# 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 Class Index** 7

3.1 Class List . . . 7

**4 File Index** 9

4.1 File List . . . 9

**5 Module Documentation** 11

5.1 Counting . . . 11

5.1.1 Detailed Description . . . 11

5.2 Statistical Models . . . 11

5.2.1 Detailed Description . . . 12

5.3 Network counters . . . 12

5.3.1 Detailed Description . . . 13

5.3.2 Function Documentation . . . 13

5.3.2.1 counter_absdiff() . . . 13

5.3.2.2 counter_ctriads() [1/2] . . . 14

5.3.2.3 counter_ctriads() [2/2] . . . 14

5.3.2.4 counter_degree() . . . 14

5.3.2.5 counter_density() . . . 14

5.3.2.6 counter_diff() . . . 14

5.3.2.7 counter_edges() . . . 15

5.3.2.8 counter_idegree() [1/2] . . . 15

5.3.2.9 counter_idegree() [2/2] . . . 15

5.3.2.10 counter_idegree15() [1/2] . . . 15

5.3.2.11 counter_idegree15() [2/2] . . . 15

5.3.2.12 counter_isolates() [1/2] . . . 16

5.3.2.13 counter_isolates() [2/2] . . . 16

5.3.2.14 counter_istar2() [1/2] . . . 16

5.3.2.15 counter_istar2() [2/2] . . . 16

5.3.2.16 counter_mutual() . . . 16

5.3.2.17 counter_nodecov() . . . 17

5.3.2.18 counter_nodeicov() . . . 17

5.3.2.19 counter_nodematch() . . . 17

5.3.2.20 counter_nodeocov() . . . 17

5.3.2.21 counter_odegree() [1/2] . . . 17

5.3.2.22 counter_odegree() [2/2] . . . 18

5.3.2.23 counter_odegree15() [1/2] . . . 18

5.3.2.24 counter_odegree15() [2/2] . . . 18

5.3.2.25 counter_ostar2() [1/2] . . . 18

5.3.2.26 counter_ostar2() [2/2] . . . 18
5.3.2.27 counter_ttriads() [1/2] . . . 19
5.3.2.28 counter_ttriads() [2/2] . . . 19
5.3.2.29 NETWORK_COUNTER() . . . 19
5.4 Phylo counters . . . 19
5.4.1 Detailed Description . . . 20
5.4.2 Function Documentation . . . 20
5.4.2.1 counter_co_opt() . . . 21
5.4.2.2 counter_cogain() . . . 21
5.4.2.3 counter_gains() . . . 21
5.4.2.4 counter_gains_from_0() . . . 22
5.4.2.5 counter_gains_k_offspring() . . . 22
5.4.2.6 counter_genes_changing() . . . 22
5.4.2.7 counter_k_genes_changing() . . . 22
5.4.2.8 counter_less_than_p_prop_genes_changing() . . . 23
5.4.2.9 counter_longest() . . . 23
5.4.2.10 counter_loss() . . . 23
5.4.2.11 counter_maxfuns() . . . 23
5.4.2.12 counter_neofun() . . . 24
5.4.2.13 counter_neofun_a2b() . . . 24
5.4.2.14 counter_overall_changes() . . . 24
5.4.2.15 counter_overall_gains() . . . 24
5.4.2.16 counter_overall_gains_from_0() . . . 25
5.4.2.17 counter_overall_loss() . . . 25
5.4.2.18 counter_pairwise_first_gain() . . . 25
5.4.2.19 counter_pairwise_overall_change() . . . 25
5.4.2.20 counter_pairwise_preserving() . . . 26
5.4.2.21 counter_preserve_pseudogene() . . . 26
5.4.2.22 counter_prop_genes_changing() . . . 26
5.4.2.23 counter_subfun() . . . 26
5.5 Phylo rules . . . 27
5.5.1 Detailed Description . . . 27
5.5.2 Function Documentation . . . 27
5.5.2.1 rule_dyn_limit_changes() . . . 27

**6 Namespace Documentation** **29**
6.1 barry Namespace Reference . . . 29
6.1.1 Detailed Description . . . 29
6.2 barry::counters Namespace Reference . . . 29
6.2.1 Detailed Description . . . 29
6.3 barry::counters::network Namespace Reference . . . 30
6.4 barry::counters::phylo Namespace Reference . . . 30

6.5 CHECK Namespace Reference . . . 30
6.5.1 Detailed Description . . . 30
6.5.2 Variable Documentation . . . 30
6.5.2.1 BOTH . . . 30
6.5.2.2 NONE . . . 30
6.5.2.3 ONE . . . 30
6.5.2.4 TWO . . . 31
6.6 EXISTS Namespace Reference . . . 31
6.6.1 Detailed Description . . . 31
6.6.2 Variable Documentation . . . 31
6.6.2.1 AS_ONE . . . 31
6.6.2.2 AS_ZERO . . . 31
6.6.2.3 BOTH . . . 32
6.6.2.4 NONE . . . 32
6.6.2.5 ONE . . . 32
6.6.2.6 TWO . . . 32
6.6.2.7 UKNOWN . . . 32

**7 Class Documentation** **33**

7.1 BArray< Cell_Type, Data_Type > Class Template Reference . . . 33
7.1.1 Detailed Description . . . 35
7.1.2 Constructor & Destructor Documentation . . . 36
7.1.2.1 BArray() [1/6] . . . 36
7.1.2.2 BArray() [2/6] . . . 36
7.1.2.3 BArray() [3/6] . . . 36
7.1.2.4 BArray() [4/6] . . . 37
7.1.2.5 BArray() [5/6] . . . 37
7.1.2.6 BArray() [6/6] . . . 37
7.1.2.7 ~BArray() . . . 37
7.1.3 Member Function Documentation . . . 37
7.1.3.1 clear() . . . 37
7.1.3.2 col() . . . 38
7.1.3.3 D() [1/2] . . . 38
7.1.3.4 D() [2/2] . . . 38
7.1.3.5 default_val() . . . 38
7.1.3.6 flush_data() . . . 38
7.1.3.7 get_cell() . . . 38
7.1.3.8 get_col_vec() [1/2] . . . 39
7.1.3.9 get_col_vec() [2/2] . . . 39
7.1.3.10 get_entries() . . . 39
7.1.3.11 get_row_vec() [1/2] . . . 39
7.1.3.12 get_row_vec() [2/2] . . . 39

