noexcept
- bool operator() (const Array_Type &a, size_t i, size_t j)

  Check whether a given cell is free or locked.
- void get_seq (const Array_Type &a, std::vector< size_t > *free, std::vector< size_t > *locked=nullptr)

  Computes the sequence of free and locked cells in an BArray.
- std::vector< std::string > get_names () const
- std::vector< std::string > get_descriptions () const
- std::vector< Rule< Array_Type, Data_Type > >::iterator begin ()
- std::vector< Rule< Array_Type, Data_Type > >::iterator end ()

Rule adding

Parameters

rule	

- void add_rule (Rule< Array_Type, Data_Type > rule)
- void add_rule (Rule_fun_type< Array_Type, Data_Type > rule_, Data_Type data_, std::string name_="", std::string description_="")

8.39.1 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 71 of file rules-bones.hpp.

8.39.2 Constructor & Destructor Documentation

8.39.2.1 Rules() [1/2]

```
template<typename Array_Type , typename Data_Type >
Rules< Array_Type, Data_Type >::Rules ( )  [inline]
```

Definition at line 77 of file rules-bones.hpp.

8.39.2.2 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 5 of file rules-meat.hpp.

8.39.2.3 ~Rules()

```
template<typename Array_Type , typename Data_Type >
Rules< Array_Type, Data_Type >::~Rules ( )  [inline]
```

Definition at line 82 of file rules-bones.hpp.

8.39.3 Member Function Documentation

8.39.3.1 add_rule() [1/2]

```
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 72 of file rules-meat.hpp.

8.39.3.2 add_rule() [2/2]

```
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_,
            std::string name_ = "",
            std::string description_ = "" )  [inline]
```

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

8.39.3.3 begin()

```
template<typename Array_Type , typename Data_Type >
std::vector< Rule<Array_Type,Data_Type> >::iterator Rules< Array_Type, Data_Type >::begin (
)  [inline]
```

Definition at line 134 of file rules-bones.hpp.

8.39.3.4 end()

```
template<typename Array_Type , typename Data_Type >
std::vector< Rule<Array_Type,Data_Type> >::iterator Rules< Array_Type, Data_Type >::end ( )
[inline]
```

Definition at line 137 of file rules-bones.hpp.

8.39.3.5 get_descriptions()

```
template<typename Array_Type , typename Data_Type >
std::vector< std::string > Rules< Array_Type, Data_Type >::get_descriptions  [inline]
```

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

8.39.3.6 get_names()

```
template<typename Array_Type , typename Data_Type >
std::vector< std::string > Rules< Array_Type, Data_Type >::get_names  [inline]
```

Definition at line 167 of file rules-meat.hpp.

8.39.3.7 get_seq()

```
template<typename Array_Type , typename Data_Type >
void Rules< Array_Type, Data_Type >::get_seq (
            const Array_Type & a,
            std::vector< size_t > * free,
            std::vector< size_t > * 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 117 of file rules-meat.hpp.

8.39.3.8 operator()()

```
template<typename Array_Type , typename Data_Type >
bool Rules< Array_Type, Data_Type >::operator() (
            const Array_Type & a,
            size_t i,
            size_t 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 101 of file rules-meat.hpp.

8.39.3.9 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 19 of file rules-meat.hpp.

8.39.3.10 size()

```
template<typename Array_Type , typename Data_Type >
size_t Rules< Array_Type, Data_Type >::size ( ) const  [inline], [noexcept]
```

Definition at line 84 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

8.40 StatsCounter< Array_Type, Data_Type > Class Template Reference

Count stats for a single Array.

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

Public Member Functions

- StatsCounter (const Array_Type *Array_)

  Creator of a `StatsCounter`
- StatsCounter (const StatsCounter< Array_Type, Data_Type > &counter)

  Copy constructor.
- 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 set_counters (Counters< Array_Type, Data_Type > *counters_)
- void count_init (size_t i, size_t 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 (size_t i, size_t j)
- std::vector< double > count_all ()
- Counters< Array_Type, Data_Type > * get_counters ()
- std::vector< std::string > get_names () const
- std::vector< std::string > get_descriptions () const
- size_t size () const

8.40.1 Detailed Description

template<typename Array_Type, typename Data_Type>
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 14 of file statscounter-bones.hpp.

8.40.2 Constructor & Destructor Documentation

8.40.2.1 StatsCounter() [1/3]

```
template<typename Array_Type , typename Data_Type >
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 37 of file statscounter-bones.hpp.

8.40.2.2 StatsCounter() [2/3]

```
template<typename Array_Type , typename Data_Type >
StatsCounter< Array_Type, Data_Type >::StatsCounter (
            const StatsCounter< Array_Type, Data_Type > & counter )
```

Copy constructor.

Parameters

counter	

8.40.2.3 StatsCounter() [3/3]

```
template<typename Array_Type , typename Data_Type >
StatsCounter< Array_Type, Data_Type >::StatsCounter ( )  [inline]
```

Can be created without setting the array.

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

8.40.2.4 ~StatsCounter()

```
template<typename Array_Type , typename Data_Type >
StatsCounter< Array_Type, Data_Type >::~StatsCounter ( )
```

8.40.3 Member Function Documentation

8.40.3.1 add_counter()

```
template<typename Array_Type , typename Data_Type >
void StatsCounter< Array_Type, Data_Type >::add_counter (
            Counter< Array_Type, Data_Type > f_ )
```

8.40.3.2 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.

8.40.3.3 count_current()

```
template<typename Array_Type , typename Data_Type >
void StatsCounter< Array_Type, Data_Type >::count_current (
            size_t i,
            size_t j )
```

8.40.3.4 count_init()

```
template<typename Array_Type , typename Data_Type >
void StatsCounter< Array_Type, Data_Type >::count_init (
            size_t i,
            size_t 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.

8.40.3.5 get_counters()

```
template<typename Array_Type , typename Data_Type >
Counters<Array_Type,Data_Type>* StatsCounter< Array_Type, Data_Type >::get_counters ( )
```

8.40.3.6 get_descriptions()

```
template<typename Array_Type , typename Data_Type >
std::vector< std::string > StatsCounter< Array_Type, Data_Type >::get_descriptions ( ) const
```

8.40.3.7 get_names()

```
template<typename Array_Type , typename Data_Type >
std::vector< std::string > StatsCounter< Array_Type, Data_Type >::get_names ( ) const
```

8.40.3.8 reset_array()

```
template<typename Array_Type , typename Data_Type >
void StatsCounter< Array_Type, Data_Type >::reset_array (
            const Array_Type * Array_ )
```

Changes the reference array for the counting.

Parameters

*Array_*	A pointer to an array of class `Array_Type`.

8.40.3.9 set_counters()

```
template<typename Array_Type , typename Data_Type >
void StatsCounter< Array_Type, Data_Type >::set_counters (
            Counters< Array_Type, Data_Type > * counters_ )
```

8.40.3.10 size()

```
template<typename Array_Type , typename Data_Type >
size_t StatsCounter< Array_Type, Data_Type >::size ( ) const  [inline]
```

Definition at line 86 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

8.41 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>
```

Public Member Functions

- Support (const Array_Type &Array_)
  Constructor passing a reference Array.
- Support (size_t N_, size_t M_)
  Constructor specifying the dimensions of the array (empty).
- Support ()
- ~Support ()
- void init_support (std::vector< Array_Type > *array_bank=nullptr, std::vector< double > *stats_bank=nullptr)
- void calc (std::vector< Array_Type > *array_bank=nullptr, std::vector< double > *stats_bank=nullptr, size_t max_num_elements_=0u)
  Computes the entire support.

- const std::vector< double > & get_counts () const
- std::vector< double > * get_current_stats ()

  List current statistics.
- void print () const
- const FreqTable< double > & get_data () const
- Counters< Array_Type, Data_Counter_Type > * get_counters ()

  Vector of couter functions.
- Rules< Array_Type, Data_Rule_Type > * get_rules ()

  Vector of static rules (cells to iterate).
- Rules< Array_Type, Data_Rule_Dyn_Type > * get_rules_dyn ()

  Vector of dynamic rules (to include/exclude a realizaton).

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 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_)
- bool eval_rules_dyn (const std::vector< double > &counts, const size_t &i, const size_t &j)

Public Attributes

- size_t N
- size_t M
- bool delete_counters = true
- bool delete_rules = true
- bool delete_rules_dyn = true
- size_t max_num_elements = BARRY_MAX_NUM_ELEMENTS
- std::vector< double > current_stats
- std::vector< size_t > coordinates_free
- std::vector< size_t > coordinates_locked
- size_t coordiantes_n_free
- size_t coordiantes_n_locked
- std::vector< double > change_stats
- std::vector< size_t > hashes
- std::vector< bool > hashes_initialized
- size_t n_counters

8.41.1 Detailed Description

template<typename Array_Type = BArray<bool, bool>, 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 42 of file support-bones.hpp.

8.41.2 Constructor & Destructor Documentation

8.41.2.1 Support() [1/3]

```
template<typename Array_Type = BArray<bool, bool>, 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 89 of file support-bones.hpp.

8.41.2.2 Support() [2/3]

```
template<typename Array_Type = BArray<bool, bool>, 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 (
            size_t N_,
            size_t M_ )  [inline]
```

Constructor specifying the dimensions of the array (empty).

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

8.41.2.3 Support() [3/3]

```
template<typename Array_Type = BArray<bool, bool>, 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 105 of file support-bones.hpp.

8.41.2.4 ~Support()

```
template<typename Array_Type = BArray<bool, bool>, 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 112 of file support-bones.hpp.

8.41.3 Member Function Documentation

8.41.3.1 add_counter()

```
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 417 of file support-meat.hpp.

8.41.3.2 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 444 of file support-meat.hpp.

8.41.3.3 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 454 of file support-meat.hpp.

8.41.3.4 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 479 of file support-meat.hpp.

8.41.3.5 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 489 of file support-meat.hpp.

8.41.3.6 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< double > * stats_bank = nullptr,
            size_t 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 383 of file support-meat.hpp.

8.41.3.7 eval_rules_dyn()

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
bool Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::eval_↩
rules_dyn (
            const std::vector< double > & counts,
            const size_t & i,
            const size_t & j )  [inline]
```

Definition at line 514 of file support-meat.hpp.

8.41.3.8 get_counters()

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
Counters< Array_Type, Data_Counter_Type > * Support< Array_Type, Data_Counter_Type, Data_↩
Rule_Type, Data_Rule_Dyn_Type >::get_counters  [inline]
```

Vector of couter functions.

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

8.41.3.9 get_counts()

```
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_counts  [inline]
```

Definition at line 557 of file support-meat.hpp.

8.41.3.10 get_current_stats()

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
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 571 of file support-meat.hpp.

8.41.3.11 get_data()

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
const FreqTable< double > & Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_↩
Rule_Dyn_Type >::get_data  [inline]
```

Definition at line 588 of file support-meat.hpp.

8.41.3.12 get_rules()

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
Rules< Array_Type, Data_Rule_Type > * Support< Array_Type, Data_Counter_Type, Data_Rule_Type,
Data_Rule_Dyn_Type >::get_rules  [inline]
```

Vector of static rules (cells to iterate).

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

8.41.3.13 get_rules_dyn()

```
template<typename Array_Type , typename Data_Counter_Type , typename Data_Rule_Type , typename
Data_Rule_Dyn_Type >
Rules< Array_Type, Data_Rule_Dyn_Type > * Support< Array_Type, Data_Counter_Type, Data_Rule↩
_Type, Data_Rule_Dyn_Type >::get_rules_dyn  [inline]
```

Vector of dynamic rules (to include/exclude a realizaton).

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

8.41.3.14 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< double > * stats_bank = nullptr )  [inline]
```

Definition at line 5 of file support-meat.hpp.

8.41.3.15 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 576 of file support-meat.hpp.

8.41.3.16 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 111 of file support-meat.hpp.

8.41.3.17 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 118 of file support-meat.hpp.

8.41.3.18 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 427 of file support-meat.hpp.

8.41.3.19 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 464 of file support-meat.hpp.

8.41.3.20 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 499 of file support-meat.hpp.

8.41.4 Member Data Documentation

8.41.4.1 change_stats

```
template<typename Array_Type = BArray<bool, bool>, 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 >::change_stats
```

Definition at line 82 of file support-bones.hpp.

8.41.4.2 coordiantes_n_free

```
template<typename Array_Type = BArray<bool, bool>, typename Data_Counter_Type = bool, typename
Data_Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
size_t Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::coordiantes↩
_n_free
```

Definition at line 80 of file support-bones.hpp.

8.41.4.3 coordiantes_n_locked

```
template<typename Array_Type = BArray<bool, bool>, typename Data_Counter_Type = bool, typename
Data_Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
size_t Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::coordiantes↩
_n_locked
```

Definition at line 81 of file support-bones.hpp.

8.41.4.4 coordinates_free

```
template<typename Array_Type = BArray<bool, bool>, typename Data_Counter_Type = bool, typename
Data_Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
std::vector< size_t > Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn↩
_Type >::coordinates_free
```

Definition at line 78 of file support-bones.hpp.

8.41.4.5 coordinates_locked

```
template<typename Array_Type = BArray<bool, bool>, typename Data_Counter_Type = bool, typename
Data_Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
std::vector< size_t > Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn↩
_Type >::coordinates_locked
```

Definition at line 79 of file support-bones.hpp.

8.41.4.6 current_stats

```
template<typename Array_Type = BArray<bool, bool>, 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 77 of file support-bones.hpp.

8.41.4.7 delete_counters

```
template<typename Array_Type = BArray<bool, bool>, 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 71 of file support-bones.hpp.

8.41.4.8 delete_rules

```
template<typename Array_Type = BArray<bool, bool>, 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 72 of file support-bones.hpp.

8.41.4.9 delete_rules_dyn

```
template<typename Array_Type = BArray<bool, bool>, 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 73 of file support-bones.hpp.

8.41.4.10 hashes

```
template<typename Array_Type = BArray<bool, bool>, typename Data_Counter_Type = bool, typename
Data_Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
std::vector< size_t > Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn↩
_Type >::hashes
```

Definition at line 83 of file support-bones.hpp.

8.41.4.11 hashes_initialized

```
template<typename Array_Type = BArray<bool, bool>, typename Data_Counter_Type = bool, typename
Data_Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
std::vector< bool > Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_↩
Type >::hashes_initialized
```

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

8.41.4.12 M

```
template<typename Array_Type = BArray<bool, bool>, typename Data_Counter_Type = bool, typename
Data_Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
size_t Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::M
```

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

8.41.4.13 max_num_elements

```
template<typename Array_Type = BArray<bool, bool>, typename Data_Counter_Type = bool, typename
Data_Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
size_t Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::max_num↩
_elements = BARRY_MAX_NUM_ELEMENTS
```

Definition at line 74 of file support-bones.hpp.

8.41.4.14 N

```
template<typename Array_Type = BArray<bool, bool>, typename Data_Counter_Type = bool, typename
Data_Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
size_t Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::N
```

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

8.41.4.15 n_counters

```
template<typename Array_Type = BArray<bool, bool>, typename Data_Counter_Type = bool, typename
Data_Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
size_t Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::n_↩
counters
```

Definition at line 85 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

8.42 vecHasher< T > Struct Template Reference

```
#include <typedefs.hpp>
```

Public Member Functions

- std::size_t operator() (std::vector< T > const &dat) const noexcept

8.42.1 Detailed Description

template<typename T>
struct vecHasher< T >

Definition at line 105 of file typedefs.hpp.

8.42.2 Member Function Documentation

8.42.2.1 operator()()

```
template<typename T >
std::size_t vecHasher< T >::operator() (
            std::vector< T > const & dat ) const  [inline], [noexcept]
```

Definition at line 108 of file typedefs.hpp.

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

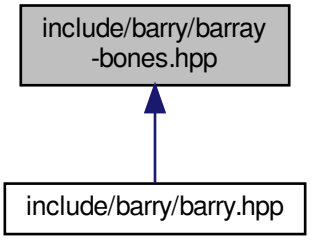
- include/barry/typedefs.hpp

Chapter 9

File Documentation

9.1 include/barry/barray-bones.hpp File Reference

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



Classes

- class BArray< Cell_Type, Data_Type >

  Baseline class for binary arrays.

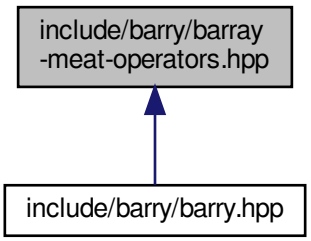
9.2 include/barry/barray-iterator.hpp File Reference

Classes

- class ConstBArrayRowIter< Cell_Type, Data_Type >

9.3 include/barry/barray-meat-operators.hpp File Reference

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



Macros

- #define BARRAY_TYPE() BArray<Cell_Type, Data_Type>
- #define BARRAY_TEMPLATE_ARGS() <typename Cell_Type, typename Data_Type>
- #define BARRAY_TEMPLATE(a, b) template BARRAY_TEMPLATE_ARGS() inline a BARRAY_TYPE()::b
- #define ROW(a) this->el_ij[a]
- #define COL(a) this->el_ji[a]

Functions

- template BARRAY_TEMPLATE_ARGS () inline void checkdim_(const BARRAY_TYPE() &lhs
- template const BARRAY_TYPE () &rhs)
- BARRAY_TEMPLATE (BARRAY_TYPE()&, operator+=)(const BArray< Cell_Type
- for (size_t i=0u;i< nrow();++i) for(size_t j=0u
- j< ncol();++j) this-> operator() (i, j)+
- BARRAY_TEMPLATE (BARRAY_TYPE()&, operator+=)(const Cell_Type &rhs)
- BARRAY_TEMPLATE (BARRAY_TYPE()&, operator-=)(const BArray< Cell_Type
- BARRAY_TEMPLATE (BARRAY_TYPE()&, operator-=)(const Cell_Type &rhs)
- BARRAY_TEMPLATE (BARRAY_TYPE()&, operator*=)(const Cell_Type &rhs)
- BARRAY_TEMPLATE (BARRAY_TYPE()&, operator/=)(const Cell_Type &rhs)

Variables

- Data_Type & rhs
- return * this

9.3.1 Macro Definition Documentation

9.3.1.1 BARRAY_TEMPLATE

```
#define BARRAY_TEMPLATE(
              a,
              b )  template BARRAY_TEMPLATE_ARGS() inline a BARRAY_TYPE()::b
```

Definition at line 11 of file barray-meat-operators.hpp.

9.3.1.2 BARRAY_TEMPLATE_ARGS

```
#define BARRAY_TEMPLATE_ARGS( ) <typename Cell_Type, typename Data_Type>
```

Definition at line 9 of file barray-meat-operators.hpp.

9.3.1.3 BARRAY_TYPE

```
#define BARRAY_TYPE( ) BArray<Cell_Type, Data_Type>
```

Definition at line 7 of file barray-meat-operators.hpp.

9.3.1.4 COL

```
#define COL(
              a ) this->el_ji[a]
```

Definition at line 15 of file barray-meat-operators.hpp.

9.3.1.5 ROW

```
#define ROW(
              a ) this->el_ij[a]
```

Definition at line 14 of file barray-meat-operators.hpp.

9.3.2 Function Documentation

9.3.2.1 BARRAY_TEMPLATE() [1/6]

```
BARRAY_TEMPLATE (
            BARRAY_TYPE()& ,
            operator*  ) const &
```

Definition at line 88 of file barray-meat-operators.hpp.

9.3.2.2 BARRAY_TEMPLATE() [2/6]

```
BARRAY_TEMPLATE (
            BARRAY_TYPE()& ,
            operator+  ) const
```

9.3.2.3 BARRAY_TEMPLATE() [3/6]

```
BARRAY_TEMPLATE (
            BARRAY_TYPE()& ,
            operator+  ) const &
```

Definition at line 46 of file barray-meat-operators.hpp.

9.3.2.4 BARRAY_TEMPLATE() [4/6]

```
BARRAY_TEMPLATE (
            BARRAY_TYPE()& ,
            operator-  ) const
```

9.3.2.5 BARRAY_TEMPLATE() [5/6]

```
BARRAY_TEMPLATE (
            BARRAY_TYPE()& ,
            operator-  ) const &
```

Definition at line 75 of file barray-meat-operators.hpp.

9.3.2.6 BARRAY_TEMPLATE() [6/6]

```
BARRAY_TEMPLATE (
              BARRAY_TYPE()& ,
              operator/  ) const &
```

Definition at line 105 of file barray-meat-operators.hpp.

9.3.2.7 BARRAY_TEMPLATE_ARGS()

```
template BARRAY_TEMPLATE_ARGS ( ) const &
```

9.3.2.8 BARRAY_TYPE()

```
template const BARRAY_TYPE ( ) &
```

Definition at line 20 of file barray-meat-operators.hpp.

9.3.2.9 for()

```
for ( )  [pure virtual]
```

Definition at line 66 of file barray-meat-operators.hpp.

9.3.2.10 operator()()

```
j< ncol(); ++j) this-> operator() (
              i ,
              j  )
```

9.3.3 Variable Documentation

9.3.3.1 rhs

```
Data_Type & rhs
```

Initial value:

```
{
    checkdim_(*this, rhs)
```

Definition at line 33 of file barray-meat-operators.hpp.

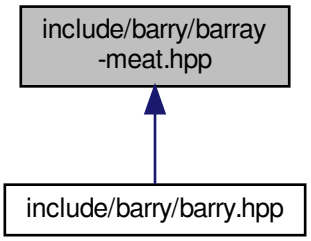
9.3.3.2 this

```
return * this
```

Definition at line 43 of file barray-meat-operators.hpp.

9.4 include/barry/barray-meat.hpp File Reference

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]

9.4.1 Macro Definition Documentation

9.4.1.1 COL

```
#define COL(
              a ) this->el_ji[a]
```

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

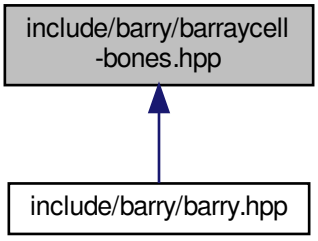
9.4.1.2 ROW

```
#define ROW(
              a ) this->el_ij[a]
```

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

9.5 include/barry/barraycell-bones.hpp File Reference

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

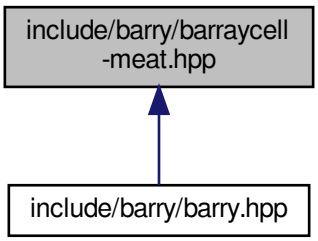


Classes

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

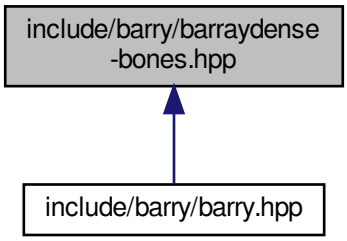
9.6 include/barry/barraycell-meat.hpp File Reference

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



9.7 include/barry/barraydense-bones.hpp File Reference

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



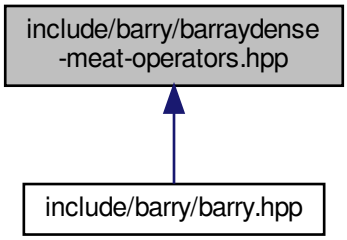
Classes

- class BArrayDense< Cell_Type, Data_Type >

  Baseline class for binary arrays.

9.8 include/barry/barraydense-meat-operators.hpp File Reference

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



Macros

- #define BDENSE_TYPE() BArrayDense<Cell_Type, Data_Type>
- #define BDENSE_TEMPLATE_ARGS() <typename Cell_Type, typename Data_Type>
- #define BDENSE_TEMPLATE(a, b) template BDENSE_TEMPLATE_ARGS() inline a BDENSE_TYPE()::b
- #define ROW(a) this->el_ij[a]
- #define COL(a) this->el_ji[a]
- #define POS(a, b) (b)*N + (a)
- #define POS_N(a, b, c) (b)*(c) + (a)

Functions

- template BDENSE_TEMPLATE_ARGS () inline void checkdim_(const BDENSE_TYPE() &lhs
- template const BDENSE_TYPE () &rhs)
- BDENSE_TEMPLATE (BDENSE_TYPE()&, operator+=)(const BDENSE_TYPE() &rhs)
- BDENSE_TEMPLATE (BDENSE_TYPE()&, operator-=)(const BDENSE_TYPE() &rhs)
- BDENSE_TEMPLATE (BDENSE_TYPE()&, operator*=)(const Cell_Type &rhs)
- BDENSE_TEMPLATE (BDENSE_TYPE()&, operator/=)(const Cell_Type &rhs)

9.8.1 Macro Definition Documentation

9.8.1.1 BDENSE_TEMPLATE