7.1.3.13 insert_cell() [1/3] . . . 40
7.1.3.14 insert_cell() [2/3] . . . 40
7.1.3.15 insert_cell() [3/3] . . . 40
7.1.3.16 is_dense() . . . 40
7.1.3.17 is_empty() . . . 40
7.1.3.18 ncol() . . . 41
7.1.3.19 nnozero() . . . 41
7.1.3.20 nrow() . . . 41
7.1.3.21 operator()() [1/2] . . . 41
7.1.3.22 operator()() [2/2] . . . 41
7.1.3.23 operator*=() . . . 41
7.1.3.24 operator+=() [1/3] . . . 42
7.1.3.25 operator+=() [2/3] . . . 42
7.1.3.26 operator+=() [3/3] . . . 42
7.1.3.27 operator-=() [1/3] . . . 42
7.1.3.28 operator-=() [2/3] . . . 42
7.1.3.29 operator-=() [3/3] . . . 42
7.1.3.30 operator/=() . . . 43
7.1.3.31 operator=() [1/2] . . . 43
7.1.3.32 operator=() [2/2] . . . 43
7.1.3.33 operator==() . . . 43
7.1.3.34 out_of_range() . . . 43
7.1.3.35 print() . . . 43
7.1.3.36 reserve() . . . 44
7.1.3.37 resize() . . . 44
7.1.3.38 rm_cell() . . . 44
7.1.3.39 row() . . . 44
7.1.3.40 set_data() . . . 44
7.1.3.41 swap_cells() . . . 45
7.1.3.42 swap_cols() . . . 45
7.1.3.43 swap_rows() . . . 45
7.1.3.44 toggle_cell() . . . 45
7.1.3.45 toggle_lock() . . . 46
7.1.3.46 transpose() . . . 46
7.1.3.47 zero_col() . . . 46
7.1.3.48 zero_row() . . . 46
7.1.4 Friends And Related Function Documentation . . . 46
7.1.4.1 BArrayCell< Cell_Type, Data_Type > . . . 46
7.1.4.2 BArrayCell_const< Cell_Type, Data_Type > . . . 47
7.1.5 Member Data Documentation . . . 47
7.1.5.1 visited . . . 47
7.2 BArrayCell< Cell_Type, Data_Type > Class Template Reference . . . 47

7.2.1 Detailed Description . . . 47
7.2.2 Constructor & Destructor Documentation . . . 48
7.2.2.1 BArrayCell() . . . 48
7.2.2.2 ∼BArrayCell() . . . 48
7.2.3 Member Function Documentation . . . 48
7.2.3.1 operator Cell_Type() . . . 48
7.2.3.2 operator∗=() . . . 48
7.2.3.3 operator+=() . . . 49
7.2.3.4 operator-=() . . . 49
7.2.3.5 operator/=() . . . 49
7.2.3.6 operator=() . . . 49
7.2.3.7 operator==() . . . 49
7.3 BArrayCell_const< Cell_Type, Data_Type > Class Template Reference . . . 50
7.3.1 Detailed Description . . . 50
7.3.2 Constructor & Destructor Documentation . . . 50
7.3.2.1 BArrayCell_const() . . . 50
7.3.2.2 ∼BArrayCell_const() . . . 50
7.3.3 Member Function Documentation . . . 51
7.3.3.1 operator Cell_Type() . . . 51
7.3.3.2 operator"!=() . . . 51
7.3.3.3 operator<() . . . 51
7.3.3.4 operator<=() . . . 51
7.3.3.5 operator==() . . . 51
7.3.3.6 operator>() . . . 52
7.3.3.7 operator>=() . . . 52
7.4 BArrayDense< Cell_Type, Data_Type > Class Template Reference . . . 52
7.4.1 Detailed Description . . . 54
7.4.2 Constructor & Destructor Documentation . . . 55
7.4.2.1 BArrayDense() [1/6] . . . 55
7.4.2.2 BArrayDense() [2/6] . . . 55
7.4.2.3 BArrayDense() [3/6] . . . 55
7.4.2.4 BArrayDense() [4/6] . . . 56
7.4.2.5 BArrayDense() [5/6] . . . 56
7.4.2.6 BArrayDense() [6/6] . . . 56
7.4.2.7 ∼BArrayDense() . . . 56
7.4.3 Member Function Documentation . . . 56
7.4.3.1 clear() . . . 56
7.4.3.2 col() [1/2] . . . 57
7.4.3.3 col() [2/2] . . . 57
7.4.3.4 colsum() . . . 57
7.4.3.5 D() [1/2] . . . 57
7.4.3.6 D() [2/2] . . . 57

7.4.3.7 default_val() . . . 57
7.4.3.8 get_cell() . . . 58
7.4.3.9 get_col_vec() [1/2] . . . 58
7.4.3.10 get_col_vec() [2/2] . . . 58
7.4.3.11 get_data() . . . 58
7.4.3.12 get_entries() . . . 58
7.4.3.13 get_row_vec() [1/2] . . . 59
7.4.3.14 get_row_vec() [2/2] . . . 59
7.4.3.15 insert_cell() [1/2] . . . 59
7.4.3.16 insert_cell() [2/2] . . . 59
7.4.3.17 is_dense() . . . 59
7.4.3.18 is_empty() . . . 60
7.4.3.19 ncol() . . . 60
7.4.3.20 nnozero() . . . 60
7.4.3.21 nrow() . . . 60
7.4.3.22 operator()() [1/2] . . . 60
7.4.3.23 operator()() [2/2] . . . 60
7.4.3.24 operator∗=() . . . 61
7.4.3.25 operator+=() [1/3] . . . 61
7.4.3.26 operator+=() [2/3] . . . 61
7.4.3.27 operator+=() [3/3] . . . 61
7.4.3.28 operator-=() [1/3] . . . 61
7.4.3.29 operator-=() [2/3] . . . 61
7.4.3.30 operator-=() [3/3] . . . 62
7.4.3.31 operator/=() . . . 62
7.4.3.32 operator=() [1/2] . . . 62
7.4.3.33 operator=() [2/2] . . . 62
7.4.3.34 operator==() . . . 62
7.4.3.35 out_of_range() . . . 62
7.4.3.36 print() . . . 63
7.4.3.37 reserve() . . . 63
7.4.3.38 resize() . . . 63
7.4.3.39 rm_cell() . . . 63
7.4.3.40 row() [1/2] . . . 63
7.4.3.41 row() [2/2] . . . 63
7.4.3.42 rowsum() . . . 64
7.4.3.43 set_data() . . . 64
7.4.3.44 swap_cells() . . . 64
7.4.3.45 swap_cols() . . . 64
7.4.3.46 swap_rows() . . . 65
7.4.3.47 toggle_cell() . . . 65
7.4.3.48 toggle_lock() . . . 65

7.4.3.49 transpose() . . . 65
7.4.3.50 zero_col() . . . 65
7.4.3.51 zero_row() . . . 66
7.4.4 Friends And Related Function Documentation . . . 66
7.4.4.1 BArrayDenseCell< Cell_Type, Data_Type > . . . 66
7.4.4.2 BArrayDenseCol< Cell_Type, Data_Type > . . . 66
7.4.4.3 BArrayDenseCol_const< Cell_Type, Data_Type > . . . 66
7.4.4.4 BArrayDenseRow< Cell_Type, Data_Type > . . . 66
7.4.4.5 BArrayDenseRow_const< Cell_Type, Data_Type > . . . 67
7.4.5 Member Data Documentation . . . 67
7.4.5.1 visited . . . 67
7.5 BArrayDenseCell< Cell_Type, Data_Type > Class Template Reference . . . 67
7.5.1 Detailed Description . . . 68
7.5.2 Constructor & Destructor Documentation . . . 68
7.5.2.1 BArrayDenseCell() . . . 68
7.5.2.2 ~BArrayDenseCell() . . . 68
7.5.3 Member Function Documentation . . . 68
7.5.3.1 operator Cell_Type() . . . 68
7.5.3.2 operator*=() . . . 69
7.5.3.3 operator+=() . . . 69
7.5.3.4 operator-=() . . . 69
7.5.3.5 operator/=() . . . 69
7.5.3.6 operator=() . . . 69
7.5.3.7 operator==() . . . 70
7.5.4 Friends And Related Function Documentation . . . 70
7.5.4.1 BArrayDense< Cell_Type, Data_Type > . . . 70
7.5.4.2 BArrayDenseCol< Cell_Type, Data_Type > . . . 70
7.5.4.3 BArrayDenseCol_const< Cell_Type, Data_Type > . . . 70
7.6 BArrayDenseCell_const< Cell_Type, Data_Type > Class Template Reference . . . 71
7.6.1 Detailed Description . . . 71
7.7 BArrayDenseCol< Cell_Type, Data_Type > Class Template Reference . . . 71
7.7.1 Detailed Description . . . 71
7.7.2 Constructor & Destructor Documentation . . . 71
7.7.2.1 BArrayDenseCol() . . . 72
7.7.3 Member Function Documentation . . . 72
7.7.3.1 begin() . . . 72
7.7.3.2 end() . . . 72
7.7.3.3 operator()() . . . 72
7.7.3.4 size() . . . 72
7.7.4 Friends And Related Function Documentation . . . 73
7.7.4.1 BArrayDense< Cell_Type, Data_Type > . . . 73
7.7.4.2 BArrayDenseCell< Cell_Type, Data_Type > . . . 73

7.7.4.3 BArrayDenseCell_const< Cell_Type, Data_Type > . . . 73
7.8 BArrayDenseCol_const< Cell_Type, Data_Type > Class Template Reference . . . 73
7.8.1 Detailed Description . . . 74
7.8.2 Constructor & Destructor Documentation . . . 74
7.8.2.1 BArrayDenseCol_const() . . . 74
7.8.3 Member Function Documentation . . . 74
7.8.3.1 begin() . . . 74
7.8.3.2 end() . . . 74
7.8.3.3 operator()() . . . 75
7.8.3.4 size() . . . 75
7.8.4 Friends And Related Function Documentation . . . 75
7.8.4.1 BArrayDenseCell< Cell_Type, Data_Type > . . . 75
7.8.4.2 BArrayDenseCell_const< Cell_Type, Data_Type > . . . 75
7.9 BArrayDenseRow< Cell_Type, Data_Type > Class Template Reference . . . 75
7.9.1 Detailed Description . . . 76
7.9.2 Constructor & Destructor Documentation . . . 76
7.9.2.1 BArrayDenseRow() . . . 76
7.9.3 Member Function Documentation . . . 76
7.9.3.1 begin() . . . 76
7.9.3.2 end() . . . 77
7.9.3.3 operator()() . . . 77
7.9.3.4 size() . . . 77
7.9.4 Friends And Related Function Documentation . . . 77
7.9.4.1 BArrayDense< Cell_Type, Data_Type > . . . 77
7.9.4.2 BArrayDenseCell< Cell_Type, Data_Type > . . . 77
7.9.4.3 BArrayDenseCell_const< Cell_Type, Data_Type > . . . 78
7.10 BArrayDenseRow_const< Cell_Type, Data_Type > Class Template Reference . . . 78
7.10.1 Detailed Description . . . 78
7.10.2 Constructor & Destructor Documentation . . . 78
7.10.2.1 BArrayDenseRow_const() . . . 79
7.10.3 Member Function Documentation . . . 79
7.10.3.1 begin() . . . 79
7.10.3.2 end() . . . 79
7.10.3.3 operator()() . . . 79
7.10.3.4 size() . . . 79
7.10.4 Friends And Related Function Documentation . . . 80
7.10.4.1 BArrayDenseCell< Cell_Type, Data_Type > . . . 80
7.10.4.2 BArrayDenseCell_const< Cell_Type, Data_Type > . . . 80
7.11 BArrayRow< Cell_Type, Data_Type > Class Template Reference . . . 80
7.11.1 Detailed Description . . . 80
7.11.2 Constructor & Destructor Documentation . . . 81
7.11.2.1 BArrayRow() . . . 81