```
#define BDENSE_TEMPLATE(
            a,
            b )  template BDENSE_TEMPLATE_ARGS() inline a BDENSE_TYPE()::b
```

Definition at line 11 of file barraydense-meat-operators.hpp.

9.8.1.2 BDENSE_TEMPLATE_ARGS

```
#define BDENSE_TEMPLATE_ARGS( ) <typename Cell_Type, typename Data_Type>
```

Definition at line 9 of file barraydense-meat-operators.hpp.

9.8.1.3 BDENSE_TYPE

```
#define BDENSE_TYPE( ) BArrayDense<Cell_Type, Data_Type>
```

Definition at line 7 of file barraydense-meat-operators.hpp.

9.8.1.4 COL

```
#define COL(
              a ) this->el_ji[a]
```

Definition at line 15 of file barraydense-meat-operators.hpp.

9.8.1.5 POS

```
#define POS(
              a,
              b ) (b)*N + (a)
```

Definition at line 16 of file barraydense-meat-operators.hpp.

9.8.1.6 POS_N

```
#define POS_N(
              a,
              b,
              c ) (b)*(c) + (a)
```

Definition at line 17 of file barraydense-meat-operators.hpp.

9.8.1.7 ROW

```
#define ROW(
              a ) this->el_ij[a]
```

Definition at line 14 of file barraydense-meat-operators.hpp.

9.8.2 Function Documentation

9.8.2.1 BDENSE_TEMPLATE() [1/4]

```
BDENSE_TEMPLATE (
             BDENSE_TYPE()& ,
             operator*  ) const &
```

Definition at line 90 of file barraydense-meat-operators.hpp.

9.8.2.2 BDENSE_TEMPLATE() [2/4]

```
BDENSE_TEMPLATE (
             BDENSE_TYPE()& ,
             operator+  ) const &
```

Definition at line 34 of file barraydense-meat-operators.hpp.

9.8.2.3 BDENSE_TEMPLATE() [3/4]

```
BDENSE_TEMPLATE (
             BDENSE_TYPE()& ,
             operator-  ) const &
```

Definition at line 61 of file barraydense-meat-operators.hpp.

9.8.2.4 BDENSE_TEMPLATE() [4/4]

```
BDENSE_TEMPLATE (
             BDENSE_TYPE()& ,
             operator/  ) const &
```

Definition at line 101 of file barraydense-meat-operators.hpp.

9.8.2.5 BDENSE_TEMPLATE_ARGS()

```
template BDENSE_TEMPLATE_ARGS ( ) const &
```

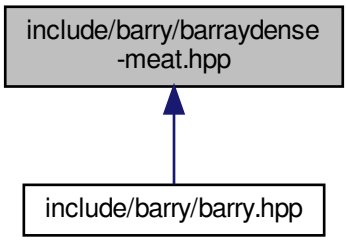
9.8.2.6 BDENSE_TYPE()

```
template const BDENSE_TYPE ( ) &
```

Definition at line 22 of file barraydense-meat-operators.hpp.

9.9 include/barry/barraydense-meat.hpp File Reference

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]
- #define POS(a, b) (b)*N + (a)
- #define POS_N(a, b, c) (b)*(c) + (a)
- #define ZERO_CELL static_cast<Cell_Type>(0.0)

9.9.1 Macro Definition Documentation

9.9.1.1 COL

```
#define COL(
              a ) this->el_ji[a]
```

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

9.9.1.2 POS

```
#define POS(
              a,
              b ) (b)*N + (a)
```

Definition at line 25 of file barraydense-meat.hpp.

9.9.1.3 POS_N

```
#define POS_N(
              a,
              b,
              c ) (b)*(c) + (a)
```

Definition at line 26 of file barraydense-meat.hpp.

9.9.1.4 ROW

```
#define ROW(
              a ) this->el_ij[a]
```

Definition at line 23 of file barraydense-meat.hpp.

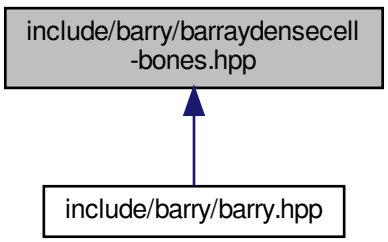
9.9.1.5 ZERO_CELL

```
#define ZERO_CELL static_cast<Cell_Type>(0.0)
```

Definition at line 31 of file barraydense-meat.hpp.

9.10 include/barry/barraydensecell-bones.hpp File Reference

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



Classes

- class BArrayDenseCell< Cell_Type, Data_Type >

Macros

- #define POS(a, b) (a) + (b) * N

9.10.1 Macro Definition Documentation

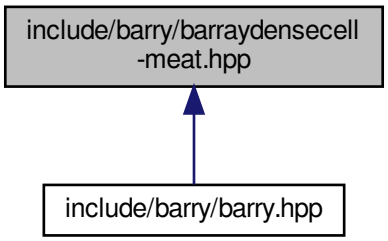
9.10.1.1 POS

```
#define POS(
              a,
              b ) (a) + (b) * N
```

Definition at line 6 of file barraydensecell-bones.hpp.

9.11 include/barry/barraydensecell-meat.hpp File Reference

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



Macros

- #define POS(a, b) (a) + (b) * dat->N

9.11.1 Macro Definition Documentation

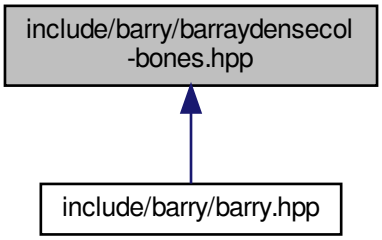
9.11.1.1 POS

```
#define POS(
              a,
              b ) (a) + (b) * dat->N
```

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

9.12 include/barry/barraydensecol-bones.hpp File Reference

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



Classes

- class BArrayDenseCol< Cell_Type, Data_Type >
- class BArrayDenseCol_const< Cell_Type, Data_Type >

Macros

- #define POS(a, b) (b)*N + (a)
- #define POS_N(a, b, c) (b)*(c) + (a)
- #define ZERO_CELL static_cast<Cell_Type>(0.0)

9.12.1 Macro Definition Documentation

9.12.1.1 POS

```
#define POS(
              a,
              b ) (b)*N + (a)
```

Definition at line 4 of file barraydensecol-bones.hpp.

9.12.1.2 POS_N

```
#define POS_N(
              a,
              b,
              c ) (b)*(c) + (a)
```

Definition at line 5 of file barraydensecol-bones.hpp.

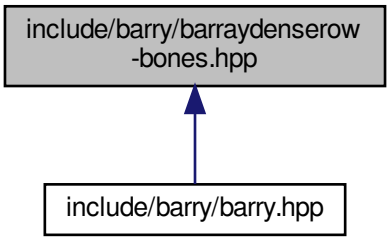
9.12.1.3 ZERO_CELL

```
#define ZERO_CELL static_cast<Cell_Type>(0.0)
```

Definition at line 6 of file barraydensecol-bones.hpp.

9.13 include/barry/barraydenserow-bones.hpp File Reference

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



Classes

- class BArrayDenseRow< Cell_Type, Data_Type >
- class BArrayDenseRow_const< Cell_Type, Data_Type >

Macros

- #define POS(a, b) (b) * N + (a)
- #define POS_N(a, b, c) (b)*(c) + (a)
- #define ZERO_CELL static_cast< Cell_Type >(0.0)

9.13.1 Macro Definition Documentation

9.13.1.1 POS

```
#define POS(
              a,
              b ) (b) * N + (a)
```

Definition at line 4 of file barraydenserow-bones.hpp.

9.13.1.2 POS_N

```
#define POS_N(
              a,
              b,
              c ) (b)*(c) + (a)
```

Definition at line 5 of file barraydenserow-bones.hpp.

9.13.1.3 ZERO_CELL

```
#define ZERO_CELL static_cast< Cell_Type >(0.0)
```

Definition at line 6 of file barraydenserow-bones.hpp.

9.14 include/barry/barrayrow-bones.hpp File Reference

Classes

- class BArrayRow< Cell_Type, Data_Type >
- class BArrayRow_const< Cell_Type, Data_Type >

9.15 include/barry/barrayrow-meat.hpp File Reference

Macros

- #define BROW_TYPE() BArrayRow<Cell_Type, Data_Type>
- #define BROW_TEMPLATE_ARGS() <typename Cell_Type, typename Data_Type>
- #define BROW_TEMPLATE(a, b) template BROW_TEMPLATE_ARGS() inline a BROW_TYPE()::b

Functions

- BROW_TEMPLATE (void, operator=)(const BROW_TYPE() &val)
- BROW_TEMPLATE (void, operator+=)(const BROW_TYPE() &val)
- BROW_TEMPLATE (void, operator-=)(const BROW_TYPE() &val)
- BROW_TEMPLATE (void, operator∗=)(const BROW_TYPE() &val)
- BROW_TEMPLATE (void, operator/=)(const BROW_TYPE() &val)

9.15.1 Macro Definition Documentation

9.15.1.1 BROW_TEMPLATE

```
#define BROW_TEMPLATE(
              a,
              b )  template BROW_TEMPLATE_ARGS() inline a BROW_TYPE()::b
```

Definition at line 8 of file barrayrow-meat.hpp.

9.15.1.2 BROW_TEMPLATE_ARGS

```
#define BROW_TEMPLATE_ARGS( ) <typename Cell_Type, typename Data_Type>
```

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

9.15.1.3 BROW_TYPE

```
#define BROW_TYPE( ) BArrayRow<Cell_Type, Data_Type>
```

Definition at line 4 of file barrayrow-meat.hpp.

9.15.2 Function Documentation

9.15.2.1 BROW_TEMPLATE() [1/5]

```
BROW_TEMPLATE (
             void ,
             operator*  ) const &
```

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

9.15.2.2 BROW_TEMPLATE() [2/5]

```
BROW_TEMPLATE (
            void ,
            operator+  ) const &
```

Definition at line 25 of file barrayrow-meat.hpp.

9.15.2.3 BROW_TEMPLATE() [3/5]

```
BROW_TEMPLATE (
            void ,
            operator-  ) const &
```

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

9.15.2.4 BROW_TEMPLATE() [4/5]

```
BROW_TEMPLATE (
            void ,
            operator/  ) const &
```

Definition at line 55 of file barrayrow-meat.hpp.

9.15.2.5 BROW_TEMPLATE() [5/5]

```
BROW_TEMPLATE (
            void ,
            operator  ) const &
```

Definition at line 11 of file barrayrow-meat.hpp.

9.16 include/barry/barrayvector-bones.hpp File Reference

Classes

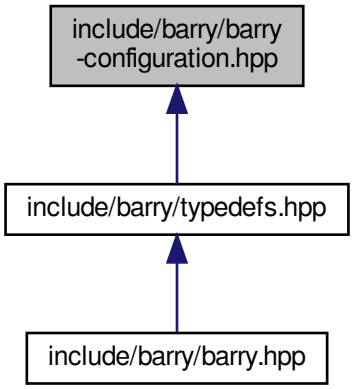
- class BArrayVector< Cell_Type, Data_Type >

  Row or column of a BArray
- class BArrayVector_const< Cell_Type, Data_Type >

9.17 include/barry/barrayvector-meat.hpp File Reference

9.18 include/barry/barry-configuration.hpp File Reference

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



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_USE_ISFINITE` When specified, it will introduce a macro that checks whether the likelihood is finite or not.
- `printf_barry` If not specified, will be defined as `printf`.
- `BARRY_DEBUG_LEVEL`, when defined, will make things verbose.

- #define BARRY_SAFE_EXP -100.0
- #define BARRY_ISFINITE(a)
- #define BARRY_CHECK_SUPPORT(x, maxs)
- #define printf_barry printf
- #define BARRY_MAX_NUM_ELEMENTS static_cast< size_t >(std::numeric_limits< size_t >::max() /2u)
- template<typename Ta , typename Tb >
  using Map = std::map< Ta, Tb >

9.18.1 Macro Definition Documentation

9.18.1.1 BARRY_CHECK_SUPPORT

```
#define BARRY_CHECK_SUPPORT(
              x,
              maxs )
```

Definition at line 47 of file barry-configuration.hpp.

9.18.1.2 BARRY_ISFINITE

```
#define BARRY_ISFINITE(
              a )
```

Definition at line 40 of file barry-configuration.hpp.

9.18.1.3 BARRY_MAX_NUM_ELEMENTS

```
#define BARRY_MAX_NUM_ELEMENTS static_cast< size_t >(std::numeric_limits< size_t >::max()
/2u)
```

Definition at line 55 of file barry-configuration.hpp.

9.18.1.4 BARRY_SAFE_EXP

```
#define BARRY_SAFE_EXP -100.0
```

Definition at line 33 of file barry-configuration.hpp.

9.18.1.5 printf_barry

```
#define printf_barry printf
```

Definition at line 51 of file barry-configuration.hpp.

9.18.2 Typedef Documentation

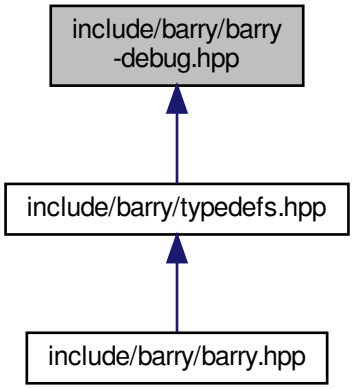
9.18.2.1 Map

```
template<typename Ta , typename Tb >
using Map = std::map<Ta,Tb>
```

Definition at line 27 of file barry-configuration.hpp.

9.19 include/barry/barry-debug.hpp File Reference

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



Macros

- #define BARRY_DEBUG_LEVEL 0

9.19.1 Macro Definition Documentation

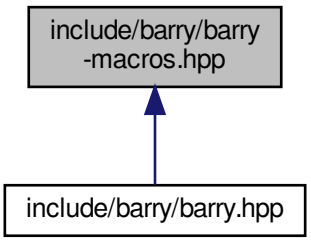
9.19.1.1 BARRY_DEBUG_LEVEL

```
#define BARRY_DEBUG_LEVEL 0
```

Definition at line 5 of file barry-debug.hpp.

9.20 include/barry/barry-macros.hpp File Reference

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



Macros

- #define BARRY_ZERO Cell<Cell_Type>(0.0)
- #define BARRY_ZERO_DENSE static_cast<Cell_Type>(0.0)
- #define BARRY_ONE Cell<Cell_Type>(1.0)
- #define BARRY_ONE_DENSE static_cast<Cell_Type>(1.0)
- #define BARRY_UNUSED(expr) do { (void)(expr); } while (0);
- #define BARRY_NCORES_ARG(default) size_t ncores default

9.20.1 Macro Definition Documentation

9.20.1.1 BARRY_NCORES_ARG

```
#define BARRY_NCORES_ARG(
              default ) size_t ncores default
```

Definition at line 15 of file barry-macros.hpp.

9.20.1.2 BARRY_ONE

```
#define BARRY_ONE Cell<Cell_Type>(1.0)
```

Definition at line 7 of file barry-macros.hpp.

9.20.1.3 BARRY_ONE_DENSE

```
#define BARRY_ONE_DENSE static_cast<Cell_Type>(1.0)
```

Definition at line 8 of file barry-macros.hpp.

9.20.1.4 BARRY_UNUSED

```
#define BARRY_UNUSED(
              expr ) do { (void)(expr); } while (0);
```

Definition at line 10 of file barry-macros.hpp.

9.20.1.5 BARRY_ZERO

```
#define BARRY_ZERO Cell<Cell_Type>(0.0)
```

Definition at line 4 of file barry-macros.hpp.

9.20.1.6 BARRY_ZERO_DENSE

```
#define BARRY_ZERO_DENSE static_cast<Cell_Type>(0.0)
```

Definition at line 5 of file barry-macros.hpp.

9.21 include/barry/barry.hpp File Reference

```
#include <iostream>
#include <cstdarg>
#include <vector>
#include <unordered_map>
#include <functional>
#include <stdexcept>
#include <cmath>
#include <map>
#include <algorithm>
#include <utility>
#include <random>
#include <climits>
#include <cfloat>
#include <string>
#include <cstdint>
#include <memory>
#include <regex>
```

```
#include <iterator>
#include "typedefs.hpp"
#include "barry-macros.hpp"
#include "freqtable.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 "barraydense-bones.hpp"
#include "barraydensecell-bones.hpp"
#include "barraydenserow-bones.hpp"
#include "barraydensecol-bones.hpp"
#include "barraydense-meat.hpp"
#include "barraydensecell-meat.hpp"
#include "barraydense-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 dependency graph for barry.hpp:

Namespaces

- barry

  barry: Your go-to motif accountant
- barry::counters

  Tree class and TreeIterator class.
- barry::counters::network

Macros

- #define BARRY_HPP
- #define BARRY_VERSION_MAYOR 0
- #define BARRY_VERSION_MINOR 1
- #define BARRY_VERSION BARRY_VERSION_MAYOR ## . ## BARRY_VERSION_MINOR
- #define COUNTER_FUNCTION(a)
- #define COUNTER_LAMBDA(a)
- #define RULE_FUNCTION(a)
- #define RULE_LAMBDA(a)

9.21.1 Macro Definition Documentation

9.21.1.1 BARRY_HPP

```
#define BARRY_HPP
```

Definition at line 29 of file barry.hpp.

9.21.1.2 BARRY_VERSION

```
#define BARRY_VERSION BARRY_VERSION_MAYOR ## .  ## BARRY_VERSION_MINOR
```

Definition at line 33 of file barry.hpp.

9.21.1.3 BARRY_VERSION_MAYOR

```
#define BARRY_VERSION_MAYOR 0
```

Definition at line 31 of file barry.hpp.

9.21.1.4 BARRY_VERSION_MINOR

```
#define BARRY_VERSION_MINOR 1
```

Definition at line 32 of file barry.hpp.

9.21.1.5 COUNTER_FUNCTION

```
#define COUNTER_FUNCTION(
              a )
```

Value:

```
    template <typename Array_Type = barry::BArray<>, typename Data_Type = bool> \
    inline double (a) (const Array_Type & Array, size_t i, size_t j, Data_Type & data)\
```

Definition at line 92 of file barry.hpp.

9.21.1.6 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, size_t i, size_t j, Data_Type & data)
```

Definition at line 95 of file barry.hpp.

9.21.1.7 RULE_FUNCTION

```
#define RULE_FUNCTION(
                a )
```

Value:

```
template <typename Array_Type = barry::BArray<>, typename Data_Type = bool> \
inline bool (a) (const Array_Type & Array, size_t i, size_t j, Data_Type & data)\
```

Definition at line 99 of file barry.hpp.

9.21.1.8 RULE_LAMBDA

```
#define RULE_LAMBDA(
                a )
```

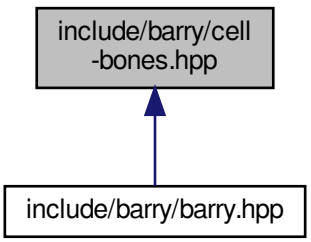
Value:

```
template <typename Array_Type = barry::BArray<>, typename Data_Type = bool> \
Rule_fun_type<Array_Type, Data_Type> a = \
[](const Array_Type & Array, size_t i, size_t j, Data_Type & data)
```

Definition at line 102 of file barry.hpp.

9.22 include/barry/cell-bones.hpp File Reference

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



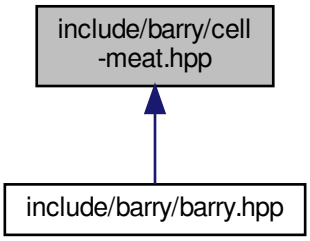
Classes

- class Cell< Cell_Type >

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

9.23 include/barry/cell-meat.hpp File Reference

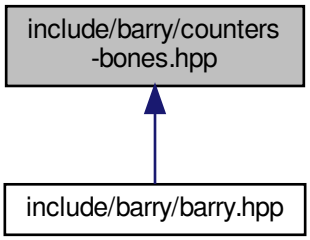
This graph shows which files directly or indirectly include this file:



9.24 include/barry/col-bones.hpp File Reference

9.25 include/barry/counters-bones.hpp File Reference

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



Classes

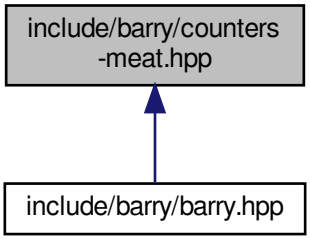
- class Counter< Array_Type, Data_Type >

  A counter function based on change statistics.
- class Counters< Array_Type, Data_Type >

  Vector of counters.

9.26 include/barry/counters-meat.hpp File Reference

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



Macros

- #define COUNTER_TYPE() Counter<Array_Type,Data_Type>
- #define COUNTER_TEMPLATE_ARGS() <typename Array_Type, typename Data_Type>
- #define COUNTER_TEMPLATE(a, b) template COUNTER_TEMPLATE_ARGS() inline a COUNTER_TYPE()↩ ::b
- #define TMP_HASHER_CALL Hasher_fun_type<Array_Type,Data_Type>
- #define COUNTERS_TYPE() Counters<Array_Type,Data_Type>
- #define COUNTERS_TEMPLATE_ARGS() <typename Array_Type, typename Data_Type>
- #define COUNTERS_TEMPLATE(a, b) template COUNTERS_TEMPLATE_ARGS() inline a COUNTERS_TYPE()↩ ::b

Functions

- COUNTER_TEMPLATE (, Counter)(const Counter< Array_Type
- Data_Type init_fun (counter_.init_fun)
- Data_Type hasher_fun (counter_.hasher_fun)
- Data_Type &&counter_ init_fun (std::move(counter_.init_fun))
- Data_Type &&counter_ hasher_fun (std::move(counter_.hasher_fun))
- Data_Type &&counter_ data (std::move(counter_.data))
- Data_Type &&counter_ name (std::move(counter_.name))
- Data_Type &&counter_ desc (std::move(counter_.desc))

  Move constructor.
- COUNTER_TEMPLATE (COUNTER_TYPE(), operator=)(const Counter< Array_Type
- COUNTER_TEMPLATE (COUNTER_TYPE() &, operator=)(Counter< Array_Type
- COUNTER_TEMPLATE (double, count)(Array_Type &Array

  < *Move assignment*
- return count_fun (Array, i, j, data)
- COUNTER_TEMPLATE (double, init)(Array_Type &Array
- return init_fun (Array, i, j, data)
- COUNTER_TEMPLATE (std::string, get_name)() const
- COUNTER_TEMPLATE (std::string, get_description)() const
- COUNTER_TEMPLATE (void, set_hasher)(Hasher_fun_type< Array_Type

- COUNTER_TEMPLATE (TMP_HASHER_CALL, get_hasher)()
- COUNTERS_TEMPLATE (, Counters)()
- COUNTERS_TEMPLATE (COUNTER_TYPE() &, operator[ ])(size_t idx)
- Data_Type hasher (counter_.hasher)
- Data_Type &&counters_ hasher (std::move(counters_.hasher))
- COUNTERS_TEMPLATE (COUNTERS_TYPE(), operator=)(const Counters< Array_Type
- COUNTERS_TEMPLATE (COUNTERS_TYPE() &, operator=)(Counters< Array_Type
- COUNTERS_TEMPLATE (void, add_counter)(Counter< Array_Type
- COUNTERS_TEMPLATE (std::vector< std::string >, get_names)() const
- COUNTERS_TEMPLATE (std::vector< std::string >, get_descriptions)() const
- COUNTERS_TEMPLATE (std::vector< double >, gen_hash)(const Array_Type &array
- for (auto &c:data)
- if (add_dims)
- if (hasher)
- if (res.size()==0u) res.push_back(0.0)
- COUNTERS_TEMPLATE (void, add_hash)(Hasher_fun_type< Array_Type

Variables

- Data_Type & counter_
- Data_Type &&counter_ noexcept
- size_t i
- size_t size_t j
- Data_Type fun
- Data_Type counter
- return
- Data_Type count_fun_
- Data_Type Counter_fun_type< Array_Type, Data_Type > init_fun_
- Data_Type Counter_fun_type< Array_Type, Data_Type > Hasher_fun_type< Array_Type, Data_Type > hasher_fun_
- Data_Type Counter_fun_type< Array_Type, Data_Type > Hasher_fun_type< Array_Type, Data_Type > Data_Type data_
- Data_Type Counter_fun_type< Array_Type, Data_Type > Hasher_fun_type< Array_Type, Data_Type > Data_Type std::string name_
- Data_Type Counter_fun_type< Array_Type, Data_Type > Hasher_fun_type< Array_Type, Data_Type > Data_Type std::string std::string desc_
- bool add_dims
- return res
- Data_Type fun_

9.26.1 Macro Definition Documentation

9.26.1.1 COUNTER_TEMPLATE