7.11.2.2 ~BArrayRow() . . . 81
7.11.3 Member Function Documentation . . . 81
7.11.3.1 operator BArrayRow< Cell_Type, Data_Type >() . . . 81
7.11.3.2 operator*=() . . . 81
7.11.3.3 operator+=() . . . 81
7.11.3.4 operator-=() . . . 82
7.11.3.5 operator/=() . . . 82
7.11.3.6 operator=() . . . 82
7.11.3.7 operator==() . . . 82
7.12 BArrayRow_const< Cell_Type, Data_Type > Class Template Reference . . . 82
7.12.1 Detailed Description . . . 83
7.12.2 Constructor & Destructor Documentation . . . 83
7.12.2.1 BArrayRow_const() . . . 83
7.12.2.2 ~BArrayRow_const() . . . 83
7.12.3 Member Function Documentation . . . 83
7.12.3.1 operator BArrayRow_const< Cell_Type, Data_Type >() . . . 83
7.12.3.2 operator"!=() . . . 83
7.12.3.3 operator<() . . . 84
7.12.3.4 operator<=() . . . 84
7.12.3.5 operator==() . . . 84
7.12.3.6 operator>() . . . 84
7.12.3.7 operator>=() . . . 84
7.13 BArrayVector< Cell_Type, Data_Type > Class Template Reference . . . 84
7.13.1 Detailed Description . . . 85
7.13.2 Constructor & Destructor Documentation . . . 85
7.13.2.1 BArrayVector() . . . 85
7.13.2.2 ~BArrayVector() . . . 86
7.13.3 Member Function Documentation . . . 86
7.13.3.1 begin() . . . 86
7.13.3.2 end() . . . 86
7.13.3.3 is_col() . . . 86
7.13.3.4 is_row() . . . 87
7.13.3.5 operator std::vector< Cell_Type >() . . . 87
7.13.3.6 operator*=() . . . 87
7.13.3.7 operator+=() . . . 87
7.13.3.8 operator-=() . . . 87
7.13.3.9 operator/=() . . . 88
7.13.3.10 operator=() . . . 88
7.13.3.11 operator==() . . . 88
7.13.3.12 size() . . . 88
7.14 BArrayVector_const< Cell_Type, Data_Type > Class Template Reference . . . 88
7.14.1 Detailed Description . . . 89

7.14.2 Constructor & Destructor Documentation . . . 89
7.14.2.1 BArrayVector_const() . . . 89
7.14.2.2 ~BArrayVector_const() . . . 89
7.14.3 Member Function Documentation . . . 90
7.14.3.1 begin() . . . 90
7.14.3.2 end() . . . 90
7.14.3.3 is_col() . . . 90
7.14.3.4 is_row() . . . 90
7.14.3.5 operator std::vector< Cell_Type >() . . . 90
7.14.3.6 operator"!=() . . . 91
7.14.3.7 operator<() . . . 91
7.14.3.8 operator<=() . . . 91
7.14.3.9 operator==() . . . 91
7.14.3.10 operator>() . . . 91
7.14.3.11 operator>=() . . . 92
7.14.3.12 size() . . . 92
7.15 Cell< Cell_Type > Class Template Reference . . . 92
7.15.1 Detailed Description . . . 93
7.15.2 Constructor & Destructor Documentation . . . 93
7.15.2.1 Cell() [1/7] . . . 93
7.15.2.2 Cell() [2/7] . . . 93
7.15.2.3 ~Cell() . . . 93
7.15.2.4 Cell() [3/7] . . . 94
7.15.2.5 Cell() [4/7] . . . 94
7.15.2.6 Cell() [5/7] . . . 94
7.15.2.7 Cell() [6/7] . . . 94
7.15.2.8 Cell() [7/7] . . . 94
7.15.3 Member Function Documentation . . . 94
7.15.3.1 add() [1/4] . . . 95
7.15.3.2 add() [2/4] . . . 95
7.15.3.3 add() [3/4] . . . 95
7.15.3.4 add() [4/4] . . . 95
7.15.3.5 operator Cell_Type() . . . 95
7.15.3.6 operator"!=() . . . 95
7.15.3.7 operator=() [1/2] . . . 96
7.15.3.8 operator=() [2/2] . . . 96
7.15.3.9 operator==() . . . 96
7.15.4 Member Data Documentation . . . 96
7.15.4.1 active . . . 96
7.15.4.2 value . . . 96
7.15.4.3 visited . . . 97
7.16 Cell_const< Cell_Type > Class Template Reference . . . 97

7.16.1 Detailed Description . . . 97
7.17 ConstBArrayRowIter< Cell_Type, Data_Type > Class Template Reference . . . 97
7.17.1 Detailed Description . . . 98
7.17.2 Constructor & Destructor Documentation . . . 98
7.17.2.1 ConstBArrayRowIter() . . . 98
7.17.2.2 ~ConstBArrayRowIter() . . . 98
7.17.3 Member Data Documentation . . . 98
7.17.3.1 Array . . . 99
7.17.3.2 current_col . . . 99
7.17.3.3 current_row . . . 99
7.17.3.4 iter . . . 99
7.18 Counter< Array_Type, Data_Type > Class Template Reference . . . 99
7.18.1 Detailed Description . . . 100
7.18.2 Constructor & Destructor Documentation . . . 100
7.18.2.1 Counter() [1/4] . . . 101
7.18.2.2 Counter() [2/4] . . . 101
7.18.2.3 Counter() [3/4] . . . 101
7.18.2.4 Counter() [4/4] . . . 101
7.18.2.5 ~Counter() . . . 101
7.18.3 Member Function Documentation . . . 102
7.18.3.1 count() . . . 102
7.18.3.2 get_description() . . . 102
7.18.3.3 get_name() . . . 102
7.18.3.4 init() . . . 102
7.18.3.5 operator=() [1/2] . . . 102
7.18.3.6 operator=() [2/2] . . . 103
7.18.4 Member Data Documentation . . . 103
7.18.4.1 count_fun . . . 103
7.18.4.2 data . . . 103
7.18.4.3 delete_data . . . 103
7.18.4.4 desc . . . 103
7.18.4.5 init_fun . . . 104
7.18.4.6 name . . . 104
7.19 Counters< Array_Type, Data_Type > Class Template Reference . . . 104
7.19.1 Detailed Description . . . 105
7.19.2 Constructor & Destructor Documentation . . . 105
7.19.2.1 Counters() [1/3] . . . 105
7.19.2.2 ~Counters() . . . 105
7.19.2.3 Counters() [2/3] . . . 105
7.19.2.4 Counters() [3/3] . . . 106
7.19.3 Member Function Documentation . . . 106
7.19.3.1 add_counter() [1/3] . . . 106

7.19.3.2 add_counter() [2/3] . . . 106
7.19.3.3 add_counter() [3/3] . . . 106
7.19.3.4 clear() . . . 107
7.19.3.5 get_descriptions() . . . 107
7.19.3.6 get_names() . . . 107
7.19.3.7 operator=() [1/2] . . . 107
7.19.3.8 operator=() [2/2] . . . 107
7.19.3.9 operator[]() . . . 108
7.19.3.10 size() . . . 108
7.20 Entries< Cell_Type > Class Template Reference . . . 109
7.20.1 Detailed Description . . . 109
7.20.2 Constructor & Destructor Documentation . . . 109
7.20.2.1 Entries() [1/2] . . . 109
7.20.2.2 Entries() [2/2] . . . 110
7.20.2.3 ~Entries() . . . 110
7.20.3 Member Function Documentation . . . 110
7.20.3.1 resize() . . . 110
7.20.4 Member Data Documentation . . . 110
7.20.4.1 source . . . 110
7.20.4.2 target . . . 110
7.20.4.3 val . . . 111
7.21 Flock Class Reference . . . 111
7.21.1 Detailed Description . . . 112
7.21.2 Constructor & Destructor Documentation . . . 112
7.21.2.1 Flock() . . . 112
7.21.2.2 ~Flock() . . . 112
7.21.3 Member Function Documentation . . . 112
7.21.3.1 add_data() . . . 112
7.21.3.2 colnames() . . . 113
7.21.3.3 get_counters() . . . 113
7.21.3.4 get_model() . . . 113
7.21.3.5 get_stats_support() . . . 113
7.21.3.6 get_stats_target() . . . 114
7.21.3.7 get_support_fun() . . . 114
7.21.3.8 init() . . . 114
7.21.3.9 likelihood_joint() . . . 114
7.21.3.10 nfuns() . . . 115
7.21.3.11 nleafs() . . . 115
7.21.3.12 nnodes() . . . 115
7.21.3.13 nterms() . . . 115
7.21.3.14 ntrees() . . . 115
7.21.3.15 operator()() . . . 115

7.21.3.16 parse_polytomies() . . . 116
7.21.3.17 print() . . . 116
7.21.3.18 set_seed() . . . 116
7.21.3.19 support_size() . . . 117
7.21.4 Member Data Documentation . . . 117
7.21.4.1 dat . . . 117
7.21.4.2 initialized . . . 117
7.21.4.3 model . . . 117
7.21.4.4 nfunctions . . . 117
7.21.4.5 rengine . . . 118
7.22 FreqTable< T > Class Template Reference . . . 118
7.22.1 Detailed Description . . . 118
7.22.2 Constructor & Destructor Documentation . . . 118
7.22.2.1 FreqTable() . . . 119
7.22.2.2 ~FreqTable() . . . 119
7.22.3 Member Function Documentation . . . 119
7.22.3.1 add() . . . 119
7.22.3.2 as_vector() . . . 119
7.22.3.3 clear() . . . 119
7.22.3.4 get_data() . . . 120
7.22.3.5 get_index() . . . 120
7.22.3.6 make_hash() . . . 120
7.22.3.7 print() . . . 120
7.22.3.8 reserve() . . . 120
7.22.3.9 size() . . . 121
7.23 Geese Class Reference . . . 121
7.23.1 Detailed Description . . . 123
7.23.2 Constructor & Destructor Documentation . . . 124
7.23.2.1 Geese() [1/4] . . . 124
7.23.2.2 Geese() [2/4] . . . 124
7.23.2.3 Geese() [3/4] . . . 124
7.23.2.4 Geese() [4/4] . . . 124
7.23.2.5 ~Geese() . . . 124
7.23.3 Member Function Documentation . . . 125
7.23.3.1 calc_reduced_sequence() . . . 125
7.23.3.2 calc_sequence() . . . 125
7.23.3.3 colnames() . . . 125
7.23.3.4 get_annotated_nodes() . . . 125
7.23.3.5 get_counters() . . . 125
7.23.3.6 get_model() . . . 126
7.23.3.7 get_probabilities() . . . 126
7.23.3.8 get_rengine() . . . 126

7.23.3.9 get_states() . . . 126
7.23.3.10 get_support_fun() . . . 126
7.23.3.11 inherit_support() . . . 127
7.23.3.12 init() . . . 127
7.23.3.13 init_node() . . . 127
7.23.3.14 likelihood() . . . 127
7.23.3.15 likelihood_exhaust() . . . 127
7.23.3.16 nannotations() . . . 128
7.23.3.17 nfuns() . . . 128
7.23.3.18 nleafs() . . . 128
7.23.3.19 nnodes() . . . 128
7.23.3.20 nterms() . . . 128
7.23.3.21 observed_counts() . . . 129
7.23.3.22 operator=() [1/2] . . . 129
7.23.3.23 operator=() [2/2] . . . 129
7.23.3.24 parse_polytomies() . . . 129
7.23.3.25 predict() . . . 129
7.23.3.26 predict_backend() . . . 130
7.23.3.27 predict_exhaust() . . . 130
7.23.3.28 predict_exhaust_backend() . . . 130
7.23.3.29 predict_sim() . . . 130
7.23.3.30 print() . . . 130
7.23.3.31 print_observed_counts() . . . 131
7.23.3.32 set_seed() . . . 131
7.23.3.33 simulate() . . . 131
7.23.3.34 support_size() . . . 131
7.23.3.35 update_annotations() . . . 131
7.23.4 Member Data Documentation . . . 131
7.23.4.1 delete_rengine . . . 132
7.23.4.2 delete_support . . . 132
7.23.4.3 initialized . . . 132
7.23.4.4 map_to_nodes . . . 132
7.23.4.5 nfunctions . . . 132
7.23.4.6 nodes . . . 132
7.23.4.7 pset_loc . . . 133
7.23.4.8 reduced_sequence . . . 133
7.23.4.9 sequence . . . 133
7.24 Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type > Class Template Reference . . . 133
7.24.1 Detailed Description . . . 136
7.24.2 Constructor & Destructor Documentation . . . 136
7.24.2.1 Model() [1/3] . . . 136