```
#define COUNTER_TEMPLATE(
            a,
            b )  template COUNTER_TEMPLATE_ARGS() inline a COUNTER_TYPE()::b
```

Definition at line 8 of file counters-meat.hpp.

9.26.1.2 COUNTER_TEMPLATE_ARGS

```
#define COUNTER_TEMPLATE_ARGS( ) <typename Array_Type, typename Data_Type>
```

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

9.26.1.3 COUNTER_TYPE

```
#define COUNTER_TYPE( ) Counter<Array_Type,Data_Type>
```

Definition at line 4 of file counters-meat.hpp.

9.26.1.4 COUNTERS_TEMPLATE

```
#define COUNTERS_TEMPLATE(
              a,
              b )  template COUNTERS_TEMPLATE_ARGS() inline a COUNTERS_TYPE()::b
```

Definition at line 129 of file counters-meat.hpp.

9.26.1.5 COUNTERS_TEMPLATE_ARGS

```
#define COUNTERS_TEMPLATE_ARGS( ) <typename Array_Type, typename Data_Type>
```

Definition at line 127 of file counters-meat.hpp.

9.26.1.6 COUNTERS_TYPE

```
#define COUNTERS_TYPE( ) Counters<Array_Type,Data_Type>
```

Definition at line 125 of file counters-meat.hpp.

9.26.1.7 TMP_HASHER_CALL

```
#define TMP_HASHER_CALL Hasher_fun_type<Array_Type,Data_Type>
```

Definition at line 115 of file counters-meat.hpp.

9.26.2 Function Documentation

9.26.2.1 count_fun()

```
return count_fun (
             Array ,
             i ,
             j ,
             data  )
```

9.26.2.2 COUNTER_TEMPLATE() [1/9]

```
COUNTER_TEMPLATE (
             Counter  ) const
```

9.26.2.3 COUNTER_TEMPLATE() [2/9]

```
COUNTER_TEMPLATE (
             COUNTER_TYPE() & ,
             operator  )
```

9.26.2.4 COUNTER_TEMPLATE() [3/9]

```
COUNTER_TEMPLATE (
             COUNTER_TYPE() ,
             operator  ) const
```

9.26.2.5 COUNTER_TEMPLATE() [4/9]

```
COUNTER_TEMPLATE (
             double ,
             count  ) &
```

< Move assignment

9.26.2.6 COUNTER_TEMPLATE() [5/9]

```
COUNTER_TEMPLATE (
            double ,
            init  ) &
```

9.26.2.7 COUNTER_TEMPLATE() [6/9]

```
COUNTER_TEMPLATE (
            std::string ,
            get_description  ) const
```

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

9.26.2.8 COUNTER_TEMPLATE() [7/9]

```
COUNTER_TEMPLATE (
            std::string ,
            get_name  ) const
```

Definition at line 103 of file counters-meat.hpp.

9.26.2.9 COUNTER_TEMPLATE() [8/9]

```
COUNTER_TEMPLATE (
            TMP_HASHER_CALL ,
            get_hasher  )
```

Definition at line 116 of file counters-meat.hpp.

9.26.2.10 COUNTER_TEMPLATE() [9/9]

```
COUNTER_TEMPLATE (
            void ,
            set_hasher  )
```

9.26.2.11 COUNTERS_TEMPLATE() [1/9]

```
COUNTERS_TEMPLATE (
            Counters  )
```

Definition at line 132 of file counters-meat.hpp.

9.26.2.12 COUNTERS_TEMPLATE() [2/9]

```
COUNTERS_TEMPLATE (
            COUNTER_TYPE() & ,
            operator [] )
```

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

9.26.2.13 COUNTERS_TEMPLATE() [3/9]

```
COUNTERS_TEMPLATE (
            COUNTERS_TYPE() & ,
            operator  )
```

9.26.2.14 COUNTERS_TEMPLATE() [4/9]

```
COUNTERS_TEMPLATE (
            COUNTERS_TYPE() ,
            operator  ) const
```

9.26.2.15 COUNTERS_TEMPLATE() [5/9]

```
COUNTERS_TEMPLATE (
            std::vector< double > ,
            gen_hash  ) const &
```

9.26.2.16 COUNTERS_TEMPLATE() [6/9]

```
COUNTERS_TEMPLATE (
            std::vector< std::string > ,
            get_descriptions  ) const
```

Definition at line 213 of file counters-meat.hpp.

9.26.2.17 COUNTERS_TEMPLATE() [7/9]

```
COUNTERS_TEMPLATE (
            std::vector< std::string > ,
            get_names  ) const
```

Definition at line 201 of file counters-meat.hpp.

9.26.2.18 COUNTERS_TEMPLATE() [8/9]

```
COUNTERS_TEMPLATE (
            void ,
            add_counter  )
```

9.26.2.19 COUNTERS_TEMPLATE() [9/9]

```
COUNTERS_TEMPLATE (
            void ,
            add_hash  )
```

9.26.2.20 data()

```
Data_Type&& counter_ data (
            std::move(counter_.data)  )
```

9.26.2.21 desc()

```
Data_Type&& counter_ desc (
            std::move(counter_.desc)  )
```

Move constructor.

Definition at line 32 of file counters-meat.hpp.

9.26.2.22 for()

```
for (
            auto &c:data  )
```

Definition at line 233 of file counters-meat.hpp.

9.26.2.23 hasher() [1/2]

```
Data_Type hasher (
            counter_.  hasher )
```

Definition at line 141 of file counters-meat.hpp.

9.26.2.24 hasher() [2/2]

```
Data_Type&& counters_ hasher (
            std::move(counters_.hasher)  )
```

Definition at line 144 of file counters-meat.hpp.

9.26.2.25 hasher_fun() [1/2]

```
Data_Type hasher_fun (
            counter_.  hasher_fun )
```

Definition at line 13 of file counters-meat.hpp.

9.26.2.26 hasher_fun() [2/2]

```
Data_Type&& counter_ hasher_fun (
            std::move(counter_.hasher_fun)  )
```

9.26.2.27 if() [1/3]

```
if (
            add_dims  )
```

Definition at line 248 of file counters-meat.hpp.

9.26.2.28 if() [2/3]

```
if (
            hasher  )
```

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

9.26.2.29 if() [3/3]

```
if (
             res.  size() = =0u )
```

9.26.2.30 init_fun() [1/3]

```
return init_fun (
             Array ,
             i ,
             j ,
             data  )
```

9.26.2.31 init_fun() [2/3]

```
Data_Type init_fun (
             counter_.  init_fun )
```

9.26.2.32 init_fun() [3/3]

```
Data_Type&& counter_ init_fun (
             std::move(counter_.init_fun)  )
```

9.26.2.33 name()

```
Data_Type&& counter_ name (
             std::move(counter_.name)  )
```

9.26.3 Variable Documentation

9.26.3.1 add_dims

```
bool add_dims
```

Initial value:

```
{
    std::vector<double> res
```

Definition at line 227 of file counters-meat.hpp.

9.26.3.2 count_fun_

```
Data_Type count_fun_
```

Definition at line 179 of file counters-meat.hpp.

9.26.3.3 counter

```
Data_Type counter
```

Initial value:

```
{

    data.push_back(counter)
```

Definition at line 170 of file counters-meat.hpp.

9.26.3.4 counter_

```
Data_Type & counter_
```

Initial value:

```
{
    if (this != &counter_) {
        this->count_fun = counter_.count_fun;
        this->init_fun = counter_.init_fun;
        this->hasher_fun = counter_.hasher_fun;

        this->data = counter_.data;
        this->name = counter_.name;
        this->desc = counter_.desc;
    }
    return *this
```

Definition at line 12 of file counters-meat.hpp.

9.26.3.5 data_

```
Data_Type Counter_fun_type<Array_Type,Data_Type> Hasher_fun_type<Array_Type,Data_Type> Data↩
_Type data_
```

Definition at line 182 of file counters-meat.hpp.

9.26.3.6 desc_

```
Data_Type Counter_fun_type<Array_Type,Data_Type> Hasher_fun_type<Array_Type,Data_Type> Data↩
_Type std::string std::string desc_
```

Initial value:

```
{

    data.emplace_back(Counter<Array_Type,Data_Type>(
        count_fun_,
        init_fun_,
        hasher_fun_,
        data_,
        name_,
        desc_
    ))
```

Definition at line 184 of file counters-meat.hpp.

9.26.3.7 fun

```
Data_Type fun
```

Initial value:

```
{
    hasher_fun = fun
```

Definition at line 111 of file counters-meat.hpp.

9.26.3.8 fun_

```
Data_Type fun_
```

Initial value:

```
{
    hasher = fun_
```

Definition at line 270 of file counters-meat.hpp.

9.26.3.9 hasher_fun_

```
Data_Type Counter_fun_type<Array_Type,Data_Type> Hasher_fun_type<Array_Type,Data_Type> hasher↩
_fun_
```

Definition at line 181 of file counters-meat.hpp.

9.26.3.10 i

```
size_t i
```

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

9.26.3.11 init_fun_

```
Data_Type Counter_fun_type<Array_Type,Data_Type> init_fun_
```

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

9.26.3.12 j

```
size_t j
```

Initial value:

```
{
    if (count_fun == nullptr)
        return 0.0
```

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

9.26.3.13 name_

```
Data_Type Counter_fun_type<Array_Type,Data_Type> Hasher_fun_type<Array_Type,Data_Type> Data↩
_Type std::string name_
```

Definition at line 183 of file counters-meat.hpp.

9.26.3.14 noexcept

```
Data_Type &&counters_ noexcept
```

Initial value:

```
{
    if (this != &counter_)
    {
        this->data = std::move(counter_.data);

        this->count_fun = std::move(counter_.count_fun);
        this->init_fun = std::move(counter_.init_fun);
        this->hasher_fun = std::move(counter_.hasher_fun);

        this->name = std::move(counter_.name);
        this->desc = std::move(counter_.desc);
    }
    return *this
```

Definition at line 26 of file counters-meat.hpp.

9.26.3.15 res

```
return res
```

Definition at line 265 of file counters-meat.hpp.

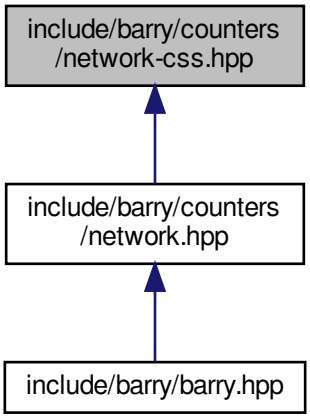
9.26.3.16 return

```
return
```

Definition at line 175 of file counters-meat.hpp.

9.27 include/barry/counters/network-css.hpp File Reference

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



Macros

- #define CSS_SIZE()
- #define CSS_MATCH_TYPE()
- #define CSS_CASE_TRUTH() if ((i < n) && (j < n))
- #define CSS_TRUE_CELLS()
- #define CSS_CASE_PERCEIVED() else if (((i >= s) && (i < e)) & ((j >= s) && (j < e)))
- #define CSS_PERCEIVED_CELLS()
- #define CSS_CASE_ELSE()
- #define CSS_CHECK_SIZE_INIT()
- #define CSS_CHECK_SIZE()
- #define CSS_APPEND(name)
- #define CSS_NET_COUNTER_LAMBDA_INIT()

Functions

- template<typename Tnet = Network>
  void counter_css_partially_false_recip_commi (NetCounters< Tnet > *counters, size_t netsize, const std::vector< size_t > &end_, size_t counter_type=0u)

  Counts errors of commission.
- template<typename Tnet = Network>
  void counter_css_partially_false_recip_omiss (NetCounters< Tnet > *counters, size_t netsize, const std::vector< size_t > &end_, size_t counter_type=0u)

  Counts errors of omission.
- template<typename Tnet = Network>
  void counter_css_completely_false_recip_comiss (NetCounters< Tnet > *counters, size_t netsize, const std::vector< size_t > &end_, size_t counter_type=0u)

  Counts completely false reciprocity (comission)
- template<typename Tnet = Network>
  void counter_css_completely_false_recip_omiss (NetCounters< Tnet > *counters, size_t netsize, const std::vector< size_t > &end_, size_t counter_type=0u)

  Counts completely false reciprocity (omission)
- template<typename Tnet = Network>
  void counter_css_mixed_recip (NetCounters< Tnet > *counters, size_t netsize, const std::vector< size_t > &end_, size_t counter_type=0u)

  Counts mixed reciprocity errors.
- template<typename Tnet = Network>
  void counter_css_census01 (NetCounters< Tnet > *counters, size_t netsize, const std::vector< size_t > &end_, size_t counter_type=0u)
- template<typename Tnet = Network>
  void counter_css_census02 (NetCounters< Tnet > *counters, size_t netsize, const std::vector< size_t > &end_, size_t counter_type=0u)
- template<typename Tnet = Network>
  void counter_css_census03 (NetCounters< Tnet > *counters, size_t netsize, const std::vector< size_t > &end_, size_t counter_type=0u)
- template<typename Tnet = Network>
  void counter_css_census04 (NetCounters< Tnet > *counters, size_t netsize, const std::vector< size_t > &end_, size_t counter_type=0u)
- template<typename Tnet = Network>
  void counter_css_census05 (NetCounters< Tnet > *counters, size_t netsize, const std::vector< size_t > &end_, size_t counter_type=0u)
- template<typename Tnet = Network>
  void counter_css_census06 (NetCounters< Tnet > *counters, size_t netsize, const std::vector< size_t > &end_, size_t counter_type=0u)
- template<typename Tnet = Network>
  void counter_css_census07 (NetCounters< Tnet > *counters, size_t netsize, const std::vector< size_t > &end_, size_t counter_type=0u)
- template<typename Tnet = Network>
  void counter_css_census08 (NetCounters< Tnet > *counters, size_t netsize, const std::vector< size_t > &end_, size_t counter_type=0u)
- template<typename Tnet = Network>
  void counter_css_census09 (NetCounters< Tnet > *counters, size_t netsize, const std::vector< size_t > &end_, size_t counter_type=0u)
- template<typename Tnet = Network>
  void counter_css_census10 (NetCounters< Tnet > *counters, size_t netsize, const std::vector< size_t > &end_, size_t counter_type=0u)

9.27.1 Macro Definition Documentation

9.27.1.1 CSS_APPEND

```
#define CSS_APPEND(
               name )
```

Value:

```
    std::string name_ = (name);\
    for (size_t i = 0u; i < end_.size(); ++i) { \
    std::string tmpname = name_ + " (" + std::to_string(i) + ")" + \
    ((counter_type == 1u) ? " (only perceiver)" : ((counter_type == 2u)? " (only alters)": ""));\
    counters->add_counter(tmp_count, tmp_init, nullptr, \
            NetCounterData({netsize, i == 0u ? netsize : end_[i-1], end_[i], counter_type, i}, {}),\
            tmpname);}
```

Definition at line 81 of file network-css.hpp.

9.27.1.2 CSS_CASE_ELSE

```
#define CSS_CASE_ELSE( )
```

Definition at line 66 of file network-css.hpp.

9.27.1.3 CSS_CASE_PERCEIVED

```
#define CSS_CASE_PERCEIVED( ) else if (((i >= s) && (i < e)) & ((j >= s) && (j < e)))
```

Definition at line 48 of file network-css.hpp.

9.27.1.4 CSS_CASE_TRUTH

```
#define CSS_CASE_TRUTH( ) if ((i < n) && (j < n))
```

Definition at line 32 of file network-css.hpp.

9.27.1.5 CSS_CHECK_SIZE

```
#define CSS_CHECK_SIZE( )
```

Value:

```
    for (size_t i = 0u; i < end_.size(); ++i) {\
    if (i == 0u) continue; \
    else if (end_[i] < end_[i-1u]) \
        throw std::logic_error("Endpoints should be specified in order.");}
```

Definition at line 76 of file network-css.hpp.

9.27.1.6 CSS_CHECK_SIZE_INIT

```
#define CSS_CHECK_SIZE_INIT( )
```

Value:

```
    /* The indices fall within the network */ \
    if ((data.indices.at(0) > Array.ncol()) \
    | (data.indices.at(2) > Array.ncol())) \
        throw std::range_error("The network does not match the prescribed size.");
```

Definition at line 70 of file network-css.hpp.

9.27.1.7 CSS_MATCH_TYPE

```
#define CSS_MATCH_TYPE( )
```

Value:

```
    if (ctype != 0u) { /* all counts */ \
        if (ctype == 1u) { /* Only if perceiver */ \
            if ((i_ != ego_id) && (j_ != ego_id)) return 0.0; \
        } else if (ctype == 2u) { /* Only if not perceiver */ \
            if ((i_ == ego_id) || (j_ == ego_id)) return 0.0; \
        } \
    };
```

Definition at line 22 of file network-css.hpp.

9.27.1.8 CSS_NET_COUNTER_LAMBDA_INIT

```
#define CSS_NET_COUNTER_LAMBDA_INIT( )
```

Value:

```
        NETWORK_COUNTER_LAMBDA(tmp_init) {\
        CSS_CHECK_SIZE_INIT() \
        return 0.0; \
    };
```

Definition at line 89 of file network-css.hpp.

9.27.1.9 CSS_PERCEIVED_CELLS

```
#define CSS_PERCEIVED_CELLS( )
```

Value:

```
    size_t i_ = i - s; \
    size_t j_ = j - s; \
    CSS_MATCH_TYPE() \
    double tji = static_cast<double>(Array(j - s, i - s, false)); \
    double pji = static_cast<double>(Array(j, i, false)); \
    double tij = static_cast<double>(Array(i - s, j - s, false));
```

Definition at line 55 of file network-css.hpp.

9.27.1.10 CSS_SIZE

```
#define CSS_SIZE( )
```

Value:

```
    size_t n      = data.indices[0u]; \
    size_t s      = data.indices[1u]; \
    size_t e      = data.indices[2u]; \
    size_t ctype  = data.indices[3u]; \
    size_t ego_id = data.indices[4u]; \
    if (ctype > 2) \
        throw std::range_error("Counter type should be 0, 1, or 2.");
```

Definition at line 8 of file network-css.hpp.

9.27.1.11 CSS_TRUE_CELLS

```
#define CSS_TRUE_CELLS( )
```

Value:

```
    size_t i_ = i; \
    size_t j_ = j; \
    CSS_MATCH_TYPE() \
    double tji = static_cast<double>(Array(j, i, false)); \
    double pij = static_cast<double>(Array(i + s, j + s, false)); \
    double pji = static_cast<double>(Array(j + s, i + s, false));
```

Definition at line 39 of file network-css.hpp.

9.27.2 Function Documentation

9.27.2.1 counter_css_census01()

```
template<typename Tnet = Network>
void counter_css_census01 (
            NetCounters< Tnet > * counters,
            size_t netsize,
            const std::vector< size_t > & end_,
            size_t counter_type = 0u )  [inline]
```

Definition at line 324 of file network-css.hpp.

9.27.2.2 counter_css_census02()

```
template<typename Tnet = Network>
void counter_css_census02 (
            NetCounters< Tnet > * counters,
            size_t netsize,
            const std::vector< size_t > & end_,
            size_t counter_type = 0u )  [inline]
```

Definition at line 389 of file network-css.hpp.

9.27.2.3 counter_css_census03()

```
template<typename Tnet = Network>
void counter_css_census03 (
             NetCounters< Tnet > * counters,
             size_t netsize,
             const std::vector< size_t > & end_,
             size_t counter_type = 0u )  [inline]
```

Definition at line 429 of file network-css.hpp.

9.27.2.4 counter_css_census04()

```
template<typename Tnet = Network>
void counter_css_census04 (
             NetCounters< Tnet > * counters,
             size_t netsize,
             const std::vector< size_t > & end_,
             size_t counter_type = 0u )  [inline]
```

Definition at line 469 of file network-css.hpp.

9.27.2.5 counter_css_census05()

```
template<typename Tnet = Network>
void counter_css_census05 (
             NetCounters< Tnet > * counters,
             size_t netsize,
             const std::vector< size_t > & end_,
             size_t counter_type = 0u )  [inline]
```

Definition at line 509 of file network-css.hpp.

9.27.2.6 counter_css_census06()

```
template<typename Tnet = Network>
void counter_css_census06 (
             NetCounters< Tnet > * counters,
             size_t netsize,
             const std::vector< size_t > & end_,
             size_t counter_type = 0u )  [inline]
```

Definition at line 549 of file network-css.hpp.

9.27.2.7 counter_css_census07()

```
template<typename Tnet = Network>
void counter_css_census07 (
            NetCounters< Tnet > * counters,
            size_t netsize,
            const std::vector< size_t > & end_,
            size_t counter_type = 0u )  [inline]
```

Definition at line 589 of file network-css.hpp.

9.27.2.8 counter_css_census08()

```
template<typename Tnet = Network>
void counter_css_census08 (
            NetCounters< Tnet > * counters,
            size_t netsize,
            const std::vector< size_t > & end_,
            size_t counter_type = 0u )  [inline]
```

Definition at line 629 of file network-css.hpp.

9.27.2.9 counter_css_census09()

```
template<typename Tnet = Network>
void counter_css_census09 (
            NetCounters< Tnet > * counters,
            size_t netsize,
            const std::vector< size_t > & end_,
            size_t counter_type = 0u )  [inline]
```

Definition at line 669 of file network-css.hpp.

9.27.2.10 counter_css_census10()

```
template<typename Tnet = Network>
void counter_css_census10 (
            NetCounters< Tnet > * counters,
            size_t netsize,
            const std::vector< size_t > & end_,
            size_t counter_type = 0u )  [inline]
```

Definition at line 709 of file network-css.hpp.

9.27.2.11 counter_css_completely_false_recip_comiss()

```
template<typename Tnet = Network>
void counter_css_completely_false_recip_comiss (
            NetCounters< Tnet > * counters,
            size_t netsize,
            const std::vector< size_t > & end_,
            size_t counter_type = 0u )  [inline]
```

Counts completely false reciprocity (comission)

Definition at line 200 of file network-css.hpp.

9.27.2.12 counter_css_completely_false_recip_omiss()

```
template<typename Tnet = Network>
void counter_css_completely_false_recip_omiss (
            NetCounters< Tnet > * counters,
            size_t netsize,
            const std::vector< size_t > & end_,
            size_t counter_type = 0u )  [inline]
```

Counts completely false reciprocity (omission)

Definition at line 241 of file network-css.hpp.

9.27.2.13 counter_css_mixed_recip()

```
template<typename Tnet = Network>
void counter_css_mixed_recip (
            NetCounters< Tnet > * counters,
            size_t netsize,
            const std::vector< size_t > & end_,
            size_t counter_type = 0u )  [inline]
```

Counts mixed reciprocity errors.

Definition at line 282 of file network-css.hpp.

9.27.2.14 counter_css_partially_false_recip_commi()

```
template<typename Tnet = Network>
void counter_css_partially_false_recip_commi (
            NetCounters< Tnet > * counters,
            size_t netsize,
            const std::vector< size_t > & end_,
            size_t counter_type = 0u )  [inline]
```

Counts errors of commission.

Parameters

netsize	Size of the reference (true) network
*end_*	Vector indicating one past the ending index of each network. (see details)
counter_type	Size_t indicating the type of counter to use. Possible values are: 0: Count all, 1: Only count if perceiver is involved, and 2: Only count if perceiver is not involved.

The `end_` parameter should be of length `N of networks` - 1. It is assumed that the first network ends at `netsize`.

Definition at line 107 of file network-css.hpp.

9.27.2.15 counter_css_partially_false_recip_omiss()

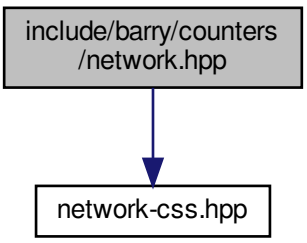
```
template<typename Tnet = Network>
void counter_css_partially_false_recip_omiss (
             NetCounters< Tnet > * counters,
             size_t netsize,
             const std::vector< size_t > & end_,
             size_t counter_type = 0u )  [inline]
```

Counts errors of omission.

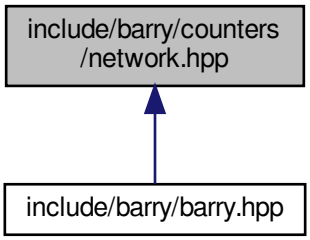
Definition at line 155 of file network-css.hpp.

9.28 include/barry/counters/network.hpp File Reference

`#include "network-css.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 size_t 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)
- #define NETWORKDENSE_COUNTER_LAMBDA(a)

Macros for defining rules

- #define NETWORK_RULE(a)
- #define NETWORK_RULE_LAMBDA(a)

Functions

- template<typename Tnet = Network>
  void counter_edges (NetCounters< Tnet > *counters)

  Number of edges.
- template<typename Tnet = Network>
  void counter_isolates (NetCounters< Tnet > *counters)

  Number of isolated vertices.
- template<> void counter_isolates (NetCounters< NetworkDense > *counters)
- template<typename Tnet = Network>
  void counter_mutual (NetCounters< Tnet > *counters)

  Number of mutual ties.
- template<typename Tnet = Network>
  void counter_istar2 (NetCounters< Tnet > *counters)
- template<> void counter_istar2 (NetCounters< NetworkDense > *counters)
- template<typename Tnet = Network>
  void counter_ostar2 (NetCounters< Tnet > *counters)
- template<> void counter_ostar2 (NetCounters< NetworkDense > *counters)
- template<typename Tnet = Network>
  void counter_ttriads (NetCounters< Tnet > *counters)
- template<> void counter_ttriads (NetCounters< NetworkDense > *counters)
- template<typename Tnet = Network>
  void counter_ctriads (NetCounters< Tnet > *counters)
- template<> void counter_ctriads (NetCounters< NetworkDense > *counters)
- template<typename Tnet = Network>
  void counter_density (NetCounters< Tnet > *counters)
- template<typename Tnet = Network>
  void counter_idegree15 (NetCounters< Tnet > *counters)
- template<> void counter_idegree15 (NetCounters< NetworkDense > *counters)
- template<typename Tnet = Network>
  void counter_odegree15 (NetCounters< Tnet > *counters)
- template<> void counter_odegree15 (NetCounters< NetworkDense > *counters)
- template<typename Tnet = Network>
  void counter_absdiff (NetCounters< Tnet > *counters, size_t attr_id, double alpha=1.0)

  Sum of absolute attribute difference between ego and alter.
- template<typename Tnet = Network>
  void counter_diff (NetCounters< Tnet > *counters, size_t 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)
- template<typename Tnet = Network>
  void counter_nodeicov (NetCounters< Tnet > *counters, size_t attr_id)
- template<typename Tnet = Network>
  void counter_nodeocov (NetCounters< Tnet > *counters, size_t attr_id)
- template<typename Tnet = Network>
  void counter_nodecov (NetCounters< Tnet > *counters, size_t attr_id)
- template<typename Tnet = Network>
  void counter_nodematch (NetCounters< Tnet > *counters, size_t attr_id)
- template<typename Tnet = Network>
  void counter_idegree (NetCounters< Tnet > *counters, std::vector< size_t > d)

  Counts number of vertices with a given in-degree.
- template<> void counter_idegree (NetCounters< NetworkDense > *counters, std::vector< size_t > d)
- template<typename Tnet = Network>
  void counter_odegree (NetCounters< Tnet > *counters, std::vector< size_t > d)

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

- template<typename Tnet = Network>
  void counter_degree (NetCounters< Tnet > *counters, std::vector< size_t > 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>`*).*

- template<typename Tnet = Network>
  void rules_zerodiag (NetRules< Tnet > *rules)

  Number of edges.

Convenient typedefs for network objects.

- #define BARRY_ZERO_NETWORK 0.0
- #define BARRY_ZERO_NETWORK_DENSE 0
- typedef BArray< double, NetworkData > Network
- typedef BArrayDense< int, NetworkData > NetworkDense
- template<typename Tnet = Network>
  using NetCounter = Counter< Tnet, NetCounterData >
- template<typename Tnet = Network>
  using NetCounters = Counters< Tnet, NetCounterData >
- template<typename Tnet = Network>
  using NetSupport = Support< Tnet, NetCounterData >
- template<typename Tnet = Network>
  using NetStatsCounter = StatsCounter< Tnet, NetCounterData >
- template<typename Tnet >
  using NetModel = Model< Tnet, NetCounterData >
- template<typename Tnet = Network>
  using NetRule = Rule< Tnet, bool >
- template<typename Tnet = Network>
  using NetRules = Rules< Tnet, bool >

9.28.1 Macro Definition Documentation

9.28.1.1 BARRY_ZERO_NETWORK

```
#define BARRY_ZERO_NETWORK 0.0
```

Definition at line 85 of file network.hpp.

9.28.1.2 BARRY_ZERO_NETWORK_DENSE

```
#define BARRY_ZERO_NETWORK_DENSE 0
```

Definition at line 86 of file network.hpp.

9.28.1.3 NET_C_DATA_IDX

```
#define NET_C_DATA_IDX(
              i ) (data.indices[i])
```

Definition at line 74 of file network.hpp.

9.28.1.4 NET_C_DATA_NUM

```
#define NET_C_DATA_NUM(
              i ) (data.numbers[i])
```

Definition at line 75 of file network.hpp.

9.28.1.5 NETWORK_COUNTER

```
#define NETWORK_COUNTER(
              a )
```

Value:

```
template<typename Tnet = Network>\
inline double (a) (const Tnet & Array, size_t i, size_t j, NetCounterData & data)
```

Function for definition of a network counter function

Definition at line 114 of file network.hpp.

9.28.1.6 NETWORK_COUNTER_LAMBDA

```
#define NETWORK_COUNTER_LAMBDA(
              a )
```

Value:

```
Counter_fun_type<Tnet, NetCounterData> a = \
    [](const Tnet & Array, size_t i, size_t j, NetCounterData & data)
```

Lambda function for definition of a network counter function

Definition at line 119 of file network.hpp.

9.28.1.7 NETWORK_RULE

```
#define NETWORK_RULE(
                a )
```

Value:
```
template<typename Tnet = Network>\
inline bool (a) (const Tnet & Array, size_t i, size_t j, bool & data)
```

Function for definition of a network counter function

Definition at line 133 of file network.hpp.

9.28.1.8 NETWORK_RULE_LAMBDA

```
#define NETWORK_RULE_LAMBDA(
                a )
```

Value:
```
Rule_fun_type<Tnet, bool> a = \
[](const Tnet & Array, size_t i, size_t j, bool & data)
```

Lambda function for definition of a network counter function

Definition at line 138 of file network.hpp.

9.28.1.9 NETWORKDENSE_COUNTER_LAMBDA

```
#define NETWORKDENSE_COUNTER_LAMBDA(
                a )
```

Value:
```
Counter_fun_type<NetworkDense, NetCounterData> a = \
    [](const NetworkDense & Array, size_t i, size_t j, NetCounterData & data)
```

Definition at line 123 of file network.hpp.

9.28.2 Typedef Documentation

9.28.2.1 NetCounter

```
template<typename Tnet = Network>
using NetCounter = Counter<Tnet, NetCounterData >
```

Definition at line 89 of file network.hpp.

9.28.2.2 NetCounters

```
template<typename Tnet = Network>
using NetCounters = Counters<Tnet, NetCounterData>
```

Definition at line 92 of file network.hpp.

9.28.2.3 NetModel

```
template<typename Tnet >
using NetModel = Model<Tnet, NetCounterData>
```

Definition at line 101 of file network.hpp.

9.28.2.4 NetRule

```
template<typename Tnet = Network>
using NetRule = Rule<Tnet, bool>
```

Definition at line 104 of file network.hpp.

9.28.2.5 NetRules

```
template<typename Tnet = Network>
using NetRules = Rules<Tnet, bool>
```

Definition at line 107 of file network.hpp.

9.28.2.6 NetStatsCounter

```
template<typename Tnet = Network>
using NetStatsCounter = StatsCounter<Tnet, NetCounterData>
```

Definition at line 98 of file network.hpp.

9.28.2.7 NetSupport

```
template<typename Tnet = Network>
using NetSupport = Support<Tnet, NetCounterData >
```

Definition at line 95 of file network.hpp.

9.28.2.8 Network

```
typedef BArray<double, NetworkData> Network
```

Definition at line 82 of file network.hpp.

9.28.2.9 NetworkDense

```
typedef BArrayDense<int, NetworkData> NetworkDense
```

Definition at line 83 of file network.hpp.

9.28.3 Function Documentation

9.28.3.1 rules_zerodiag()

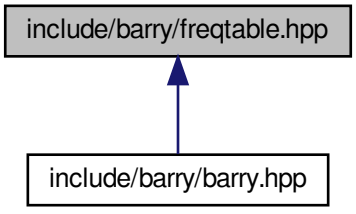
```
template<typename Tnet = Network>
void rules_zerodiag (
            NetRules< Tnet > * rules )  [inline]
```

Number of edges.

Definition at line 1381 of file network.hpp.

9.29 include/barry/freqtable.hpp File Reference

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



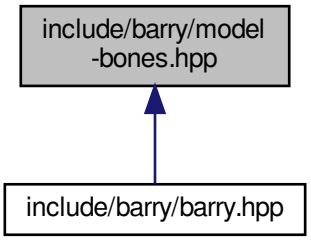
Classes

- class FreqTable< T >

  Frequency table of vectors.

9.30 include/barry/model-bones.hpp File Reference

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



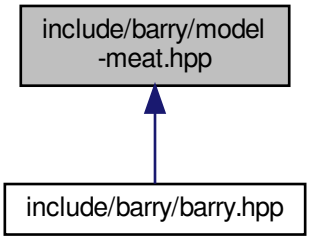
Classes

- 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:

9.31 include/barry/model-meat.hpp File Reference

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



Functions

- double update_normalizing_constant (const std::vector< double > &params, const double *support, size_t k, size_t n)
- double likelihood_ (const double *stats_target, const std::vector< double > &params, const double normalizing_constant, size_t n_params, bool log_=false)

9.31.1 Function Documentation

9.31.1.1 likelihood_()

```
double likelihood_ (
            const double * stats_target,
            const std::vector< double > & params,
            const double normalizing_constant,
            size_t n_params,
            bool log_ = false )  [inline]
```

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

9.31.1.2 update_normalizing_constant()

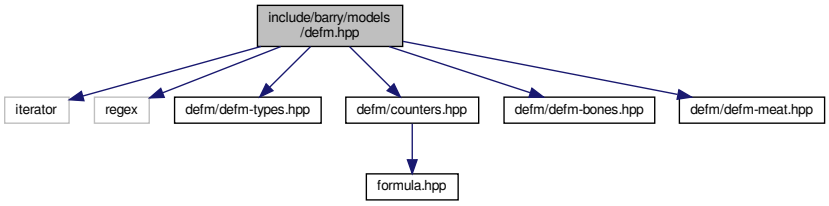
```
double update_normalizing_constant (
            const std::vector< double > & params,
            const double * support,
            size_t k,
            size_t n )  [inline]
```

Definition at line 9 of file model-meat.hpp.

9.32 include/barry/models/defm.hpp File Reference

```
#include <iterator>
#include <regex>
#include "defm/defm-types.hpp"
#include "defm/counters.hpp"
#include "defm/defm-bones.hpp"
#include "defm/defm-meat.hpp"
```

Include dependency graph for defm.hpp:



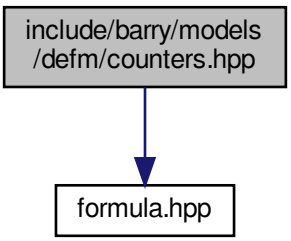
Namespaces

- defm

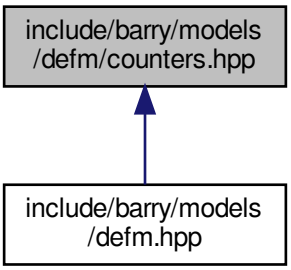
9.33 include/barry/models/defm/counters.hpp File Reference

```
#include "formula.hpp"
```

Include dependency graph for counters.hpp:



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



Macros

- #define MAKE_DEFM_HASHER(hasher, a, cov)
- #define DEFM_RULEDYN_LAMBDA(a)
- #define UNI_SUB(a)

Macros for defining counters

- #define DEFM_COUNTER(a) inline double (a) (const DEFMArray & Array, size_t i, size_t j, DEFMCounterData & data)
- #define DEFM_COUNTER_LAMBDA(a)

Macros for defining rules

- #define DEFM_RULE(a) inline bool (a) (const DEFMArray & Array, size_t i, size_t j, bool & data)
- #define DEFM_RULE_LAMBDA(a)

Functions

- void counter_ones (DEFMCounters *counters, int covar_index=-1, std::string vname="", const std::vector< std::string > *x_names=nullptr)

  Prevalence of ones.
- void counter_logit_intercept (DEFMCounters *counters, size_t n_y, std::vector< size_t > which={}, int covar_index=-1, std::string vname="", const std::vector< std::string > *x_names=nullptr, const std::vector< std::string > *y_names=nullptr)
- void counter_transition (DEFMCounters *counters, std::vector< size_t > coords, std::vector< bool > signs, size_t m_order, size_t n_y, int covar_index=-1, std::string vname="", const std::vector< std::string > *x_↩ names=nullptr, const std::vector< std::string > *y_names=nullptr)

  Prevalence of ones.
- void counter_transition_formula (DEFMCounters *counters, std::string formula, size_t m_order, size_t n_y, int covar_index=-1, std::string vname="", const std::vector< std::string > *x_names=nullptr, const std::vector< std::string > *y_names=nullptr)

  Prevalence of ones.
- void counter_fixed_effect (DEFMCounters *counters, int covar_index, double k, std::string vname="", const std::vector< std::string > *x_names=nullptr)

  Prevalence of ones.

Returns true if the cell is free

Parameters

rules	*A pointer to a* `DEFMRules` *object (*`Rules<DEFMArray, bool>`*).*

- void rules_markov_fixed (DEFMRules *rules, size_t markov_order)

  Number of edges.
- void rules_dont_become_zero (DEFMSupport *support, std::vector< size_t > ids)

  Blocks switching a one to zero.

9.33.1 Macro Definition Documentation

9.33.1.1 DEFM_COUNTER

```
#define DEFM_COUNTER(
              a ) inline double (a) (const DEFMArray & Array, size_t i, size_t j, DEFMCounterData
& data)
```

Function for definition of a network counter function

Definition at line 39 of file counters.hpp.

9.33.1.2 DEFM_COUNTER_LAMBDA

```
#define DEFM_COUNTER_LAMBDA(
              a )
```

Value:

```
barry::Counter_fun_type<DEFMArray, DEFMCounterData> a = \
    [](const DEFMArray & Array, size_t i, size_t j, DEFMCounterData & data) -> double
```

Lambda function for definition of a network counter function

Definition at line 43 of file counters.hpp.

9.33.1.3 DEFM_RULE

```
#define DEFM_RULE(
              a ) inline bool (a) (const DEFMArray & Array, size_t i, size_t j, bool & data)
```

Function for definition of a network counter function

Definition at line 53 of file counters.hpp.

9.33.1.4 DEFM_RULE_LAMBDA

```
#define DEFM_RULE_LAMBDA(
              a )
```

Value:

```
barry::Rule_fun_type<DEFMArray, DEFMRuleData> a = \
[](const DEFMArray & Array, size_t i, size_t j, DEFMRuleData & data) -> bool
```

Lambda function for definition of a network counter function

Definition at line 57 of file counters.hpp.

9.33.1.5 DEFM_RULEDYN_LAMBDA

```
#define DEFM_RULEDYN_LAMBDA(
              a )
```

Value:

```
barry::Rule_fun_type<DEFMArray, DEFMRuleDynData> a = \
[](const DEFMArray & Array, size_t i, size_t j, DEFMRuleDynData & data) -> bool
```

Lambda function for definition of a network counter function

Definition at line 63 of file counters.hpp.

9.33.1.6 UNI_SUB

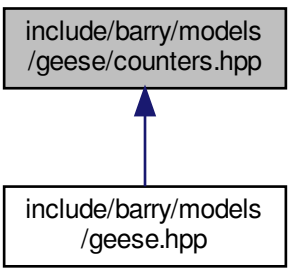
```
#define UNI_SUB(
              a )
```

Value:

```
(\
    ((a) == 0) ? u8"\u2080" : (\
    ((a) == 1) ? u8"\u2081" : (\
    ((a) == 2) ? u8"\u2082" : (\
    ((a) == 3) ? u8"\u2083" : (\
    ((a) == 4) ? u8"\u2084" : (\
    ((a) == 5) ? u8"\u2085" : (\
    ((a) == 6) ? u8"\u2086" : (\
    ((a) == 7) ? u8"\u2087" : (\
    ((a) == 8) ? u8"\u2088" : \
    u8"\u2089"))))))))\
)
```

9.34 include/barry/models/geese/counters.hpp File Reference

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



Functions

- void rule_dyn_limit_changes (PhyloSupport *support, size_t pos, size_t lb, size_t ub, size_t duplication=Geese::etype_default)

  Overall functional gains.

- #define MAKE_DUPL_VARS()
- #define IS_EITHER() (DATA_AT == Geese::etype_either)
- #define IS_DUPLICATION() ((DATA_AT == Geese::etype_duplication) & (DPL))
- #define IS_SPECIATION() ((DATA_AT == Geese::etype_speciation) & (!DPL))
- #define IF_MATCHES()
- #define IF_NOTMATCHES()
- #define PHYLO_COUNTER_LAMBDA(a)

  Extension of a simple counter.
- #define PHYLO_RULE_DYN_LAMBDA(a)
- #define PHYLO_CHECK_MISSING()
- std::string get_last_name (size_t d)
- void counter_overall_gains (PhyloCounters *counters, size_t duplication=Geese::etype_default)

  Overall functional gains.
- void counter_gains (PhyloCounters *counters, std::vector< size_t > nfun, size_t duplication=Geese::etype_default)

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

  k genes gain function nfun
- void counter_genes_changing (PhyloCounters *counters, size_t duplication=Geese::etype_default)

  Keeps track of how many genes are changing (either 0, 1, or 2 if dealing with regular trees.)
- void counter_preserve_pseudogene (PhyloCounters *counters, size_t nfunA, size_t nfunB, size_t duplication=Geese::etype_default)

  Keeps track of how many pairs of genes preserve pseudostate.
- void counter_prop_genes_changing (PhyloCounters *counters, size_t duplication=Geese::etype_default)

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

  Overall functional loss.
- void counter_maxfuns (PhyloCounters *counters, size_t lb, size_t ub, size_t duplication=Geese::etype_default)

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

  Total count of losses for an specific function.
- void counter_overall_changes (PhyloCounters *counters, size_t duplication=Geese::etype_default)

  Total number of changes. Use this statistic to account for "preservation".
- void counter_subfun (PhyloCounters *counters, size_t nfunA, size_t nfunB, size_t duplication=Geese::etype_default)

  Total count of Sub-functionalization events.
- void counter_cogain (PhyloCounters *counters, size_t nfunA, size_t nfunB, size_t duplication=Geese::etype_default)

  Co-evolution (joint gain or loss)
- void counter_longest (PhyloCounters *counters, size_t duplication=Geese::etype_default)

  Longest branch mutates (either by gain or by loss)
- void counter_neofun (PhyloCounters *counters, size_t nfunA, size_t nfunB, size_t duplication=Geese::etype_default)

  Total number of neofunctionalization events.
- void counter_pairwise_neofun_singlefun (PhyloCounters *counters, size_t nfunA, size_t duplication=Geese::etype_default)

  Total number of neofunctionalization events sum_u sum_{w < u} [x(u,a)(1 - x(w,a)) + (1 - x(u,a)) * x(w,a)] change stat: delta{x(u,a): 0->1} = 1 - 2 * x(w,a)*
- void counter_neofun_a2b (PhyloCounters *counters, size_t nfunA, size_t nfunB, size_t duplication=Geese::etype_default)

  Total number of neofunctionalization events.

- void counter_co_opt (PhyloCounters *counters, size_t nfunA, size_t nfunB, size_t duplication=Geese::etype_default)

  Function co-opting.
- void counter_k_genes_changing (PhyloCounters *counters, size_t k, size_t duplication=Geese::etype_default)

  Indicator function. Equals to one if k genes changed and zero otherwise.
- void counter_less_than_p_prop_genes_changing (PhyloCounters *counters, double p, size_t duplication=Geese::etype_default

  Indicator function. Equals to one if k genes changed and zero otherwise.
- void counter_gains_from_0 (PhyloCounters *counters, std::vector< size_t > nfun, size_t duplication=Geese::etype_default)

  Used when all the functions are in 0 (like the root node prob.)
- void counter_overall_gains_from_0 (PhyloCounters *counters, size_t duplication=Geese::etype_default)

  Used when all the functions are in 0 (like the root node prob.)
- void counter_pairwise_overall_change (PhyloCounters *counters, size_t duplication=Geese::etype_default)

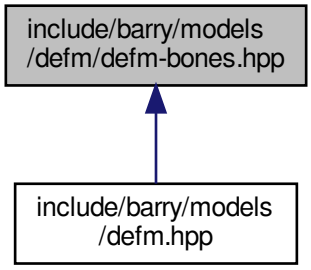
  Used when all the functions are in 0 (like the root node prob.)
- void counter_pairwise_preserving (PhyloCounters *counters, size_t nfunA, size_t nfunB, size_t duplication=Geese::etype_default)

  Used when all the functions are in 0 (like the root node prob.)
- void counter_pairwise_first_gain (PhyloCounters *counters, size_t nfunA, size_t nfunB, size_t duplication=Geese::etype_default)

  Used when all the functions are in 0 (like the root node prob.)

9.35 include/barry/models/defm/defm-bones.hpp File Reference

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

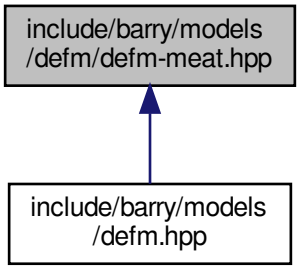


Classes

- class DEFM

9.36 include/barry/models/defm/defm-meat.hpp File Reference

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



Macros

- #define DEFM_RANGES(a)
- #define DEFM_LOOP_ARRAYS(a) for (size_t a = 0u; a < (nobs_i - M_order); ++a)

Functions

- std::vector< double > keygen_defm (const DEFMArray &Array_, DEFMCounterData *data)

9.36.1 Macro Definition Documentation

9.36.1.1 DEFM_LOOP_ARRAYS

```
#define DEFM_LOOP_ARRAYS(
              a )  for (size_t a = 0u; a < (nobs_i - M_order); ++a)
```

Definition at line 36 of file defm-meat.hpp.

9.36.1.2 DEFM_RANGES

```
#define DEFM_RANGES(
              a )
```

Value:

```
    size_t start_i = start_end[a * 2u];\
    size_t end_i   = start_end[a * 2u + 1u];\
    size_t nobs_i  = end_i - start_i + 1u;
```

Definition at line 31 of file defm-meat.hpp.

9.36.2 Function Documentation

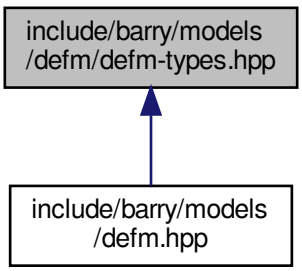
9.36.2.1 keygen_defm()

```
std::vector< double > keygen_defm (
            const DEFMArray & Array_,
            DEFMCounterData * data )  [inline]
```

Definition at line 4 of file defm-meat.hpp.

9.37 include/barry/models/defm/defm-types.hpp File Reference

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



Classes

- class DEFMData

  Data class for DEFM arrays.
- class DEFMCounterData

  Data class used to store arbitrary size_t or double vectors.
- class DEFMRuleData
- class DEFMRuleDynData

Typedefs

- typedef barry::BArrayDense< int, DEFMData > DEFMArray

Convenient typedefs for network objects.

- typedef barry::Counter< DEFMArray, DEFMCounterData > DEFMCounter
- typedef barry::Counters< DEFMArray, DEFMCounterData > DEFMCounters
- typedef barry::Support< DEFMArray, DEFMCounterData, DEFMRuleData, DEFMRuleDynData > DEFMSupport
- typedef barry::StatsCounter< DEFMArray, DEFMCounterData > DEFMStatsCounter
- typedef barry::Model< DEFMArray, DEFMCounterData, DEFMRuleData, DEFMRuleDynData > DEFMModel
- typedef barry::Rule< DEFMArray, DEFMRuleData > DEFMRule
- typedef barry::Rules< DEFMArray, DEFMRuleData > DEFMRules
- typedef barry::Rule< DEFMArray, DEFMRuleDynData > DEFMRuleDyn
- typedef barry::Rules< DEFMArray, DEFMRuleDynData > DEFMRulesDyn

9.37.1 Typedef Documentation

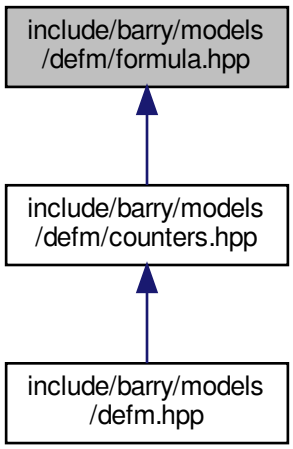
9.37.1.1 DEFMArray