7.24.2.2 Model() [2/3] . . . 136
7.24.2.3 Model() [3/3] . . . 137
7.24.2.4 ~Model() . . . 137
7.24.3 Member Function Documentation . . . 137
7.24.3.1 add_array() . . . 137
7.24.3.2 add_counter() [1/3] . . . 137
7.24.3.3 add_counter() [2/3] . . . 138
7.24.3.4 add_counter() [3/3] . . . 138
7.24.3.5 add_rule() [1/3] . . . 138
7.24.3.6 add_rule() [2/3] . . . 138
7.24.3.7 add_rule() [3/3] . . . 138
7.24.3.8 add_rule_dyn() [1/3] . . . 139
7.24.3.9 add_rule_dyn() [2/3] . . . 139
7.24.3.10 add_rule_dyn() [3/3] . . . 139
7.24.3.11 colnames() . . . 139
7.24.3.12 conditional_prob() . . . 139
7.24.3.13 gen_key() . . . 140
7.24.3.14 get_arrays2support() . . . 140
7.24.3.15 get_counters() . . . 140
7.24.3.16 get_norm_const() . . . 140
7.24.3.17 get_pset() . . . 141
7.24.3.18 get_pset_arrays() . . . 141
7.24.3.19 get_pset_probs() . . . 141
7.24.3.20 get_pset_stats() [1/2] . . . 141
7.24.3.21 get_pset_stats() [2/2] . . . 141
7.24.3.22 get_rengine() . . . 142
7.24.3.23 get_rules() . . . 142
7.24.3.24 get_rules_dyn() . . . 142
7.24.3.25 get_stats_support() . . . 142
7.24.3.26 get_stats_target() . . . 142
7.24.3.27 get_support_fun() . . . 143
7.24.3.28 likelihood() [1/3] . . . 143
7.24.3.29 likelihood() [2/3] . . . 143
7.24.3.30 likelihood() [3/3] . . . 143
7.24.3.31 likelihood_total() . . . 144
7.24.3.32 nterms() . . . 144
7.24.3.33 operator=() . . . 144
7.24.3.34 print() . . . 144
7.24.3.35 print_stats() . . . 144
7.24.3.36 sample() [1/2] . . . 145
7.24.3.37 sample() [2/2] . . . 145
7.24.3.38 set_counters() . . . 145

7.24.3.39 set_keygen() . . . 145
7.24.3.40 set_rengine() . . . 145
7.24.3.41 set_rules() . . . 146
7.24.3.42 set_rules_dyn() . . . 146
7.24.3.43 set_seed() . . . 146
7.24.3.44 size() . . . 146
7.24.3.45 size_unique() . . . 146
7.24.3.46 store_psets() . . . 147
7.24.3.47 support_size() . . . 147
7.25 NetCounterData Class Reference . . . 147
7.25.1 Detailed Description . . . 147
7.25.2 Constructor & Destructor Documentation . . . 147
7.25.2.1 NetCounterData() [1/2] . . . 148
7.25.2.2 NetCounterData() [2/2] . . . 148
7.25.2.3 ~NetCounterData() . . . 148
7.25.3 Member Data Documentation . . . 148
7.25.3.1 indices . . . 148
7.25.3.2 numbers . . . 148
7.26 NetworkData Class Reference . . . 149
7.26.1 Detailed Description . . . 149
7.26.2 Constructor & Destructor Documentation . . . 149
7.26.2.1 NetworkData() [1/3] . . . 149
7.26.2.2 NetworkData() [2/3] . . . 149
7.26.2.3 NetworkData() [3/3] . . . 150
7.26.2.4 ~NetworkData() . . . 150
7.26.3 Member Data Documentation . . . 150
7.26.3.1 directed . . . 150
7.26.3.2 vertex_attr . . . 151
7.27 Node Class Reference . . . 151
7.27.1 Detailed Description . . . 152
7.27.2 Constructor & Destructor Documentation . . . 152
7.27.2.1 Node() [1/5] . . . 152
7.27.2.2 Node() [2/5] . . . 153
7.27.2.3 Node() [3/5] . . . 153
7.27.2.4 Node() [4/5] . . . 153
7.27.2.5 Node() [5/5] . . . 153
7.27.2.6 ~Node() . . . 153
7.27.3 Member Function Documentation . . . 153
7.27.3.1 get_parent() . . . 154
7.27.3.2 is_leaf() . . . 154
7.27.3.3 noffspring() . . . 154
7.27.4 Member Data Documentation . . . 154

7.27.4.1 annotations . . . 154
7.27.4.2 array . . . 154
7.27.4.3 arrays . . . 155
7.27.4.4 duplication . . . 155
7.27.4.5 id . . . 155
7.27.4.6 narray . . . 155
7.27.4.7 offspring . . . 155
7.27.4.8 ord . . . 156
7.27.4.9 parent . . . 156
7.27.4.10 probability . . . 156
7.27.4.11 subtree_prob . . . 156
7.27.4.12 visited . . . 156
7.28 NodeData Class Reference . . . 157
7.28.1 Detailed Description . . . 157
7.28.2 Constructor & Destructor Documentation . . . 157
7.28.2.1 NodeData() . . . 157
7.28.3 Member Data Documentation . . . 157
7.28.3.1 blengths . . . 158
7.28.3.2 duplication . . . 158
7.28.3.3 states . . . 158
7.29 PhyloCounterData Class Reference . . . 158
7.29.1 Detailed Description . . . 159
7.29.2 Constructor & Destructor Documentation . . . 159
7.29.2.1 PhyloCounterData() . . . 159
7.29.3 Member Function Documentation . . . 159
7.29.3.1 at() . . . 159
7.29.3.2 begin() . . . 159
7.29.3.3 empty() . . . 159
7.29.3.4 end() . . . 160
7.29.3.5 get_counters() . . . 160
7.29.3.6 operator()() . . . 160
7.29.3.7 operator[]() . . . 160
7.29.3.8 push_back() . . . 160
7.29.3.9 reserve() . . . 160
7.29.3.10 shrink_to_fit() . . . 161
7.29.3.11 size() . . . 161
7.30 PhyloRuleDynData Class Reference . . . 161
7.30.1 Detailed Description . . . 161
7.30.2 Constructor & Destructor Documentation . . . 161
7.30.2.1 PhyloRuleDynData() . . . 162
7.30.2.2 ~PhyloRuleDynData() . . . 162
7.30.3 Member Data Documentation . . . 162