```
typedef barry::BArrayDense<int, DEFMData> DEFMArray
```

Definition at line 5 of file defm-types.hpp.

9.38 include/barry/models/defm/formula.hpp File Reference

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



Functions

- void defm_motif_parser (std::string formula, std::vector< size_t > &locations, std::vector< bool > &signs, size_t m_order, size_t y_ncol)

  Parses a motif formula.

9.38.1 Function Documentation

9.38.1.1 defm_motif_parser()

```
void defm_motif_parser (
            std::string formula,
            std::vector< size_t > & locations,
            std::vector< bool > & signs,
            size_t m_order,
            size_t y_ncol )  [inline]
```

Parses a motif formula.

This function will take the formula and generate the corresponding input for defm::counter_transition(). Formulas can be specified in the following ways:

- Intercept effect: {...} No transition, only including the current state.
- Transition effect: {...} > {...} Includes current and previous states.

The general notation is `[0]y[column id]_[row id]`. A preceeding zero means that the value of the cell is considered to be zero. The column id goes between 0 and the number of columns in the array - 1 (so it is indexed from 0,) and the row id goes from 0 to m_order.

Intercept effects

Intercept effects only involve a single set of curly brackets. Using the 'greater-than' symbol (i.e., '<') is only for transition effects. When specifying intercept effects, users can skip the `row_id`, e.g., `y0_0` is equivalent to `y0`. If the passed `row id` is different from the Markov order, i.e., `row_id != m_order`, then the function returns with an error.

Examples:

- "`{y0, 0y1}`" is equivalent to set a motif with the first element equal to one and the second to zero.

Transition effects

Transition effects can be specified using two sets of curly brackets and an greater-than symbol, i.e., `{...} > {...}`. The first set of brackets, which we call LHS, can only hold `row id` that are less than `m_order`.

Parameters

formula	
locations	
signs	
m_order	
y_ncol	

Definition at line 46 of file formula.hpp.

9.39 include/barry/models/geese.hpp File Reference

```
#include "geese/geese-types.hpp"
#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/geese-meat-predict_exhaust.hpp"
#include "geese/geese-meat-predict_sim.hpp"
#include "geese/flock-bones.hpp"
#include "geese/flock-meat.hpp"
#include "geese/counters.hpp"
```

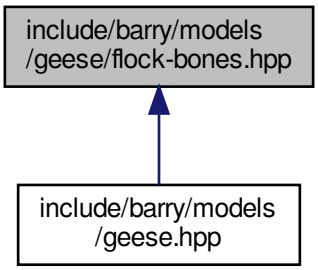
Include dependency graph for geese.hpp:

Namespaces

- geese

9.40 include/barry/models/geese/flock-bones.hpp File Reference

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



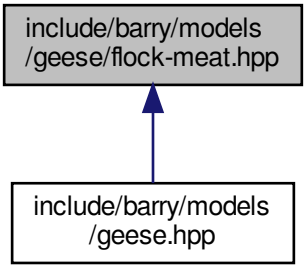
Classes

- class Flock

  A Flock is a group of Geese.

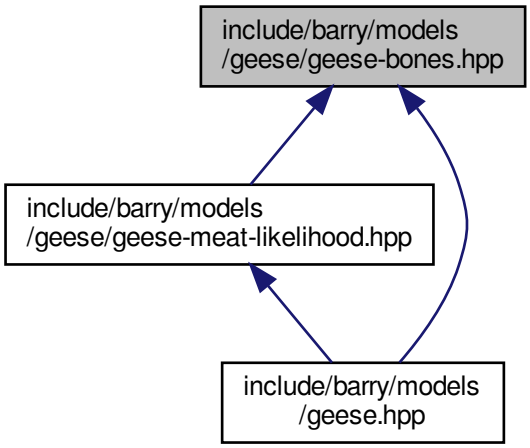
9.41 include/barry/models/geese/flock-meat.hpp File Reference

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



9.42 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 PhyloArray &array, const PhyloCounterData *d)
- bool vec_diff (const std::vector< size_t > &s, const std::vector< size_t > &a)

9.42.1 Macro Definition Documentation

9.42.1.1 INITIALIZED

```
#define INITIALIZED( )
```

Value:

```
    if (!this->initialized) \
    throw std::logic_error("The model has not been initialized yet.");
```

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

9.42.2 Function Documentation

9.42.2.1 keygen_full()

```
std::vector< double > keygen_full (
            const PhyloArray & array,
            const PhyloCounterData * d )  [inline]
```

Definition at line 36 of file geese-bones.hpp.

9.42.2.2 RULE_FUNCTION()

```
RULE_FUNCTION (
            rule_empty_free  )
```

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

9.42.2.3 vec_diff()

```
bool vec_diff (
            const std::vector< size_t > & s,
            const std::vector< size_t > & a )  [inline]
```

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

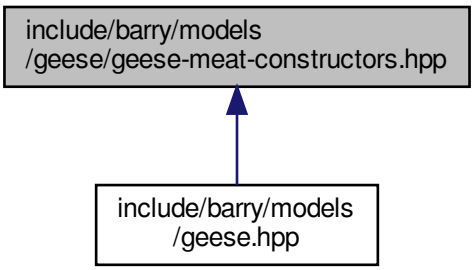
9.42.2.4 vector_caster()

```
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.

9.43 include/barry/models/geese/geese-meat-constructors.hpp File Reference

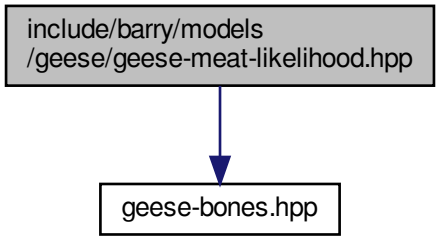
This graph shows which files directly or indirectly include this file:



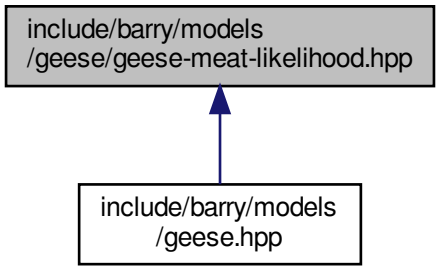
9.44 include/barry/models/geese/geese-meat-likelihood.hpp File Reference

```
#include "geese-bones.hpp"
```

Include dependency graph for geese-meat-likelihood.hpp:



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



Functions

- void pset_loop (size_t n, size_t s, size_t nfunctions, const size_t node_id, const size_t array_id, std::vector< double > &totprob_n, const std::vector< double > &par0, const std::vector< std::vector< bool >> &states, const std::vector< PhyloArray > &psets, const std::vector< double > &psets_stats, const std::vector< std::vector< size_t > > &locations, const std::vector< geese::Node * > &node_offspring, const std::vector< double > &psetprobs)

9.44.1 Function Documentation

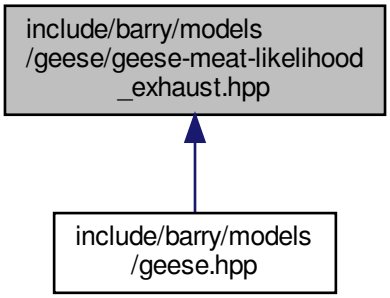
9.44.1.1 pset_loop()

```
void pset_loop (
             size_t n,
             size_t s,
             size_t nfunctions,
             const size_t node_id,
             const size_t array_id,
             std::vector< double > & totprob_n,
             const std::vector< double > & par0,
             const std::vector< std::vector< bool >> & states,
             const std::vector< PhyloArray > & psets,
             const std::vector< double > & psets_stats,
             const std::vector< std::vector< size_t > > & locations,
             const std::vector< geese::Node * > & node_offspring,
             const std::vector< double > & psetprobs )  [inline]
```

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

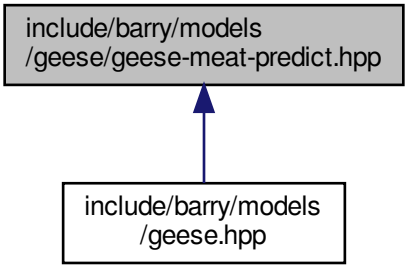
9.45 include/barry/models/geese/geese-meat-likelihood_exhaust.hpp File Reference

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



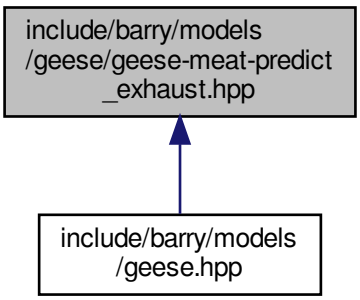
9.46 include/barry/models/geese/geese-meat-predict.hpp File Reference

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



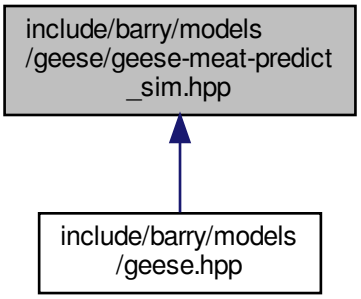
9.47 include/barry/models/geese/geese-meat-predict_exhaust.hpp File Reference

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



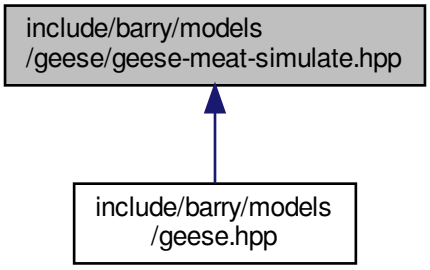
9.48 include/barry/models/geese/geese-meat-predict_sim.hpp File Reference

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



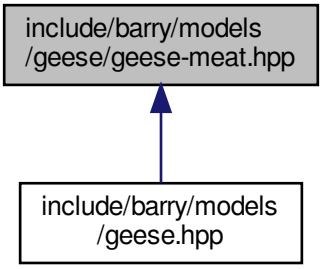
9.49 include/barry/models/geese/geese-meat-simulate.hpp File Reference

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



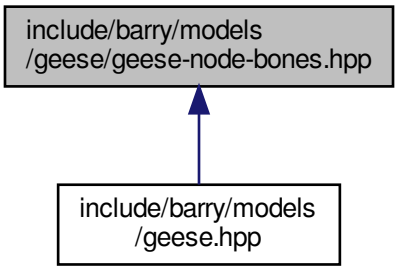
9.50 include/barry/models/geese/geese-meat.hpp File Reference

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



9.51 include/barry/models/geese/geese-node-bones.hpp File Reference

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



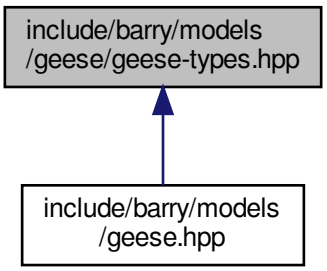
Classes

- class Node

  A single node for the model.

9.52 include/barry/models/geese/geese-types.hpp File Reference

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



Classes

- class NodeData

  Data definition for the `PhyloArray` class.
- class PhyloCounterData
- class PhyloRuleDynData

Macros

- #define POS(a, b) (b)∗N + (a)

Typedefs

Convenient typedefs for Node objects.

- typedef std::vector< std::pair< size_t, size_t > > PhyloRuleData
- typedef barry::BArrayDense< size_t, NodeData > PhyloArray
- typedef barry::Counter< PhyloArray, PhyloCounterData > PhyloCounter
- typedef barry::Counters< PhyloArray, PhyloCounterData > PhyloCounters
- typedef barry::Rule< PhyloArray, PhyloRuleData > PhyloRule
- typedef barry::Rules< PhyloArray, PhyloRuleData > PhyloRules
- typedef barry::Rule< PhyloArray, PhyloRuleDynData > PhyloRuleDyn
- typedef barry::Rules< PhyloArray, PhyloRuleDynData > PhyloRulesDyn
- typedef barry::Support< PhyloArray, PhyloCounterData, PhyloRuleData, PhyloRuleDynData > PhyloSupport
- typedef barry::StatsCounter< PhyloArray, PhyloCounterData > PhyloStatsCounter
- typedef barry::Model< PhyloArray, PhyloCounterData, PhyloRuleData, PhyloRuleDynData > PhyloModel
- typedef barry::PowerSet< PhyloArray, PhyloRuleData > PhyloPowerSet

9.52.1 Macro Definition Documentation

9.52.1.1 POS

```
#define POS(
              a,
              b ) (b)*N + (a)
```

Definition at line 4 of file geese-types.hpp.

9.52.2 Typedef Documentation

9.52.2.1 PhyloArray

```
typedef barry::BArrayDense<size_t, NodeData> PhyloArray
```

Definition at line 104 of file geese-types.hpp.

9.52.2.2 PhyloCounter

```
typedef barry::Counter<PhyloArray, PhyloCounterData > PhyloCounter
```

Definition at line 105 of file geese-types.hpp.

9.52.2.3 PhyloCounters

```
typedef barry::Counters< PhyloArray, PhyloCounterData> PhyloCounters
```

Definition at line 106 of file geese-types.hpp.

9.52.2.4 PhyloModel

```
typedef barry::Model<PhyloArray, PhyloCounterData, PhyloRuleData, PhyloRuleDynData > PhyloModel
```

Definition at line 116 of file geese-types.hpp.

9.52.2.5 PhyloPowerSet

```
typedef barry::PowerSet<PhyloArray, PhyloRuleData> PhyloPowerSet
```

Definition at line 117 of file geese-types.hpp.

9.52.2.6 PhyloRule

```
typedef barry::Rule<PhyloArray,PhyloRuleData> PhyloRule
```

Definition at line 108 of file geese-types.hpp.

9.52.2.7 PhyloRuleData

```
typedef std::vector< std::pair< size_t, size_t > > PhyloRuleData
```

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

9.52.2.8 PhyloRuleDyn

```
typedef barry::Rule<PhyloArray,PhyloRuleDynData> PhyloRuleDyn
```

Definition at line 111 of file geese-types.hpp.

9.52.2.9 PhyloRules

```
typedef barry::Rules<PhyloArray,PhyloRuleData> PhyloRules
```

Definition at line 109 of file geese-types.hpp.

9.52.2.10 PhyloRulesDyn

```
typedef barry::Rules<PhyloArray,PhyloRuleDynData> PhyloRulesDyn
```

Definition at line 112 of file geese-types.hpp.

9.52.2.11 PhyloStatsCounter

```
typedef barry::StatsCounter<PhyloArray, PhyloCounterData> PhyloStatsCounter
```

Definition at line 115 of file geese-types.hpp.

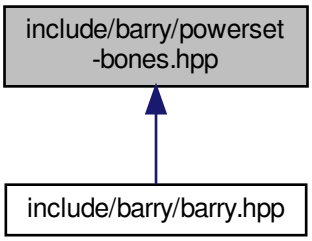
9.52.2.12 PhyloSupport

```
typedef barry::Support<PhyloArray, PhyloCounterData, PhyloRuleData, PhyloRuleDynData > PhyloSupport
```

Definition at line 114 of file geese-types.hpp.

9.53 include/barry/powerset-bones.hpp File Reference

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



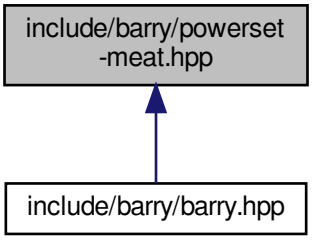
Classes

- class PowerSet< Array_Type, Data_Rule_Type >

  Powerset of a binary array.

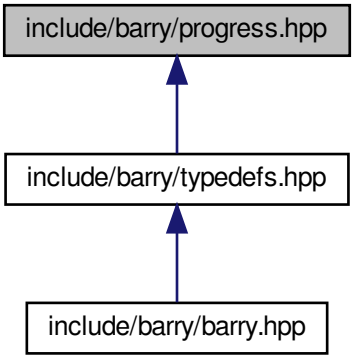
9.54 include/barry/powerset-meat.hpp File Reference

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



9.55 include/barry/progress.hpp File Reference

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



Classes

- class Progress

  A simple progress bar.

Macros

- #define BARRY_PROGRESS_BAR_WIDTH 80

9.55.1 Macro Definition Documentation

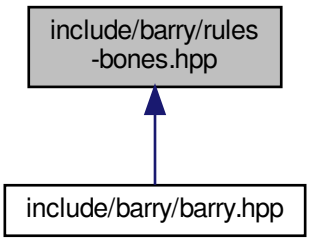
9.55.1.1 BARRY_PROGRESS_BAR_WIDTH

```
#define BARRY_PROGRESS_BAR_WIDTH 80
```

Definition at line 5 of file progress.hpp.

9.56 include/barry/rules-bones.hpp File Reference

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, size_t i, size_t j, Data_Type *dat)

9.56.1 Function Documentation

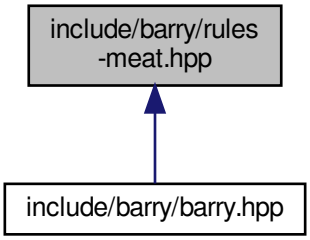
9.56.1.1 rule_fun_default()

```
template<typename Array_Type , typename Data_Type >
bool rule_fun_default (
             const Array_Type * array,
             size_t i,
             size_t j,
             Data_Type * dat )
```

Definition at line 5 of file rules-bones.hpp.

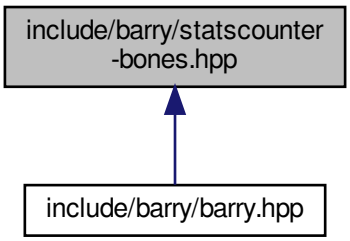
9.57 include/barry/rules-meat.hpp File Reference

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



9.58 include/barry/statscounter-bones.hpp File Reference

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



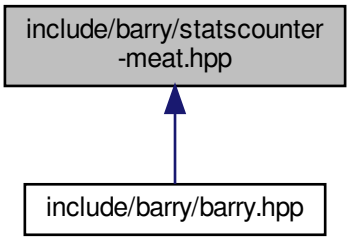
Classes

- class StatsCounter< Array_Type, Data_Type >

  Count stats for a single Array.

9.59 include/barry/statscounter-meat.hpp File Reference

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



Macros

- #define STATSCOUNTER_TYPE() StatsCounter<Array_Type,Data_Type>
- #define STATSCOUNTER_TEMPLATE_ARGS() <typename Array_Type, typename Data_Type>
- #define STATSCOUNTER_TEMPLATE(a, b) template STATSCOUNTER_TEMPLATE_ARGS() inline a STATSCOUNTER_TYPE()::b

Functions

- STATSCOUNTER_TEMPLATE (, StatsCounter)(const StatsCounter< Array_Type
- EmptyArray clear ()
- STATSCOUNTER_TEMPLATE (,~StatsCounter)()
- STATSCOUNTER_TEMPLATE (void, reset_array)(const Array_Type *Array_)
- STATSCOUNTER_TEMPLATE (void, add_counter)(Counter< Array_Type
- STATSCOUNTER_TEMPLATE (void, set_counters)(Counters< Array_Type
- STATSCOUNTER_TEMPLATE (void, count_init)(size_t i
- current_stats resize (counters->size(), 0.0)
- for (size_t n=0u;n< counters->size();++n) current_stats[n]
- STATSCOUNTER_TEMPLATE (void, count_current)(size_t i
- STATSCOUNTER_TEMPLATE (std::vector< std::string >, get_names)() const
- STATSCOUNTER_TEMPLATE (std::vector< std::string >, get_descriptions)() const

Variables

- Data_Type & counter
- EmptyArray = ∗Array
- current_stats = counter.current_stats
- counters = new Counters<Array_Type,Data_Type>((∗counter.counters))
- counter_deleted = false
- Data_Type f_
- return
- Data_Type ∗ counters_
- size_t j

9.59.1 Macro Definition Documentation

9.59.1.1 STATSCOUNTER_TEMPLATE

```
#define STATSCOUNTER_TEMPLATE(
              a,
              b )  template STATSCOUNTER_TEMPLATE_ARGS() inline a STATSCOUNTER_TYPE()::b
```

Definition at line 8 of file statscounter-meat.hpp.

9.59.1.2 STATSCOUNTER_TEMPLATE_ARGS

```
template STATSCOUNTER_TEMPLATE_ARGS( ) <typename Array_Type, typename Data_Type>
```

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

9.59.1.3 STATSCOUNTER_TYPE

```
template Data_Type * STATSCOUNTER_TYPE( ) StatsCounter<Array_Type,Data_Type>
```

Definition at line 4 of file statscounter-meat.hpp.

9.59.2 Function Documentation

9.59.2.1 clear()

```
EmptyArray clear ( )
```

9.59.2.2 for()

```
for (
            size_t n = 0u;n< counters->size();++n )
```

9.59.2.3 resize()

```
current_stats resize (
            counters-> size(),
            0.  0 )
```

9.59.2.4 STATSCOUNTER_TEMPLATE() [1/9]

```
STATSCOUNTER_TEMPLATE (
            StatsCounter  ) const
```

9.59.2.5 STATSCOUNTER_TEMPLATE() [2/9]

```
STATSCOUNTER_TEMPLATE (
            ~ StatsCounter )
```

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

9.59.2.6 STATSCOUNTER_TEMPLATE() [3/9]

```
STATSCOUNTER_TEMPLATE (
            std::vector< std::string > ,
            get_descriptions  ) const
```

Definition at line 256 of file statscounter-meat.hpp.

9.59.2.7 STATSCOUNTER_TEMPLATE() [4/9]

```
STATSCOUNTER_TEMPLATE (
            std::vector< std::string > ,
            get_names  ) const
```

Definition at line 251 of file statscounter-meat.hpp.

9.59.2.8 STATSCOUNTER_TEMPLATE() [5/9]

```
STATSCOUNTER_TEMPLATE (
            void ,
            add_counter  )
```

9.59.2.9 STATSCOUNTER_TEMPLATE() [6/9]

```
STATSCOUNTER_TEMPLATE (
            void ,
            count_current  )
```

9.59.2.10 STATSCOUNTER_TEMPLATE() [7/9]

```
STATSCOUNTER_TEMPLATE (
            void ,
            count_init  )
```

9.59.2.11 STATSCOUNTER_TEMPLATE() [8/9]

```
STATSCOUNTER_TEMPLATE (
            void ,
            reset_array  ) const
```

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

9.59.2.12 STATSCOUNTER_TEMPLATE() [9/9]

```
STATSCOUNTER_TEMPLATE (
            void ,
            set_counters  )
```

9.59.3 Variable Documentation

9.59.3.1 counter

```
Data_Type& counter
```

Initial value:

```
{
    Array      = counter.Array
```

Definition at line 12 of file statscounter-meat.hpp.

9.59.3.2 counter_deleted

```
counter_deleted = false
```

Definition at line 23 of file statscounter-meat.hpp.

9.59.3.3 counters

```
counters = new Counters<Array_Type,Data_Type>((*counter.counters))
```

Definition at line 22 of file statscounter-meat.hpp.

9.59.3.4 counters_

```
Data_Type* counters_
```

Initial value:

```
{

    if (!counter_deleted)
        delete counters
```

Definition at line 53 of file statscounter-meat.hpp.

9.59.3.5 current_stats

```
current_stats = counter.current_stats
```

Definition at line 19 of file statscounter-meat.hpp.

9.59.3.6 EmptyArray

```
EmptyArray = *Array
```

Definition at line 17 of file statscounter-meat.hpp.

9.59.3.7 f_

```
Data_Type f_
```

Initial value:

```
{

    counters->add_counter(f_)
```

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

9.59.3.8 j

```
size_t j
```

Initial value:

```
{


    if (counters->size() == 0u)
        throw std::logic_error("No counters added: Cannot count without knowning what to count!")
```

Definition at line 66 of file statscounter-meat.hpp.

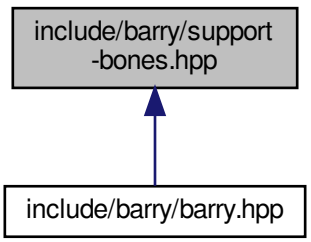
9.59.3.9 return

```
return
```

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

9.60 include/barry/support-bones.hpp File Reference

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



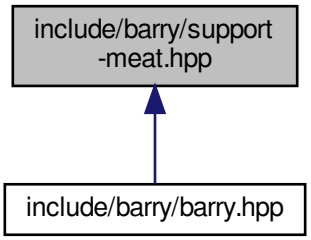
Classes

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

  Compute the support of sufficient statistics.

9.61 include/barry/support-meat.hpp File Reference

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



Macros

- #define BARRY_SUPPORT_MEAT_HPP 1

9.61.1 Macro Definition Documentation

9.61.1.1 BARRY_SUPPORT_MEAT_HPP

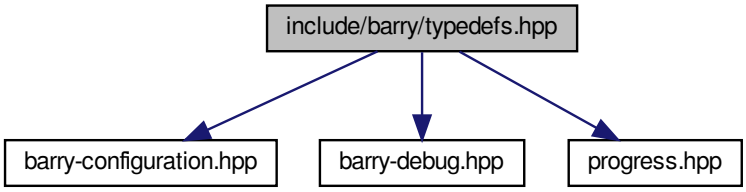
```
#define BARRY_SUPPORT_MEAT_HPP 1
```

Definition at line 2 of file support-meat.hpp.

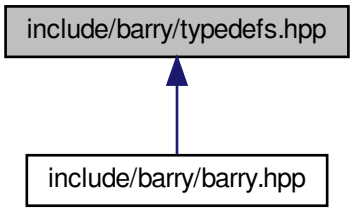
9.62 include/barry/typedefs.hpp File Reference

```
#include "barry-configuration.hpp"
#include "barry-debug.hpp"
#include "progress.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 std::vector< std::pair< std::vector< double >, size_t > > Counts_type
- template<typename Cell_Type >
  using Row_type = Map< size_t, Cell< Cell_Type > >
- template<typename Cell_Type >
  using Col_type = Map< size_t, Cell< Cell_Type > * >
- template<typename Ta = double, typename Tb = size_t>
  using MapVec_type = std::unordered_map< std::vector< Ta >, Tb, vecHasher< Ta > >
- template<typename Array_Type , typename Data_Type >
  using Hasher_fun_type = std::function< std::vector< double >(const Array_Type &, Data_Type *)>

  Hasher function used by the counter.

- template<typename Array_Type , typename Data_Type >
  using Counter_fun_type = std::function< double(const Array_Type &, size_t, size_t, Data_Type &)>

  Counter and rule functions.
- template<typename Array_Type , typename Data_Type >
  using Rule_fun_type = std::function< bool(const Array_Type &, size_t, size_t, Data_Type &)>

Functions

- std::vector< size_t > sort_array (const double *v, size_t start, size_t ncols, size_t nrows)

  Ascending sorting an array.
- template<typename T >
  T vec_inner_prod (const T *a, const T *b, size_t n)
- template<> double vec_inner_prod (const double *a, const double *b, size_t n)

- 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-100)

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

9.62.1 Typedef Documentation

9.62.1.1 Col_type

```
template<typename Cell_Type >
using Col_type = Map< size_t, Cell<Cell_Type>* >
```

Definition at line 70 of file typedefs.hpp.

9.62.1.2 Counter_fun_type

```
template<typename Array_Type , typename Data_Type >
using Counter_fun_type = std::function<double(const Array_Type &, size_t, size_t, Data_Type
&)>
```

Counter and rule functions.

Parameters

Array_Type	a BArray
unit,size_t	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 187 of file typedefs.hpp.

9.62.1.3 Counts_type

```
typedef std::vector< std::pair< std::vector<double>, size_t > > Counts_type
```

Definition at line 51 of file typedefs.hpp.

9.62.1.4 Hasher_fun_type

```
template<typename Array_Type , typename Data_Type >
using Hasher_fun_type = std::function<std::vector<double>(const Array_Type &, Data_Type *)>
```

Hasher function used by the counter.

Used to characterize the support of the array.

Template Parameters

Array_Type	

Definition at line 200 of file typedefs.hpp.

9.62.1.5 MapVec_type

```
template<typename Ta = double, typename Tb = size_t>
using MapVec_type = std::unordered_map< std::vector< Ta >, Tb, vecHasher<Ta> >
```

Definition at line 128 of file typedefs.hpp.

9.62.1.6 Row_type

```
template<typename Cell_Type >
using Row_type = Map< size_t, Cell<Cell_Type> >
```

Definition at line 67 of file typedefs.hpp.

9.62.1.7 Rule_fun_type

```
template<typename Array_Type , typename Data_Type >
using Rule_fun_type = std::function<bool(const Array_Type &, size_t, size_t, Data_Type &)>
```

Definition at line 190 of file typedefs.hpp.

9.62.2 Function Documentation

9.62.2.1 sort_array()

```
std::vector< size_t > sort_array (
            const double * v,
            size_t start,
            size_t ncols,
            size_t nrows )  [inline]
```

Ascending sorting an array.

It will sort an array solving ties using the next column. Data is stored column-wise.

Template Parameters

T	

Parameters

v	
nrows	

Returns

std::vector<size_t> The sorting index.

Definition at line 141 of file typedefs.hpp.

9.62.2.2 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 210 of file typedefs.hpp.

9.62.2.3 vec_equal_approx()

```
template<typename T >
bool vec_equal_approx (
            const std::vector< T > & a,
            const std::vector< T > & b,
            double eps = 1e-100 )  [inline]
```

Definition at line 235 of file typedefs.hpp.

9.62.2.4 vec_inner_prod() [1/2]

```
template<>
double vec_inner_prod (
             const double * a,
             const double * b,
             size_t n )  [inline]
```

Definition at line 286 of file typedefs.hpp.

9.62.2.5 vec_inner_prod() [2/2]