7.30.3.1 counts . . . 162
7.30.3.2 duplication . . . 162
7.30.3.3 lb . . . 162
7.30.3.4 pos . . . 163
7.30.3.5 ub . . . 163
7.31 PowerSet< Array_Type, Data_Rule_Type > Class Template Reference . . . 163
7.31.1 Detailed Description . . . 164
7.31.2 Constructor & Destructor Documentation . . . 165
7.31.2.1 PowerSet() [1/3] . . . 165
7.31.2.2 PowerSet() [2/3] . . . 165
7.31.2.3 PowerSet() [3/3] . . . 165
7.31.2.4 ~PowerSet() . . . 165
7.31.3 Member Function Documentation . . . 166
7.31.3.1 add_rule() [1/3] . . . 166
7.31.3.2 add_rule() [2/3] . . . 166
7.31.3.3 add_rule() [3/3] . . . 166
7.31.3.4 begin() . . . 166
7.31.3.5 calc() . . . 166
7.31.3.6 end() . . . 167
7.31.3.7 get_data() . . . 167
7.31.3.8 get_data_ptr() . . . 167
7.31.3.9 init_support() . . . 167
7.31.3.10 operator[]() . . . 167
7.31.3.11 reset() . . . 168
7.31.3.12 size() . . . 168
7.31.4 Member Data Documentation . . . 168
7.31.4.1 coordinates_free . . . 168
7.31.4.2 coordinates_locked . . . 168
7.31.4.3 data . . . 168
7.31.4.4 EmptyArray . . . 169
7.31.4.5 M . . . 169
7.31.4.6 N . . . 169
7.31.4.7 n_free . . . 169
7.31.4.8 n_locked . . . 169
7.31.4.9 rules . . . 169
7.31.4.10 rules_deleted . . . 170
7.32 Progress Class Reference . . . 170
7.32.1 Detailed Description . . . 170
7.32.2 Constructor & Destructor Documentation . . . 170
7.32.2.1 Progress() . . . 170
7.32.2.2 ~Progress() . . . 171
7.32.3 Member Function Documentation . . . 171

7.32.3.1 end() . . . 171
7.32.3.2 next() . . . 171
7.33 Rule< Array_Type, Data_Type > Class Template Reference . . . 171
7.33.1 Detailed Description . . . 172
7.33.2 Constructor & Destructor Documentation . . . 172
7.33.2.1 Rule() [1/2] . . . 172
7.33.2.2 Rule() [2/2] . . . 172
7.33.2.3 ~Rule() . . . 173
7.33.3 Member Function Documentation . . . 173
7.33.3.1 D() . . . 173
7.33.3.2 operator()() . . . 173
7.34 Rules< Array_Type, Data_Type > Class Template Reference . . . 173
7.34.1 Detailed Description . . . 174
7.34.2 Constructor & Destructor Documentation . . . 174
7.34.2.1 Rules() [1/2] . . . 174
7.34.2.2 Rules() [2/2] . . . 174
7.34.2.3 ~Rules() . . . 175
7.34.3 Member Function Documentation . . . 175
7.34.3.1 add_rule() [1/3] . . . 175
7.34.3.2 add_rule() [2/3] . . . 175
7.34.3.3 add_rule() [3/3] . . . 175
7.34.3.4 clear() . . . 175
7.34.3.5 get_seq() . . . 175
7.34.3.6 operator()() . . . 176
7.34.3.7 operator=() . . . 176
7.34.3.8 size() . . . 177
7.35 StatsCounter< Array_Type, Data_Type > Class Template Reference . . . 177
7.35.1 Detailed Description . . . 177
7.35.2 Constructor & Destructor Documentation . . . 178
7.35.2.1 StatsCounter() [1/3] . . . 178
7.35.2.2 StatsCounter() [2/3] . . . 178
7.35.2.3 StatsCounter() [3/3] . . . 178
7.35.2.4 ~StatsCounter() . . . 178
7.35.3 Member Function Documentation . . . 179
7.35.3.1 add_counter() [1/2] . . . 179
7.35.3.2 add_counter() [2/2] . . . 179
7.35.3.3 count_all() . . . 179
7.35.3.4 count_current() . . . 179
7.35.3.5 count_init() . . . 179
7.35.3.6 get_counters() . . . 180
7.35.3.7 get_descriptions() . . . 180
7.35.3.8 get_names() . . . 180

7.35.3.9 reset_array() . . . 180
7.35.3.10 set_counters() . . . 180
7.36 Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type > Class Template Reference . . . 181
7.36.1 Detailed Description . . . 182
7.36.2 Constructor & Destructor Documentation . . . 183
7.36.2.1 Support() [1/3] . . . 183
7.36.2.2 Support() [2/3] . . . 183
7.36.2.3 Support() [3/3] . . . 183
7.36.2.4 ~Support() . . . 183
7.36.3 Member Function Documentation . . . 184
7.36.3.1 add_counter() [1/2] . . . 184
7.36.3.2 add_counter() [2/2] . . . 184
7.36.3.3 add_rule() [1/2] . . . 184
7.36.3.4 add_rule() [2/2] . . . 184
7.36.3.5 add_rule_dyn() [1/2] . . . 184
7.36.3.6 add_rule_dyn() [2/2] . . . 185
7.36.3.7 calc() . . . 185
7.36.3.8 eval_rules_dyn() . . . 185
7.36.3.9 get_counters() . . . 185
7.36.3.10 get_counts() . . . 186
7.36.3.11 get_current_stats() . . . 186
7.36.3.12 get_data() . . . 186
7.36.3.13 get_rules() . . . 186
7.36.3.14 get_rules_dyn() . . . 186
7.36.3.15 init_support() . . . 187
7.36.3.16 print() . . . 187
7.36.3.17 reset_array() [1/2] . . . 187
7.36.3.18 reset_array() [2/2] . . . 187
7.36.3.19 set_counters() . . . 187
7.36.3.20 set_rules() . . . 188
7.36.3.21 set_rules_dyn() . . . 188
7.36.4 Member Data Documentation . . . 188
7.36.4.1 change_stats . . . 188
7.36.4.2 coordiantes_n_free . . . 188
7.36.4.3 coordiantes_n_locked . . . 188
7.36.4.4 coordinates_free . . . 189
7.36.4.5 coordinates_locked . . . 189
7.36.4.6 current_stats . . . 189
7.36.4.7 delete_counters . . . 189
7.36.4.8 delete_rules . . . 189
7.36.4.9 delete_rules_dyn . . . 190