```
template<typename T >
T vec_inner_prod (
             const T * a,
             const T * b,
             size_t n )  [inline]
```

Definition at line 263 of file typedefs.hpp.

9.63 README.md File Reference

Index

~BArray
BArray< Cell_Type, Data_Type >, 63
~BArrayCell
BArrayCell< Cell_Type, Data_Type >, 74
~BArrayCell_const
BArrayCell_const< Cell_Type, Data_Type >, 77
~BArrayDense
BArrayDense< Cell_Type, Data_Type >, 83
~BArrayDenseCell
BArrayDenseCell< Cell_Type, Data_Type >, 97
~BArrayRow
BArrayRow< Cell_Type, Data_Type >, 110
~BArrayRow_const
BArrayRow_const< Cell_Type, Data_Type >, 112
~BArrayVector
BArrayVector< Cell_Type, Data_Type >, 115
~BArrayVector_const
BArrayVector_const< Cell_Type, Data_Type >, 118
~Cell
Cell< Cell_Type >, 122
~ConstBArrayRowIter
ConstBArrayRowIter< Cell_Type, Data_Type >, 127
~Counter
Counter< Array_Type, Data_Type >, 130
~Counters
Counters< Array_Type, Data_Type >, 135
~DEFMCounterData
Phylo rules, 43
~DEFMData
Phylo rules, 43
~DEFMRuleDynData
Phylo rules, 43
~Entries
Entries< Cell_Type >, 149
~Flock
Flock, 151
~FreqTable
FreqTable< T >, 158
~Geese
Geese, 164
~Model
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 180
~NetCounterData
NetCounterData, 201
~NetworkData
NetworkData, 203
~Node
Node, 206
~PhyloRuleDynData
PhyloRuleDynData, 215
~PowerSet
PowerSet< Array_Type, Data_Rule_Type >, 218
~Progress
Progress, 223
~Rule
Rule< Array_Type, Data_Type >, 225
~Rules
Rules< Array_Type, Data_Type >, 228
~StatsCounter
StatsCounter< Array_Type, Data_Type >, 233
~Support
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 238

active
Cell< Cell_Type >, 125
add
Cell< Cell_Type >, 123, 124
FreqTable< T >, 158
add_array
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 180
add_counter
Counters< Array_Type, Data_Type >, 135, 136
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 181
StatsCounter< Array_Type, Data_Type >, 233
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 238
add_data
Flock, 151
add_dims
counters-meat.hpp, 285
add_hash
Counters< Array_Type, Data_Type >, 136
add_hasher
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 181
add_rule

Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 181
PowerSet< Array_Type, Data_Rule_Type >, 218, 219
Rules< Array_Type, Data_Type >, 229
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 238, 239
add_rule_dyn
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 182
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 239
annotations
Node, 207
Array
ConstBArrayRowIter< Cell_Type, Data_Type >, 127
array
Node, 207
Phylo rules, 44
arrays
Node, 207
arrays2support
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 194
AS_ONE
EXISTS, 57
as_vector
FreqTable< T >, 158
AS_ZERO
EXISTS, 57
at
Phylo rules, 39
PhyloCounterData, 212

BArray
BArray< Cell_Type, Data_Type >, 62, 63
BArray< Cell_Type, Data_Type >, 59
~BArray, 63
BArray, 62, 63
BArrayCell< Cell_Type, Data_Type >, 73
BArrayCell_const< Cell_Type, Data_Type >, 73
clear, 63
col, 63
D, 64
D_ptr, 64
default_val, 64
flush_data, 64
get_cell, 64
get_col_vec, 65
get_entries, 65
get_row_vec, 65
insert_cell, 66
is_dense, 66
is_empty, 66
ncol, 67
nnozero, 67
nrow, 67
operator*=, 67
operator(), 67
operator+=, 68
operator-=, 68
operator/=, 69
operator=, 69
operator==, 69
out_of_range, 69
print, 69
print_n, 70
reserve, 70
resize, 70
rm_cell, 70
row, 70
set_data, 71
swap_cells, 71
swap_cols, 71
swap_rows, 71
toggle_cell, 72
toggle_lock, 72
transpose, 72
visited, 73
zero_col, 72
zero_row, 72
barray-meat-operators.hpp
BARRAY_TEMPLATE, 250–252
BARRAY_TEMPLATE_ARGS, 251, 253
BARRAY_TYPE, 251, 253
COL, 251
for, 253
operator(), 253
rhs, 253
ROW, 251
this, 254
barray-meat.hpp
COL, 254
ROW, 255
BARRAY_TEMPLATE
barray-meat-operators.hpp, 250–252
BARRAY_TEMPLATE_ARGS
barray-meat-operators.hpp, 251, 253
BARRAY_TYPE
barray-meat-operators.hpp, 251, 253
BArrayCell
BArrayCell< Cell_Type, Data_Type >, 74
BArrayCell< Cell_Type, Data_Type >, 74
~BArrayCell, 74
BArray< Cell_Type, Data_Type >, 73
BArrayCell, 74
operator Cell_Type, 75
operator*=, 75
operator+=, 75
operator-=, 75
operator/=, 75

operator=, 75
operator==, 76
BArrayCell_const
BArrayCell_const< Cell_Type, Data_Type >, 77
BArrayCell_const< Cell_Type, Data_Type >, 76
~BArrayCell_const, 77
BArray< Cell_Type, Data_Type >, 73
BArrayCell_const, 77
operator Cell_Type, 77
operator!=, 77
operator<, 77
operator<=, 78
operator>, 78
operator>=, 78
operator==, 78
BArrayDense
BArrayDense< Cell_Type, Data_Type >, 82, 83
BArrayDense< Cell_Type, Data_Type >, 79
~BArrayDense, 83
BArrayDense, 82, 83
BArrayDenseCell< Cell_Type, Data_Type >, 95, 99
BArrayDenseCol< Cell_Type, Data_Type >, 95, 102
BArrayDenseCol_const< Cell_Type, Data_Type >, 95
BArrayDenseRow< Cell_Type, Data_Type >, 95, 106
BArrayDenseRow_const< Cell_Type, Data_Type >, 95
clear, 83
col, 84
colsum, 84
D, 84
D_ptr, 85
default_val, 85
get_cell, 85
get_col_vec, 85, 86
get_data, 86
get_entries, 86
get_row_vec, 86, 87
insert_cell, 87
is_dense, 87
is_empty, 88
ncol, 88
nnozero, 88
nrow, 88
operator*=, 89
operator(), 88, 89
operator+=, 89
operator-=, 90
operator/=, 90
operator=, 90, 91
operator==, 91
out_of_range, 91
print, 91
reserve, 91
resize, 92
rm_cell, 92
row, 92
rowsum, 92
set_data, 93
swap_cells, 93
swap_cols, 93
swap_rows, 93
toggle_cell, 94
toggle_lock, 94
transpose, 94
visited, 96
zero_col, 94
zero_row, 94
barraydense-meat-operators.hpp
BDENSE_TEMPLATE, 257, 259
BDENSE_TEMPLATE_ARGS, 257, 259
BDENSE_TYPE, 258, 260
COL, 258
POS, 258
POS_N, 258
ROW, 258
barraydense-meat.hpp
COL, 260
POS, 260
POS_N, 261
ROW, 261
ZERO_CELL, 261
BArrayDenseCell
BArrayDenseCell< Cell_Type, Data_Type >, 97
BArrayDenseCell< Cell_Type, Data_Type >, 96
~BArrayDenseCell, 97
BArrayDense< Cell_Type, Data_Type >, 95, 99
BArrayDenseCell, 97
BArrayDenseCol< Cell_Type, Data_Type >, 99, 102
BArrayDenseCol_const< Cell_Type, Data_Type >, 99, 104
BArrayDenseRow< Cell_Type, Data_Type >, 106
BArrayDenseRow_const< Cell_Type, Data_Type >, 109
operator Cell_Type, 97
operator*=, 97
operator+=, 98
operator-=, 98
operator/=, 98
operator=, 98
operator==, 99
barraydensecell-bones.hpp
POS, 262
barraydensecell-meat.hpp
POS, 262
BArrayDenseCell_const< Cell_Type, Data_Type >, 100
BArrayDenseCol< Cell_Type, Data_Type >, 102
BArrayDenseCol_const< Cell_Type, Data_Type >, 104
BArrayDenseRow< Cell_Type, Data_Type >, 106
BArrayDenseRow_const< Cell_Type, Data_Type >, 109

BArrayDenseCol
BArrayDenseCol< Cell_Type, Data_Type >, 100
BArrayDenseCol< Cell_Type, Data_Type >, 100
BArrayDense< Cell_Type, Data_Type >, 95, 102
BArrayDenseCell< Cell_Type, Data_Type >, 99, 102
BArrayDenseCell_const< Cell_Type, Data_Type >, 102
BArrayDenseCol, 100
begin, 101
end, 101
operator(), 101
size, 101
barraydensecol-bones.hpp
POS, 263
POS_N, 263
ZERO_CELL, 264
BArrayDenseCol_const
BArrayDenseCol_const< Cell_Type, Data_Type >, 103
BArrayDenseCol_const< Cell_Type, Data_Type >, 102
BArrayDense< Cell_Type, Data_Type >, 95
BArrayDenseCell< Cell_Type, Data_Type >, 99, 104
BArrayDenseCell_const< Cell_Type, Data_Type >, 104
BArrayDenseCol_const, 103
begin, 103
end, 103
operator(), 103
size, 104
BArrayDenseRow
BArrayDenseRow< Cell_Type, Data_Type >, 105
BArrayDenseRow< Cell_Type, Data_Type >, 104
BArrayDense< Cell_Type, Data_Type >, 95, 106
BArrayDenseCell< Cell_Type, Data_Type >, 106
BArrayDenseCell_const< Cell_Type, Data_Type >, 106
BArrayDenseRow, 105
begin, 105
end, 105
operator(), 106
size, 106
barraydenserow-bones.hpp
POS, 265
POS_N, 265
ZERO_CELL, 265
BArrayDenseRow_const
BArrayDenseRow_const< Cell_Type, Data_Type >, 107
BArrayDenseRow_const< Cell_Type, Data_Type >, 107
BArrayDense< Cell_Type, Data_Type >, 95
BArrayDenseCell< Cell_Type, Data_Type >, 109
BArrayDenseCell_const< Cell_Type, Data_Type >, 109
BArrayDenseRow_const, 107
begin, 108
end, 108
operator(), 108
size, 108
BArrayRow
BArrayRow< Cell_Type, Data_Type >, 110
BArrayRow< Cell_Type, Data_Type >, 109
~BArrayRow, 110
BArrayRow, 110
operator BArrayRow< Cell_Type, Data_Type >, 110
operator*=, 110
operator+=, 110
operator-=, 110
operator/=, 111
operator=, 111
operator==, 111
barrayrow-meat.hpp
BROW_TEMPLATE, 266, 267
BROW_TEMPLATE_ARGS, 266
BROW_TYPE, 266
BArrayRow_const
BArrayRow_const< Cell_Type, Data_Type >, 112
BArrayRow_const< Cell_Type, Data_Type >, 111
~BArrayRow_const, 112
BArrayRow_const, 112
operator BArrayRow_const< Cell_Type, Data_Type >, 112
operator!=, 112
operator<, 112
operator<=, 113
operator>, 113
operator>=, 113
operator==, 113
BArrayVector
BArrayVector< Cell_Type, Data_Type >, 114
BArrayVector< Cell_Type, Data_Type >, 113
~BArrayVector, 115
BArrayVector, 114
begin, 115
end, 115
is_col, 115
is_row, 115
operator std::vector< Cell_Type >, 116
operator*=, 116
operator+=, 116
operator-=, 116
operator/=, 116
operator=, 117
operator==, 117
size, 117
BArrayVector_const
BArrayVector_const< Cell_Type, Data_Type >, 118
BArrayVector_const< Cell_Type, Data_Type >, 117
~BArrayVector_const, 118
BArrayVector_const, 118
begin, 119
end, 119

is_col, 119
is_row, 119
operator std::vector< Cell_Type >, 119
operator!=, 119
operator<, 120
operator<=, 120
operator>, 120
operator>=, 120
operator==, 120
size, 121
barry, 55
barry-configuration.hpp
BARRY_CHECK_SUPPORT, 269
BARRY_ISFINITE, 269
BARRY_MAX_NUM_ELEMENTS, 269
BARRY_SAFE_EXP, 269
Map, 270
printf_barry, 269
barry-debug.hpp
BARRY_DEBUG_LEVEL, 270
barry-macros.hpp
BARRY_NCORES_ARG, 271
BARRY_ONE, 271
BARRY_ONE_DENSE, 271
BARRY_UNUSED, 272
BARRY_ZERO, 272
BARRY_ZERO_DENSE, 272
barry.hpp
BARRY_HPP, 274
BARRY_VERSION, 274
BARRY_VERSION_MAYOR, 274
BARRY_VERSION_MINOR, 274
COUNTER_FUNCTION, 274
COUNTER_LAMBDA, 274
RULE_FUNCTION, 275
RULE_LAMBDA, 275
barry::counters, 55
barry::counters::network, 56
BARRY_CHECK_SUPPORT
barry-configuration.hpp, 269
BARRY_DEBUG_LEVEL
barry-debug.hpp, 270
BARRY_HPP
barry.hpp, 274
BARRY_ISFINITE
barry-configuration.hpp, 269
BARRY_MAX_NUM_ELEMENTS
barry-configuration.hpp, 269
BARRY_NCORES_ARG
barry-macros.hpp, 271
BARRY_ONE
barry-macros.hpp, 271
BARRY_ONE_DENSE
barry-macros.hpp, 271
BARRY_PROGRESS_BAR_WIDTH
progress.hpp, 331
BARRY_SAFE_EXP
barry-configuration.hpp, 269
BARRY_SUPPORT_MEAT_HPP
support-meat.hpp, 339
BARRY_UNUSED
barry-macros.hpp, 272
BARRY_VERSION
barry.hpp, 274
BARRY_VERSION_MAYOR
barry.hpp, 274
BARRY_VERSION_MINOR
barry.hpp, 274
BARRY_ZERO
barry-macros.hpp, 272
BARRY_ZERO_DENSE
barry-macros.hpp, 272
BARRY_ZERO_NETWORK
network.hpp, 300
BARRY_ZERO_NETWORK_DENSE
network.hpp, 300
BDENSE_TEMPLATE
barraydense-meat-operators.hpp, 257, 259
BDENSE_TEMPLATE_ARGS
barraydense-meat-operators.hpp, 257, 259
BDENSE_TYPE
barraydense-meat-operators.hpp, 258, 260
begin
BArrayDenseCol< Cell_Type, Data_Type >, 101
BArrayDenseCol_const< Cell_Type, Data_Type >, 103
BArrayDenseRow< Cell_Type, Data_Type >, 105
BArrayDenseRow_const< Cell_Type, Data_Type >, 108
BArrayVector< Cell_Type, Data_Type >, 115
BArrayVector_const< Cell_Type, Data_Type >, 119
PhyloCounterData, 212
PowerSet< Array_Type, Data_Rule_Type >, 219
Rules< Array_Type, Data_Type >, 229
blengths
NodeData, 210
BOTH
CHECK, 56
EXISTS, 57
BROW_TEMPLATE
barrayrow-meat.hpp, 266, 267
BROW_TEMPLATE_ARGS
barrayrow-meat.hpp, 266
BROW_TYPE
barrayrow-meat.hpp, 266

calc
PowerSet< Array_Type, Data_Rule_Type >, 219
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 239
calc_reduced_sequence
Geese, 164
calc_sequence
Geese, 165
Cell

Cell< Cell_Type >, 122, 123
Cell< Cell_Type >, 121
~Cell, 122
active, 125
add, 123, 124
Cell, 122, 123
operator Cell_Type, 124
operator!=, 124
operator=, 124, 125
operator==, 125
value, 125
visited, 125
Cell_const< Cell_Type >, 126
change_stats
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 243
CHECK, 56
BOTH, 56
NONE, 56
ONE, 56
TWO, 56
clear
BArray< Cell_Type, Data_Type >, 63
BArrayDense< Cell_Type, Data_Type >, 83
FreqTable< T >, 158
statscounter-meat.hpp, 334
COL
barray-meat-operators.hpp, 251
barray-meat.hpp, 254
barraydense-meat-operators.hpp, 258
barraydense-meat.hpp, 260
col
BArray< Cell_Type, Data_Type >, 63
BArrayDense< Cell_Type, Data_Type >, 84
Col_type
typedefs.hpp, 342
colnames
Flock, 152
Geese, 165
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 182
colsum
BArrayDense< Cell_Type, Data_Type >, 84
conditional_prob
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 182
ConstBArrayRowIter
ConstBArrayRowIter< Cell_Type, Data_Type >, 127
ConstBArrayRowIter< Cell_Type, Data_Type >, 126
~ConstBArrayRowIter, 127
Array, 127
ConstBArrayRowIter, 127
current_col, 128
current_row, 128
iter, 128
coordiantes_n_free
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 243
coordiantes_n_locked
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 244
coordinates_free
PowerSet< Array_Type, Data_Rule_Type >, 221
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 244
coordinates_locked
PowerSet< Array_Type, Data_Rule_Type >, 221
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 244
count
Counter< Array_Type, Data_Type >, 131
count_all
StatsCounter< Array_Type, Data_Type >, 233
count_current
StatsCounter< Array_Type, Data_Type >, 234
count_fun
Counter< Array_Type, Data_Type >, 132
counters-meat.hpp, 280
count_fun_
counters-meat.hpp, 285
count_init
StatsCounter< Array_Type, Data_Type >, 234
Counter
Counter< Array_Type, Data_Type >, 130
counter
counters-meat.hpp, 286
statscounter-meat.hpp, 336
Counter< Array_Type, Data_Type >, 128
~Counter, 130
count, 131
count_fun, 132
Counter, 130
data, 132
desc, 133
get_description, 131
get_hasher, 131
get_name, 131
hasher_fun, 133
init, 131
init_fun, 133
name, 133
operator=, 131, 132
set_hasher, 132
counter_
counters-meat.hpp, 286
counter_absdiff
Network counters, 26
counter_co_opt

Counting, 17
Phylo counters, 48
counter_cogain
Counting, 18
Phylo counters, 49
counter_css_census01
network-css.hpp, 293
counter_css_census02
network-css.hpp, 293
counter_css_census03
network-css.hpp, 293
counter_css_census04
network-css.hpp, 294
counter_css_census05
network-css.hpp, 294
counter_css_census06
network-css.hpp, 294
counter_css_census07
network-css.hpp, 294
counter_css_census08
network-css.hpp, 295
counter_css_census09
network-css.hpp, 295
counter_css_census10
network-css.hpp, 295
counter_css_completely_false_recip_comiss
network-css.hpp, 295
counter_css_completely_false_recip_omiss
network-css.hpp, 296
counter_css_mixed_recip
network-css.hpp, 296
counter_css_partially_false_recip_commi
network-css.hpp, 296
counter_css_partially_false_recip_omiss
network-css.hpp, 297
counter_ctriads
Network counters, 27
counter_degree
Network counters, 27
counter_deleted
statscounter-meat.hpp, 337
counter_density
Network counters, 27
counter_diff
Network counters, 28
counter_edges
Network counters, 28
counter_fixed_effect
Network counters, 28
counter_fun
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 194
Counter_fun_type
typedefs.hpp, 342
COUNTER_FUNCTION
barry.hpp, 274
counter_gains
Counting, 18
Phylo counters, 49
counter_gains_from_0
Counting, 18
Phylo counters, 49
counter_gains_k_offspring
Counting, 19
Phylo counters, 49
counter_genes_changing
Counting, 19
Phylo counters, 50
counter_idegree
Network counters, 29
counter_idegree15
Network counters, 29
counter_isolates
Network counters, 29, 30
counter_istar2
Network counters, 30
counter_k_genes_changing
Counting, 19
Phylo counters, 50
COUNTER_LAMBDA
barry.hpp, 274
counter_less_than_p_prop_genes_changing
Counting, 19
Phylo counters, 50
counter_logit_intercept
Network counters, 30
counter_longest
Counting, 20
Phylo counters, 50
counter_loss
Counting, 20
Phylo counters, 51
counter_maxfuns
Counting, 20
Phylo counters, 51
counter_mutual
Network counters, 30
counter_neofun
Counting, 20
Phylo counters, 51
counter_neofun_a2b
Counting, 21
Phylo counters, 51
counter_nodecov
Network counters, 31
counter_nodeicov
Network counters, 31
counter_nodematch
Network counters, 31
counter_nodeocov
Network counters, 31
counter_odegree
Network counters, 31, 32
counter_odegree15
Network counters, 32

counter_ones
Network counters, 32
counter_ostar2
Network counters, 33
counter_overall_changes
Counting, 21
Phylo counters, 52
counter_overall_gains
Counting, 21
Phylo counters, 52
counter_overall_gains_from_0
Counting, 21
Phylo counters, 52
counter_overall_loss
Counting, 22
Phylo counters, 52
counter_pairwise_first_gain
Counting, 22
Phylo counters, 53
counter_pairwise_neofun_singlefun
Counting, 22
Phylo counters, 53
counter_pairwise_overall_change
Counting, 22
Phylo counters, 53
counter_pairwise_preserving
Counting, 23
Phylo counters, 53
counter_preserve_pseudogene
Counting, 23
Phylo counters, 54
counter_prop_genes_changing
Counting, 23
Phylo counters, 54
counter_subfun
Counting, 23
Phylo counters, 54
COUNTER_TEMPLATE
counters-meat.hpp, 278, 280, 281
COUNTER_TEMPLATE_ARGS
counters-meat.hpp, 278
counter_transition
Network counters, 33
counter_transition_formula
Network counters, 34
counter_ttriads
Network counters, 34
COUNTER_TYPE
counters-meat.hpp, 279
Counters
Counters< Array_Type, Data_Type >, 134, 135
counters
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 194
statscounter-meat.hpp, 337
Counters< Array_Type, Data_Type >, 134
~Counters, 135
add_counter, 135, 136
add_hash, 136
Counters, 134, 135
gen_hash, 136
get_descriptions, 136
get_names, 137
operator=, 137
operator[], 138
size, 138
counters-meat.hpp
add_dims, 285
count_fun, 280
count_fun_, 285
counter, 286
counter_, 286
COUNTER_TEMPLATE, 278, 280, 281
COUNTER_TEMPLATE_ARGS, 278
COUNTER_TYPE, 279
COUNTERS_TEMPLATE, 279, 281–283
COUNTERS_TEMPLATE_ARGS, 279
COUNTERS_TYPE, 279
data, 283
data_, 286
desc, 283
desc_, 286
for, 283
fun, 287
fun_, 287
hasher, 283, 284
hasher_fun, 284
hasher_fun_, 287
i, 287
if, 284
init_fun, 285
init_fun_, 288
j, 288
name, 285
name_, 288
noexcept, 288
res, 288
return, 289
TMP_HASHER_CALL, 279
counters.hpp
DEFM_COUNTER, 308
DEFM_COUNTER_LAMBDA, 309
DEFM_RULE, 309
DEFM_RULE_LAMBDA, 309
DEFM_RULEDYN_LAMBDA, 309
UNI_SUB, 310
counters_
statscounter-meat.hpp, 337
COUNTERS_TEMPLATE
counters-meat.hpp, 279, 281–283
COUNTERS_TEMPLATE_ARGS
counters-meat.hpp, 279
COUNTERS_TYPE
counters-meat.hpp, 279
Counting, 13

counter_co_opt, 17
counter_cogain, 18
counter_gains, 18
counter_gains_from_0, 18
counter_gains_k_offspring, 19
counter_genes_changing, 19
counter_k_genes_changing, 19
counter_less_than_p_prop_genes_changing, 19
counter_longest, 20
counter_loss, 20
counter_maxfuns, 20
counter_neofun, 20
counter_neofun_a2b, 21
counter_overall_changes, 21
counter_overall_gains, 21
counter_overall_gains_from_0, 21
counter_overall_loss, 22
counter_pairwise_first_gain, 22
counter_pairwise_neofun_singlefun, 22
counter_pairwise_overall_change, 22
counter_pairwise_preserving, 23
counter_preserve_pseudogene, 23
counter_prop_genes_changing, 23
counter_subfun, 23
get_last_name, 24
IF_MATCHES, 15
IF_NOTMATCHES, 15
IS_DUPLICATION, 15
IS_EITHER, 15
IS_SPECIATION, 16
MAKE_DEFM_HASHER, 16
MAKE_DUPL_VARS, 16
PHYLO_CHECK_MISSING, 16
PHYLO_COUNTER_LAMBDA, 17
PHYLO_RULE_DYN_LAMBDA, 17

counts
Phylo rules, 44
PhyloRuleDynData, 215

Counts_type
typedefs.hpp, 342

covar_sort
Phylo rules, 44

covar_used
Phylo rules, 44

covariates
Phylo rules, 44

CSS_APPEND
network-css.hpp, 290

CSS_CASE_ELSE
network-css.hpp, 291

CSS_CASE_PERCEIVED
network-css.hpp, 291

CSS_CASE_TRUTH
network-css.hpp, 291

CSS_CHECK_SIZE
network-css.hpp, 291

CSS_CHECK_SIZE_INIT
network-css.hpp, 291

CSS_MATCH_TYPE
network-css.hpp, 292

CSS_NET_COUNTER_LAMBDA_INIT
network-css.hpp, 292

CSS_PERCEIVED_CELLS
network-css.hpp, 292

CSS_SIZE
network-css.hpp, 292

CSS_TRUE_CELLS
network-css.hpp, 293

current_col
ConstBArrayRowIter< Cell_Type, Data_Type >, 128

current_row
ConstBArrayRowIter< Cell_Type, Data_Type >, 128

current_stats
statscounter-meat.hpp, 337
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 244

D
BArray< Cell_Type, Data_Type >, 64
BArrayDense< Cell_Type, Data_Type >, 84
Rule< Array_Type, Data_Type >, 226

D_ptr
BArray< Cell_Type, Data_Type >, 64
BArrayDense< Cell_Type, Data_Type >, 85

dat
Flock, 156

data
Counter< Array_Type, Data_Type >, 132
counters-meat.hpp, 283
PowerSet< Array_Type, Data_Rule_Type >, 221

data_
counters-meat.hpp, 286

default_val
BArray< Cell_Type, Data_Type >, 64
BArrayDense< Cell_Type, Data_Type >, 85

DEFM, 139
DEFM, 140
get_column_major, 140
get_ID, 140
get_m_order, 140
get_model, 141
get_n_covars, 141
get_n_obs, 141
get_n_rows, 141
get_n_y, 141
get_X, 141
get_X_names, 142
get_Y, 142
get_Y_names, 142
init, 142
is_motif, 142
logodds, 142
motif_census, 143
print, 143

set_names, 143
simulate, 143
defm, 57
defm-meat.hpp
DEFM_LOOP_ARRAYS, 313
DEFM_RANGES, 313
keygen_defm, 314
defm-types.hpp
DEFMArray, 315
DEFM_COUNTER
counters.hpp, 308
DEFM_COUNTER_LAMBDA
counters.hpp, 309
DEFM_LOOP_ARRAYS
defm-meat.hpp, 313
defm_motif_parser
formula.hpp, 315
DEFM_RANGES
defm-meat.hpp, 313
DEFM_RULE
counters.hpp, 309
DEFM_RULE_LAMBDA
counters.hpp, 309
DEFM_RULEDYN_LAMBDA
counters.hpp, 309
DEFMArray
defm-types.hpp, 315
DEFMCounter
Phylo rules, 37
DEFMCounterData, 144
Phylo rules, 39
DEFMCounters
Phylo rules, 37
DEFMData, 144
Phylo rules, 39
DEFMModel
Phylo rules, 37
DEFMRule
Phylo rules, 38
DEFMRuleData, 146
Phylo rules, 40
DEFMRuleDyn
Phylo rules, 38
DEFMRuleDynData, 147
Phylo rules, 40
DEFMRules
Phylo rules, 38
DEFMRulesDyn
Phylo rules, 38
DEFMStatsCounter
Phylo rules, 38
DEFMSupport
Phylo rules, 38
delete_counters
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 194
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 244
delete_rengine
Geese, 172
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 194
delete_rules
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 195
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 245
delete_rules_dyn
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 195
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 245
delete_support
Geese, 172
desc
Counter< Array_Type, Data_Type >, 133
counters-meat.hpp, 283
desc_
counters-meat.hpp, 286
directed
NetworkData, 203
duplication
Node, 208
NodeData, 211
PhyloRuleDynData, 215

empty
PhyloCounterData, 212
EmptyArray
PowerSet< Array_Type, Data_Rule_Type >, 221
statscounter-meat.hpp, 337
end
BArrayDenseCol< Cell_Type, Data_Type >, 101
BArrayDenseCol_const< Cell_Type, Data_Type >, 103
BArrayDenseRow< Cell_Type, Data_Type >, 105
BArrayDenseRow_const< Cell_Type, Data_Type >, 108
BArrayVector< Cell_Type, Data_Type >, 115
BArrayVector_const< Cell_Type, Data_Type >, 119
PhyloCounterData, 213
PowerSet< Array_Type, Data_Rule_Type >, 219
Progress, 224
Rules< Array_Type, Data_Type >, 229
Entries
Entries< Cell_Type >, 148
Entries< Cell_Type >, 148
~Entries, 149
Entries, 148

resize, 149
source, 149
target, 149
val, 149
etype_default
Geese, 172
etype_duplication
Geese, 172
etype_either
Geese, 172
etype_speciation
Geese, 173
eval_rules_dyn
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 240
EXISTS, 57
AS_ONE, 57
AS_ZERO, 57
BOTH, 57
NONE, 58
ONE, 58
TWO, 58
UKNOWN, 58

f_
statscounter-meat.hpp, 338
first_calc_done
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 195
Flock, 150
∼Flock, 151
add_data, 151
colnames, 152
dat, 156
Flock, 151
get_counters, 152
get_model, 152
get_stats_support, 152
get_stats_target, 152
get_support_fun, 153
init, 153
initialized, 156
likelihood_joint, 153
model, 156
nfunctions, 156
nfuns, 153
nleafs, 154
nnodes, 154
nterms, 154
ntrees, 154
operator(), 154
parse_polytomies, 155
print, 155
rengine, 156
set_seed, 155
support_size, 155
flush_data
BArray< Cell_Type, Data_Type >, 64
for
barray-meat-operators.hpp, 253
counters-meat.hpp, 283
statscounter-meat.hpp, 334
formula.hpp
defm_motif_parser, 315
FreqTable
FreqTable< T >, 158
FreqTable< T >, 157
∼FreqTable, 158
add, 158
as_vector, 158
clear, 158
FreqTable, 158
get_data, 159
get_index, 159
make_hash, 159
print, 159
reserve, 159
size, 160
fun
counters-meat.hpp, 287
fun_
counters-meat.hpp, 287

Geese, 160
∼Geese, 164
calc_reduced_sequence, 164
calc_sequence, 165
colnames, 165
delete_rengine, 172
delete_support, 172
etype_default, 172
etype_duplication, 172
etype_either, 172
etype_speciation, 173
Geese, 163, 164
get_annotated_nodes, 165
get_annotations, 165
get_counters, 165
get_model, 166
get_probabilities, 166
get_rengine, 166
get_states, 166
get_support_fun, 166
inherit_support, 167
init, 167
init_node, 167
initialized, 173
likelihood, 167
likelihood_exhaust, 167
map_to_state_id, 173
nannotations, 168
nfunctions, 173
nfuns, 168
nleafs, 168
nnodes, 168
nodes, 173

nterms, 168
observed_counts, 169
operator=, 169
parse_polytomies, 169
predict, 169
predict_backend, 170
predict_exhaust, 170
predict_exhaust_backend, 170
predict_sim, 170
print, 170
print_nodes, 171
print_observed_counts, 171
pset_loc, 173
reduced_sequence, 174
sequence, 174
set_seed, 171
simulate, 171
support_size, 171
update_annotations, 171
geese, 58
geese-bones.hpp
INITIALIZED, 319
keygen_full, 319
RULE_FUNCTION, 319
vec_diff, 319
vector_caster, 320
geese-meat-likelihood.hpp
pset_loop, 321
geese-types.hpp
PhyloArray, 327
PhyloCounter, 327
PhyloCounters, 327
PhyloModel, 327
PhyloPowerSet, 327
PhyloRule, 328
PhyloRuleData, 328
PhyloRuleDyn, 328
PhyloRules, 328
PhyloRulesDyn, 328
PhyloStatsCounter, 328
PhyloSupport, 329
POS, 326
gen_hash
Counters< Array_Type, Data_Type >, 136
gen_key
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 183
get_annotated_nodes
Geese, 165
get_annotations
Geese, 165
get_arrays2support
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 183
get_cell
BArray< Cell_Type, Data_Type >, 64
BArrayDense< Cell_Type, Data_Type >, 85
get_col_vec
BArray< Cell_Type, Data_Type >, 65
BArrayDense< Cell_Type, Data_Type >, 85, 86
get_column_major
DEFM, 140
get_counters
Flock, 152
Geese, 165
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 183
PhyloCounterData, 213
StatsCounter< Array_Type, Data_Type >, 234
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 240
get_counts
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 240
get_current_stats
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 240
get_data
BArrayDense< Cell_Type, Data_Type >, 86
FreqTable< T >, 159
PowerSet< Array_Type, Data_Rule_Type >, 219
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 241
get_data_ptr
PowerSet< Array_Type, Data_Rule_Type >, 220
get_description
Counter< Array_Type, Data_Type >, 131
Rule< Array_Type, Data_Type >, 226
get_descriptions
Counters< Array_Type, Data_Type >, 136
Rules< Array_Type, Data_Type >, 229
StatsCounter< Array_Type, Data_Type >, 234
get_entries
BArray< Cell_Type, Data_Type >, 65
BArrayDense< Cell_Type, Data_Type >, 86
get_hasher
Counter< Array_Type, Data_Type >, 131
get_ID
DEFM, 140
get_index
FreqTable< T >, 159
get_last_name
Counting, 24
get_likelihoods
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 183
get_m_order
DEFM, 140

get_model
DEFM, 141
Flock, 152
Geese, 166
get_n_covars
DEFM, 141
get_n_obs
DEFM, 141
get_n_rows
DEFM, 141
get_n_y
DEFM, 141
get_name
Counter< Array_Type, Data_Type >, 131
Rule< Array_Type, Data_Type >, 226
get_names
Counters< Array_Type, Data_Type >, 137
Rules< Array_Type, Data_Type >, 230
StatsCounter< Array_Type, Data_Type >, 234
get_normalizing_constants
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 184
get_parent
Node, 206
get_probabilities
Geese, 166
get_pset
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 184
get_pset_arrays
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 184
get_pset_probs
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 184
get_pset_stats
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 184, 185
get_rengine
Geese, 166
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 185
get_row_vec
BArray< Cell_Type, Data_Type >, 65
BArrayDense< Cell_Type, Data_Type >, 86, 87
get_rules
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 185
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 241
get_rules_dyn
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 185
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 241
get_seq
Rules< Array_Type, Data_Type >, 230
get_states
Geese, 166
get_stats_support
Flock, 152
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 186
get_stats_support_sizes
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 186
get_stats_support_sizes_acc
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 186
get_stats_target
Flock, 152
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 186
get_support_fun
Flock, 153
Geese, 166
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 187
get_X
DEFM, 141
get_X_names
DEFM, 142
get_Y
DEFM, 142
get_Y_names
DEFM, 142

hasher
counters-meat.hpp, 283, 284
hasher_fun
Counter< Array_Type, Data_Type >, 133
counters-meat.hpp, 284
hasher_fun_
counters-meat.hpp, 287
Hasher_fun_type
typedefs.hpp, 342
hashes
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 245
hashes_initialized

Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 245

i
counters-meat.hpp, 287
id
Node, 208
idx
Phylo rules, 40, 41
if
counters-meat.hpp, 284
IF_MATCHES
Counting, 15
IF_NOTMATCHES
Counting, 15
include/barry/barray-bones.hpp, 249
include/barry/barray-iterator.hpp, 249
include/barry/barray-meat-operators.hpp, 250
include/barry/barray-meat.hpp, 254
include/barry/barraycell-bones.hpp, 255
include/barry/barraycell-meat.hpp, 256
include/barry/barraydense-bones.hpp, 256
include/barry/barraydense-meat-operators.hpp, 257
include/barry/barraydense-meat.hpp, 260
include/barry/barraydensecell-bones.hpp, 261
include/barry/barraydensecell-meat.hpp, 262
include/barry/barraydensecol-bones.hpp, 263
include/barry/barraydenserow-bones.hpp, 264
include/barry/barrayrow-bones.hpp, 265
include/barry/barrayrow-meat.hpp, 265
include/barry/barrayvector-bones.hpp, 267
include/barry/barrayvector-meat.hpp, 268
include/barry/barry-configuration.hpp, 268
include/barry/barry-debug.hpp, 270
include/barry/barry-macros.hpp, 271
include/barry/barry.hpp, 272
include/barry/cell-bones.hpp, 275
include/barry/cell-meat.hpp, 276
include/barry/col-bones.hpp, 276
include/barry/counters-bones.hpp, 276
include/barry/counters-meat.hpp, 277
include/barry/counters/network-css.hpp, 289
include/barry/counters/network.hpp, 297
include/barry/freqtable.hpp, 304
include/barry/model-bones.hpp, 305
include/barry/model-meat.hpp, 305
include/barry/models/defm.hpp, 306
include/barry/models/defm/counters.hpp, 307
include/barry/models/defm/defm-bones.hpp, 312
include/barry/models/defm/defm-meat.hpp, 313
include/barry/models/defm/defm-types.hpp, 314
include/barry/models/defm/formula.hpp, 315
include/barry/models/geese.hpp, 317
include/barry/models/geese/counters.hpp, 310
include/barry/models/geese/flock-bones.hpp, 317
include/barry/models/geese/flock-meat.hpp, 318
include/barry/models/geese/geese-bones.hpp, 318
include/barry/models/geese/geese-meat-constructors.hpp, 320
include/barry/models/geese/geese-meat-likelihood.hpp, 321
include/barry/models/geese/geese-meat-likelihood_exhaust.hpp, 322
include/barry/models/geese/geese-meat-predict.hpp, 323
include/barry/models/geese/geese-meat-predict_exhaust.hpp, 323
include/barry/models/geese/geese-meat-predict_sim.hpp, 324
include/barry/models/geese/geese-meat-simulate.hpp, 324
include/barry/models/geese/geese-meat.hpp, 325
include/barry/models/geese/geese-node-bones.hpp, 325
include/barry/models/geese/geese-types.hpp, 326
include/barry/powerset-bones.hpp, 329
include/barry/powerset-meat.hpp, 330
include/barry/progress.hpp, 330
include/barry/rules-bones.hpp, 331
include/barry/rules-meat.hpp, 332
include/barry/statscounter-bones.hpp, 332
include/barry/statscounter-meat.hpp, 333
include/barry/support-bones.hpp, 338
include/barry/support-meat.hpp, 339
include/barry/typedefs.hpp, 340
indices
NetCounterData, 201
Phylo rules, 44, 45
inherit_support
Geese, 167
init
Counter< Array_Type, Data_Type >, 131
DEFM, 142
Flock, 153
Geese, 167
Phylo rules, 45
init_fun
Counter< Array_Type, Data_Type >, 133
counters-meat.hpp, 285
init_fun_
counters-meat.hpp, 288
init_node
Geese, 167
init_support
PowerSet< Array_Type, Data_Rule_Type >, 220
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 241
INITIALIZED
geese-bones.hpp, 319
initialized
Flock, 156
Geese, 173
insert_cell
BArray< Cell_Type, Data_Type >, 66

BArrayDense< Cell_Type, Data_Type >, 87
is_col
BArrayVector< Cell_Type, Data_Type >, 115
BArrayVector_const< Cell_Type, Data_Type >, 119
is_dense
BArray< Cell_Type, Data_Type >, 66
BArrayDense< Cell_Type, Data_Type >, 87
IS_DUPLICATION
Counting, 15
IS_EITHER
Counting, 15
is_empty
BArray< Cell_Type, Data_Type >, 66
BArrayDense< Cell_Type, Data_Type >, 88
is_leaf
Node, 207
is_motif
DEFM, 142
Phylo rules, 45
is_row
BArrayVector< Cell_Type, Data_Type >, 115
BArrayVector_const< Cell_Type, Data_Type >, 119
IS_SPECIATION
Counting, 16
is_true
Phylo rules, 41
iter
ConstBArrayRowIter< Cell_Type, Data_Type >, 128

j
counters-meat.hpp, 288
statscounter-meat.hpp, 338

keygen_defm
defm-meat.hpp, 314
keygen_full
geese-bones.hpp, 319
keys2support
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 195

lb
PhyloRuleDynData, 216
likelihood
Geese, 167
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 187, 188
likelihood_
model-meat.hpp, 306
likelihood_exhaust
Geese, 167
likelihood_joint
Flock, 153
likelihood_total
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 188
logical
Phylo rules, 45
logodds
DEFM, 142

M
PowerSet< Array_Type, Data_Rule_Type >, 221
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 245
MAKE_DEFM_HASHER
Counting, 16
MAKE_DUPL_VARS
Counting, 16
make_hash
FreqTable< T >, 159
Map
barry-configuration.hpp, 270
map_to_state_id
Geese, 173
MapVec_type
typedefs.hpp, 343
max_num_elements
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 246
Model
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 179, 180
model
Flock, 156
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 175
~Model, 180
add_array, 180
add_counter, 181
add_hasher, 181
add_rule, 181
add_rule_dyn, 182
arrays2support, 194
colnames, 182
conditional_prob, 182
counter_fun, 194
counters, 194
delete_counters, 194
delete_rengine, 194
delete_rules, 195
delete_rules_dyn, 195
first_calc_done, 195
gen_key, 183
get_arrays2support, 183
get_counters, 183
get_likelihoods, 183
get_normalizing_constants, 184
get_pset, 184

get_pset_arrays, 184
get_pset_probs, 184
get_pset_stats, 184, 185
get_rengine, 185
get_rules, 185
get_rules_dyn, 185
get_stats_support, 186
get_stats_support_sizes, 186
get_stats_support_sizes_acc, 186
get_stats_target, 186
get_support_fun, 187
keys2support, 195
likelihood, 187, 188
likelihood_total, 188
Model, 179, 180
normalizing_constants, 195
nrules, 188
nrules_dyn, 188
nterms, 189
operator=, 189
params_last, 196
print, 189
print_stats, 189
pset_arrays, 196
pset_probs, 196
pset_stats, 196
rengine, 197
rules, 197
rules_dyn, 197
sample, 190
set_counters, 190
set_rengine, 190
set_rules, 191
set_rules_dyn, 191
set_seed, 191
set_transform_model, 191
size, 192
size_unique, 192
stats_likelihood, 197
stats_support, 197
stats_support_n_arrays, 198
stats_support_sizes, 198
stats_support_sizes_acc, 198
stats_target, 198
store_psets, 192
support_fun, 199
support_size, 192
transform_model, 192
transform_model_fun, 199
transform_model_term_names, 199
update_likelihoods, 193
update_normalizing_constants, 193
update_pset_probs, 193
with_pset, 200
model-meat.hpp
likelihood_, 306
update_normalizing_constant, 306
motif_census
DEFM, 143
N
PowerSet< Array_Type, Data_Rule_Type >, 222
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 246
n_counters
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 246
n_free
PowerSet< Array_Type, Data_Rule_Type >, 222
n_locked
PowerSet< Array_Type, Data_Rule_Type >, 222
name
Counter< Array_Type, Data_Type >, 133
counters-meat.hpp, 285
name_
counters-meat.hpp, 288
nannotations
Geese, 168
narray
Node, 208
ncol
BArray< Cell_Type, Data_Type >, 67
BArrayDense< Cell_Type, Data_Type >, 88
Phylo rules, 41
NET_C_DATA_IDX
network.hpp, 301
NET_C_DATA_NUM
network.hpp, 301
NetCounter
network.hpp, 302
NetCounterData, 200
∼NetCounterData, 201
indices, 201
NetCounterData, 201
numbers, 201
NetCounters
network.hpp, 302
NetModel
network.hpp, 303
NetRule
network.hpp, 303
NetRules
network.hpp, 303
NetStatsCounter
network.hpp, 303
NetSupport
network.hpp, 303
Network
network.hpp, 303
Network counters, 24
counter_absdiff, 26
counter_ctriads, 27
counter_degree, 27
counter_density, 27
counter_diff, 28

counter_edges, 28
counter_fixed_effect, 28
counter_idegree, 29
counter_idegree15, 29
counter_isolates, 29, 30
counter_istar2, 30
counter_logit_intercept, 30
counter_mutual, 30
counter_nodecov, 31
counter_nodeicov, 31
counter_nodematch, 31
counter_nodeocov, 31
counter_odegree, 31, 32
counter_odegree15, 32
counter_ones, 32
counter_ostar2, 33
counter_transition, 33
counter_transition_formula, 34
counter_ttriads, 34
NETWORK_COUNTER, 34
rules_dont_become_zero, 34
rules_markov_fixed, 35
network-css.hpp
counter_css_census01, 293
counter_css_census02, 293
counter_css_census03, 293
counter_css_census04, 294
counter_css_census05, 294
counter_css_census06, 294
counter_css_census07, 294
counter_css_census08, 295
counter_css_census09, 295
counter_css_census10, 295
counter_css_completely_false_recip_comiss, 295
counter_css_completely_false_recip_omiss, 296
counter_css_mixed_recip, 296
counter_css_partially_false_recip_commi, 296
counter_css_partially_false_recip_omiss, 297
CSS_APPEND, 290
CSS_CASE_ELSE, 291
CSS_CASE_PERCEIVED, 291
CSS_CASE_TRUTH, 291
CSS_CHECK_SIZE, 291
CSS_CHECK_SIZE_INIT, 291
CSS_MATCH_TYPE, 292
CSS_NET_COUNTER_LAMBDA_INIT, 292
CSS_PERCEIVED_CELLS, 292
CSS_SIZE, 292
CSS_TRUE_CELLS, 293
network.hpp
BARRY_ZERO_NETWORK, 300
BARRY_ZERO_NETWORK_DENSE, 300
NET_C_DATA_IDX, 301
NET_C_DATA_NUM, 301
NetCounter, 302
NetCounters, 302
NetModel, 303
NetRule, 303
NetRules, 303
NetStatsCounter, 303
NetSupport, 303
Network, 303
NETWORK_COUNTER, 301
NETWORK_COUNTER_LAMBDA, 301
NETWORK_RULE, 301
NETWORK_RULE_LAMBDA, 302
NetworkDense, 304
NETWORKDENSE_COUNTER_LAMBDA, 302
rules_zerodiag, 304
NETWORK_COUNTER
Network counters, 34
network.hpp, 301
NETWORK_COUNTER_LAMBDA
network.hpp, 301
NETWORK_RULE
network.hpp, 301
NETWORK_RULE_LAMBDA
network.hpp, 302
NetworkData, 202
~NetworkData, 203
directed, 203
NetworkData, 202, 203
vertex_attr, 203
NetworkDense
network.hpp, 304
NETWORKDENSE_COUNTER_LAMBDA
network.hpp, 302
next
Progress, 224
nfunctions
Flock, 156
Geese, 173
nfuns
Flock, 153
Geese, 168
nleafs
Flock, 154
Geese, 168
nnodes
Flock, 154
Geese, 168
nnozero
BArray< Cell_Type, Data_Type >, 67
BArrayDense< Cell_Type, Data_Type >, 88
Node, 204
~Node, 206
annotations, 207
array, 207
arrays, 207
duplication, 208
get_parent, 206
id, 208
is_leaf, 207
narray, 208
Node, 205, 206
noffspring, 207

offspring, 208
ord, 208
parent, 209
probability, 209
subtree_prob, 209
visited, 209
NodeData, 210
blengths, 210
duplication, 211
NodeData, 210
states, 211
nodes
Geese, 173
noexcept
counters-meat.hpp, 288
noffspring
Node, 207
NONE
CHECK, 56
EXISTS, 58
normalizing_constants
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 195
nrow
BArray< Cell_Type, Data_Type >, 67
BArrayDense< Cell_Type, Data_Type >, 88
Phylo rules, 41
nrules
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 188
nrules_dyn
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 188
nterms
Flock, 154
Geese, 168
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 189
ntrees
Flock, 154
num
Phylo rules, 41, 42
numbers
NetCounterData, 201
Phylo rules, 45, 46

obs_start
Phylo rules, 46
observed_counts
Geese, 169
offspring
Node, 208
ONE
CHECK, 56
EXISTS, 58

operator BArrayRow< Cell_Type, Data_Type >
BArrayRow< Cell_Type, Data_Type >, 110
operator BArrayRow_const< Cell_Type, Data_Type >
BArrayRow_const< Cell_Type, Data_Type >, 112
operator Cell_Type
BArrayCell< Cell_Type, Data_Type >, 75
BArrayCell_const< Cell_Type, Data_Type >, 77
BArrayDenseCell< Cell_Type, Data_Type >, 97
Cell< Cell_Type >, 124
operator std::vector< Cell_Type >
BArrayVector< Cell_Type, Data_Type >, 116
BArrayVector_const< Cell_Type, Data_Type >, 119
operator!=
BArrayCell_const< Cell_Type, Data_Type >, 77
BArrayRow_const< Cell_Type, Data_Type >, 112
BArrayVector_const< Cell_Type, Data_Type >, 119
Cell< Cell_Type >, 124
operator<
BArrayCell_const< Cell_Type, Data_Type >, 77
BArrayRow_const< Cell_Type, Data_Type >, 112
BArrayVector_const< Cell_Type, Data_Type >, 120
operator<=
BArrayCell_const< Cell_Type, Data_Type >, 78
BArrayRow_const< Cell_Type, Data_Type >, 113
BArrayVector_const< Cell_Type, Data_Type >, 120
operator>
BArrayCell_const< Cell_Type, Data_Type >, 78
BArrayRow_const< Cell_Type, Data_Type >, 113
BArrayVector_const< Cell_Type, Data_Type >, 120
operator>=
BArrayCell_const< Cell_Type, Data_Type >, 78
BArrayRow_const< Cell_Type, Data_Type >, 113
BArrayVector_const< Cell_Type, Data_Type >, 120
operator*=
BArray< Cell_Type, Data_Type >, 67
BArrayCell< Cell_Type, Data_Type >, 75
BArrayDense< Cell_Type, Data_Type >, 89
BArrayDenseCell< Cell_Type, Data_Type >, 97
BArrayRow< Cell_Type, Data_Type >, 110
BArrayVector< Cell_Type, Data_Type >, 116
operator()
BArray< Cell_Type, Data_Type >, 67
barray-meat-operators.hpp, 253
BArrayDense< Cell_Type, Data_Type >, 88, 89
BArrayDenseCol< Cell_Type, Data_Type >, 101
BArrayDenseCol_const< Cell_Type, Data_Type >, 103
BArrayDenseRow< Cell_Type, Data_Type >, 106
BArrayDenseRow_const< Cell_Type, Data_Type >, 108
Flock, 154
Phylo rules, 42

PhyloCounterData, 213
PhyloRuleDynData, 215
Rule< Array_Type, Data_Type >, 227
Rules< Array_Type, Data_Type >, 230
vecHasher< T >, 247
operator+=
BArray< Cell_Type, Data_Type >, 68
BArrayCell< Cell_Type, Data_Type >, 75
BArrayDense< Cell_Type, Data_Type >, 89
BArrayDenseCell< Cell_Type, Data_Type >, 98
BArrayRow< Cell_Type, Data_Type >, 110
BArrayVector< Cell_Type, Data_Type >, 116
operator-=
BArray< Cell_Type, Data_Type >, 68
BArrayCell< Cell_Type, Data_Type >, 75
BArrayDense< Cell_Type, Data_Type >, 90
BArrayDenseCell< Cell_Type, Data_Type >, 98
BArrayRow< Cell_Type, Data_Type >, 110
BArrayVector< Cell_Type, Data_Type >, 116
operator/=
BArray< Cell_Type, Data_Type >, 69
BArrayCell< Cell_Type, Data_Type >, 75
BArrayDense< Cell_Type, Data_Type >, 90
BArrayDenseCell< Cell_Type, Data_Type >, 98
BArrayRow< Cell_Type, Data_Type >, 111
BArrayVector< Cell_Type, Data_Type >, 116
operator=
BArray< Cell_Type, Data_Type >, 69
BArrayCell< Cell_Type, Data_Type >, 75
BArrayDense< Cell_Type, Data_Type >, 90, 91
BArrayDenseCell< Cell_Type, Data_Type >, 98
BArrayRow< Cell_Type, Data_Type >, 111
BArrayVector< Cell_Type, Data_Type >, 117
Cell< Cell_Type >, 124, 125
Counter< Array_Type, Data_Type >, 131, 132
Counters< Array_Type, Data_Type >, 137
Geese, 169
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 189
Rules< Array_Type, Data_Type >, 231
operator==
BArray< Cell_Type, Data_Type >, 69
BArrayCell< Cell_Type, Data_Type >, 76
BArrayCell_const< Cell_Type, Data_Type >, 78
BArrayDense< Cell_Type, Data_Type >, 91
BArrayDenseCell< Cell_Type, Data_Type >, 99
BArrayRow< Cell_Type, Data_Type >, 111
BArrayRow_const< Cell_Type, Data_Type >, 113
BArrayVector< Cell_Type, Data_Type >, 117
BArrayVector_const< Cell_Type, Data_Type >, 120
Cell< Cell_Type >, 125
operator[]
Counters< Array_Type, Data_Type >, 138
PhyloCounterData, 213
PowerSet< Array_Type, Data_Rule_Type >, 220
ord
Node, 208
out_of_range
BArray< Cell_Type, Data_Type >, 69
BArrayDense< Cell_Type, Data_Type >, 91
params_last
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 196
parent
Node, 209
parse_polytomies
Flock, 155
Geese, 169
Phylo counters, 47
counter_co_opt, 48
counter_cogain, 49
counter_gains, 49
counter_gains_from_0, 49
counter_gains_k_offspring, 49
counter_genes_changing, 50
counter_k_genes_changing, 50
counter_less_than_p_prop_genes_changing, 50
counter_longest, 50
counter_loss, 51
counter_maxfuns, 51
counter_neofun, 51
counter_neofun_a2b, 51
counter_overall_changes, 52
counter_overall_gains, 52
counter_overall_gains_from_0, 52
counter_overall_loss, 52
counter_pairwise_first_gain, 53
counter_pairwise_neofun_singlefun, 53
counter_pairwise_overall_change, 53
counter_pairwise_preserving, 53
counter_preserve_pseudogene, 54
counter_prop_genes_changing, 54
counter_subfun, 54
Phylo rules, 35
~DEFMCounterData, 43
~DEFMData, 43
~DEFMRuleDynData, 43
array, 44
at, 39
counts, 44
covar_sort, 44
covar_used, 44
covariates, 44
DEFMCounter, 37
DEFMCounterData, 39
DEFMCounters, 37
DEFMData, 39
DEFMModel, 37
DEFMRule, 38
DEFMRuleData, 40
DEFMRuleDyn, 38
DEFMRuleDynData, 40
DEFMRules, 38

DEFMRulesDyn, 38
DEFMStatsCounter, 38
DEFMSupport, 38
idx, 40, 41
indices, 44, 45
init, 45
is_motif, 45
is_true, 41
logical, 45
ncol, 41
nrow, 41
num, 41, 42
numbers, 45, 46
obs_start, 46
operator(), 42
print, 42
rule_dyn_limit_changes, 42
X_ncol, 46
X_nrow, 46
PHYLO_CHECK_MISSING
Counting, 16
PHYLO_COUNTER_LAMBDA
Counting, 17
PHYLO_RULE_DYN_LAMBDA
Counting, 17
PhyloArray
geese-types.hpp, 327
PhyloCounter
geese-types.hpp, 327
PhyloCounterData, 211
at, 212
begin, 212
empty, 212
end, 213
get_counters, 213
operator(), 213
operator[], 213
PhyloCounterData, 212
push_back, 213
reserve, 213
shrink_to_fit, 214
size, 214
PhyloCounters
geese-types.hpp, 327
PhyloModel
geese-types.hpp, 327
PhyloPowerSet
geese-types.hpp, 327
PhyloRule
geese-types.hpp, 328
PhyloRuleData
geese-types.hpp, 328
PhyloRuleDyn
geese-types.hpp, 328
PhyloRuleDynData, 214
~PhyloRuleDynData, 215
counts, 215
duplication, 215
lb, 216
operator(), 215
PhyloRuleDynData, 215
pos, 216
ub, 216
PhyloRules
geese-types.hpp, 328
PhyloRulesDyn
geese-types.hpp, 328
PhyloStatsCounter
geese-types.hpp, 328
PhyloSupport
geese-types.hpp, 329
POS
barraydense-meat-operators.hpp, 258
barraydense-meat.hpp, 260
barraydensecell-bones.hpp, 262
barraydensecell-meat.hpp, 262
barraydensecol-bones.hpp, 263
barraydenserow-bones.hpp, 265
geese-types.hpp, 326
pos
PhyloRuleDynData, 216
POS_N
barraydense-meat-operators.hpp, 258
barraydense-meat.hpp, 261
barraydensecol-bones.hpp, 263
barraydenserow-bones.hpp, 265
PowerSet
PowerSet< Array_Type, Data_Rule_Type >, 218
PowerSet< Array_Type, Data_Rule_Type >, 216
~PowerSet, 218
add_rule, 218, 219
begin, 219
calc, 219
coordinates_free, 221
coordinates_locked, 221
data, 221
EmptyArray, 221
end, 219
get_data, 219
get_data_ptr, 220
init_support, 220
M, 221
N, 222
n_free, 222
n_locked, 222
operator[], 220
PowerSet, 218
reset, 220
rules, 222
rules_deleted, 222
size, 220
predict
Geese, 169
predict_backend
Geese, 170
predict_exhaust

Geese, 170
predict_exhaust_backend
Geese, 170
predict_sim
Geese, 170
print
BArray< Cell_Type, Data_Type >, 69
BArrayDense< Cell_Type, Data_Type >, 91
DEFM, 143
Flock, 155
FreqTable< T >, 159
Geese, 170
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 189
Phylo rules, 42
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 242
print_n
BArray< Cell_Type, Data_Type >, 70
print_nodes
Geese, 171
print_observed_counts
Geese, 171
print_stats
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 189
printf_barry
barry-configuration.hpp, 269
probability
Node, 209
Progress, 223
~Progress, 223
end, 224
next, 224
Progress, 223
progress.hpp
BARRY_PROGRESS_BAR_WIDTH, 331
pset_arrays
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 196
pset_loc
Geese, 173
pset_loop
geese-meat-likelihood.hpp, 321
pset_probs
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 196
pset_stats
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 196
push_back
PhyloCounterData, 213
README.md, 345
reduced_sequence
Geese, 174
rengine
Flock, 156
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 197
res
counters-meat.hpp, 288
reserve
BArray< Cell_Type, Data_Type >, 70
BArrayDense< Cell_Type, Data_Type >, 91
FreqTable< T >, 159
PhyloCounterData, 213
reset
PowerSet< Array_Type, Data_Rule_Type >, 220
reset_array
StatsCounter< Array_Type, Data_Type >, 234
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 242
resize
BArray< Cell_Type, Data_Type >, 70
BArrayDense< Cell_Type, Data_Type >, 92
Entries< Cell_Type >, 149
statscounter-meat.hpp, 335
return
counters-meat.hpp, 289
statscounter-meat.hpp, 338
rhs
barray-meat-operators.hpp, 253
rm_cell
BArray< Cell_Type, Data_Type >, 70
BArrayDense< Cell_Type, Data_Type >, 92
ROW
barray-meat-operators.hpp, 251
barray-meat.hpp, 255
barraydense-meat-operators.hpp, 258
barraydense-meat.hpp, 261
row
BArray< Cell_Type, Data_Type >, 70
BArrayDense< Cell_Type, Data_Type >, 92
Row_type
typedefs.hpp, 343
rowsum
BArrayDense< Cell_Type, Data_Type >, 92
Rule
Rule< Array_Type, Data_Type >, 225
Rule< Array_Type, Data_Type >, 224
~Rule, 225
D, 226
get_description, 226
get_name, 226
operator(), 227
Rule, 225
rule_dyn_limit_changes
Phylo rules, 42

rule_fun_default
rules-bones.hpp, 331
Rule_fun_type
typedefs.hpp, 343
RULE_FUNCTION
barry.hpp, 275
geese-bones.hpp, 319
RULE_LAMBDA
barry.hpp, 275
Rules
Rules< Array_Type, Data_Type >, 228
rules
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 197
PowerSet< Array_Type, Data_Rule_Type >, 222
Rules< Array_Type, Data_Type >, 227
~Rules, 228
add_rule, 229
begin, 229
end, 229
get_descriptions, 229
get_names, 230
get_seq, 230
operator(), 230
operator=, 231
Rules, 228
size, 231
rules-bones.hpp
rule_fun_default, 331
rules_deleted
PowerSet< Array_Type, Data_Rule_Type >, 222
rules_dont_become_zero
Network counters, 34
rules_dyn
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 197
rules_markov_fixed
Network counters, 35
rules_zerodiag
network.hpp, 304

sample
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 190
sequence
Geese, 174
set_counters
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 190
StatsCounter< Array_Type, Data_Type >, 235
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 242
set_data
BArray< Cell_Type, Data_Type >, 71
BArrayDense< Cell_Type, Data_Type >, 93
set_hasher
Counter< Array_Type, Data_Type >, 132
set_names
DEFM, 143
set_rengine
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 190
set_rules
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 191
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 243
set_rules_dyn
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 191
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 243
set_seed
Flock, 155
Geese, 171
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 191
set_transform_model
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 191
shrink_to_fit
PhyloCounterData, 214
simulate
DEFM, 143
Geese, 171
size
BArrayDenseCol< Cell_Type, Data_Type >, 101
BArrayDenseCol_const< Cell_Type, Data_Type >, 104
BArrayDenseRow< Cell_Type, Data_Type >, 106
BArrayDenseRow_const< Cell_Type, Data_Type >, 108
BArrayVector< Cell_Type, Data_Type >, 117
BArrayVector_const< Cell_Type, Data_Type >, 121
Counters< Array_Type, Data_Type >, 138
FreqTable< T >, 160
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 192
PhyloCounterData, 214
PowerSet< Array_Type, Data_Rule_Type >, 220
Rules< Array_Type, Data_Type >, 231
StatsCounter< Array_Type, Data_Type >, 235
size_unique

Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 192
sort_array
typedefs.hpp, 343
source
Entries< Cell_Type >, 149
states
NodeData, 211
Statistical Models, 24
stats_likelihood
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 197
stats_support
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 197
stats_support_n_arrays
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 198
stats_support_sizes
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 198
stats_support_sizes_acc
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 198
stats_target
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 198
StatsCounter
StatsCounter< Array_Type, Data_Type >, 232, 233
StatsCounter< Array_Type, Data_Type >, 231
~StatsCounter, 233
add_counter, 233
count_all, 233
count_current, 234
count_init, 234
get_counters, 234
get_descriptions, 234
get_names, 234
reset_array, 234
set_counters, 235
size, 235
StatsCounter, 232, 233
statscounter-meat.hpp
clear, 334
counter, 336
counter_deleted, 337
counters, 337
counters_, 337
current_stats, 337
EmptyArray, 337
f_, 338
for, 334
j, 338
resize, 335
return, 338
STATSCOUNTER_TEMPLATE, 334–336
STATSCOUNTER_TEMPLATE_ARGS, 334
STATSCOUNTER_TYPE, 334
STATSCOUNTER_TEMPLATE
statscounter-meat.hpp, 334–336
STATSCOUNTER_TEMPLATE_ARGS
statscounter-meat.hpp, 334
STATSCOUNTER_TYPE
statscounter-meat.hpp, 334
store_psets
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 192
subtree_prob
Node, 209
Support
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 237, 238
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 235
~Support, 238
add_counter, 238
add_rule, 238, 239
add_rule_dyn, 239
calc, 239
change_stats, 243
coordiantes_n_free, 243
coordiantes_n_locked, 244
coordinates_free, 244
coordinates_locked, 244
current_stats, 244
delete_counters, 244
delete_rules, 245
delete_rules_dyn, 245
eval_rules_dyn, 240
get_counters, 240
get_counts, 240
get_current_stats, 240
get_data, 241
get_rules, 241
get_rules_dyn, 241
hashes, 245
hashes_initialized, 245
init_support, 241
M, 245
max_num_elements, 246
N, 246
n_counters, 246
print, 242
reset_array, 242
set_counters, 242
set_rules, 243

set_rules_dyn, 243
Support, 237, 238
support-meat.hpp
BARRY_SUPPORT_MEAT_HPP, 339
support_fun
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 199
support_size
Flock, 155
Geese, 171
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 192
swap_cells
BArray< Cell_Type, Data_Type >, 71
BArrayDense< Cell_Type, Data_Type >, 93
swap_cols
BArray< Cell_Type, Data_Type >, 71
BArrayDense< Cell_Type, Data_Type >, 93
swap_rows
BArray< Cell_Type, Data_Type >, 71
BArrayDense< Cell_Type, Data_Type >, 93

target
Entries< Cell_Type >, 149
this
barray-meat-operators.hpp, 254
TMP_HASHER_CALL
counters-meat.hpp, 279
toggle_cell
BArray< Cell_Type, Data_Type >, 72
BArrayDense< Cell_Type, Data_Type >, 94
toggle_lock
BArray< Cell_Type, Data_Type >, 72
BArrayDense< Cell_Type, Data_Type >, 94
transform_model
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 192
transform_model_fun
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 199
transform_model_term_names
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 199
transpose
BArray< Cell_Type, Data_Type >, 72
BArrayDense< Cell_Type, Data_Type >, 94
TWO
CHECK, 56
EXISTS, 58
typedefs.hpp
Col_type, 342
Counter_fun_type, 342
Counts_type, 342
Hasher_fun_type, 342
MapVec_type, 343
Row_type, 343
Rule_fun_type, 343
sort_array, 343
vec_equal, 344
vec_equal_approx, 344
vec_inner_prod, 344, 345

ub
PhyloRuleDynData, 216
UKNOWN
EXISTS, 58
UNI_SUB
counters.hpp, 310
update_annotations
Geese, 171
update_likelihoods
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 193
update_normalizing_constant
model-meat.hpp, 306
update_normalizing_constants
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 193
update_pset_probs
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 193

val
Entries< Cell_Type >, 149
value
Cell< Cell_Type >, 125
vec_diff
geese-bones.hpp, 319
vec_equal
typedefs.hpp, 344
vec_equal_approx
typedefs.hpp, 344
vec_inner_prod
typedefs.hpp, 344, 345
vecHasher< T >, 246
operator(), 247
vector_caster
geese-bones.hpp, 320
vertex_attr
NetworkData, 203
visited
BArray< Cell_Type, Data_Type >, 73
BArrayDense< Cell_Type, Data_Type >, 96
Cell< Cell_Type >, 125
Node, 209

with_pset
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 200

X_ncol
Phylo rules, 46
X_nrow
Phylo rules, 46

ZERO_CELL
barraydense-meat.hpp, 261
barraydensecol-bones.hpp, 264
barraydenserow-bones.hpp, 265
zero_col
BArray< Cell_Type, Data_Type >, 72
BArrayDense< Cell_Type, Data_Type >, 94
zero_row
BArray< Cell_Type, Data_Type >, 72
BArrayDense< Cell_Type, Data_Type >, 94