7.36.4.10 hashes . . . 190
7.36.4.11 hashes_initialized . . . 190
7.36.4.12 M . . . 190
7.36.4.13 max_num_elements . . . 190
7.36.4.14 N . . . 191
7.36.4.15 n_counters . . . 191
7.37 vecHasher< T > Struct Template Reference . . . 191
7.37.1 Detailed Description . . . 191
7.37.2 Member Function Documentation . . . 191
7.37.2.1 operator()() . . . 191

**8 File Documentation** **193**

8.1 include/barry/barray-bones.hpp File Reference . . . 193
8.1.1 Macro Definition Documentation . . . 194
8.1.1.1 BARRAY_BONES_HPP . . . 194
8.2 include/barry/barray-iterator.hpp File Reference . . . 194
8.3 include/barry/barray-meat-operators.hpp File Reference . . . 195
8.3.1 Macro Definition Documentation . . . 196
8.3.1.1 BARRAY_TEMPLATE . . . 196
8.3.1.2 BARRAY_TEMPLATE_ARGS . . . 196
8.3.1.3 BARRAY_TYPE . . . 196
8.3.1.4 BARRY_BARRAY_MEAT_OPERATORS_HPP . . . 197
8.3.1.5 COL . . . 197
8.3.1.6 ROW . . . 197
8.3.2 Function Documentation . . . 197
8.3.2.1 BARRAY_TEMPLATE() [1/6] . . . 197
8.3.2.2 BARRAY_TEMPLATE() [2/6] . . . 197
8.3.2.3 BARRAY_TEMPLATE() [3/6] . . . 198
8.3.2.4 BARRAY_TEMPLATE() [4/6] . . . 198
8.3.2.5 BARRAY_TEMPLATE() [5/6] . . . 198
8.3.2.6 BARRAY_TEMPLATE() [6/6] . . . 198
8.3.2.7 BARRAY_TEMPLATE_ARGS() . . . 198
8.3.2.8 BARRAY_TYPE() . . . 198
8.3.2.9 for() . . . 199
8.3.2.10 operator()() . . . 199
8.3.3 Variable Documentation . . . 199
8.3.3.1 rhs . . . 199
8.3.3.2 this . . . 199
8.4 include/barry/barray-meat.hpp File Reference . . . 200
8.4.1 Macro Definition Documentation . . . 202
8.4.1.1 BARRAY_TEMPLATE . . . 202
8.4.1.2 BARRAY_TEMPLATE_ARGS . . . 202

8.4.1.3 BARRAY_TYPE . . . 203
8.4.1.4 COL . . . 203
8.4.1.5 ROW . . . 203
8.4.2 Function Documentation . . . 203
8.4.2.1 ans() . . . 203
8.4.2.2 BARRAY_TEMPLATE() [1/23] . . . 203
8.4.2.3 BARRAY_TEMPLATE() [2/23] . . . 204
8.4.2.4 BARRAY_TEMPLATE() [3/23] . . . 204
8.4.2.5 BARRAY_TEMPLATE() [4/23] . . . 204
8.4.2.6 BARRAY_TEMPLATE() [5/23] . . . 204
8.4.2.7 BARRAY_TEMPLATE() [6/23] . . . 204
8.4.2.8 BARRAY_TEMPLATE() [7/23] . . . 204
8.4.2.9 BARRAY_TEMPLATE() [8/23] . . . 205
8.4.2.10 BARRAY_TEMPLATE() [9/23] . . . 205
8.4.2.11 BARRAY_TEMPLATE() [10/23] . . . 205
8.4.2.12 BARRAY_TEMPLATE() [11/23] . . . 205
8.4.2.13 BARRAY_TEMPLATE() [12/23] . . . 205
8.4.2.14 BARRAY_TEMPLATE() [13/23] . . . 206
8.4.2.15 BARRAY_TEMPLATE() [14/23] . . . 206
8.4.2.16 BARRAY_TEMPLATE() [15/23] . . . 206
8.4.2.17 BARRAY_TEMPLATE() [16/23] . . . 206
8.4.2.18 BARRAY_TEMPLATE() [17/23] . . . 206
8.4.2.19 BARRAY_TEMPLATE() [18/23] . . . 206
8.4.2.20 BARRAY_TEMPLATE() [19/23] . . . 207
8.4.2.21 BARRAY_TEMPLATE() [20/23] . . . 207
8.4.2.22 BARRAY_TEMPLATE() [21/23] . . . 207
8.4.2.23 BARRAY_TEMPLATE() [22/23] . . . 207
8.4.2.24 BARRAY_TEMPLATE() [23/23] . . . 207
8.4.2.25 COL() . . . 207
8.4.2.26 for() [1/3] . . . 208
8.4.2.27 for() [2/3] . . . 208
8.4.2.28 for() [3/3] . . . 208
8.4.2.29 if() [1/17] . . . 208
8.4.2.30 if() [2/17] . . . 208
8.4.2.31 if() [3/17] . . . 208
8.4.2.32 if() [4/17] . . . 209
8.4.2.33 if() [5/17] . . . 209
8.4.2.34 if() [6/17] . . . 209
8.4.2.35 if() [7/17] . . . 209
8.4.2.36 if() [8/17] . . . 209
8.4.2.37 if() [9/17] . . . 209
8.4.2.38 if() [10/17] . . . 210

8.4.2.39 if() [11/17] . . . 210
8.4.2.40 if() [12/17] . . . 210
8.4.2.41 if() [13/17] . . . 210
8.4.2.42 if() [14/17] . . . 210
8.4.2.43 if() [15/17] . . . 210
8.4.2.44 if() [16/17] . . . 210
8.4.2.45 if() [17/17] . . . 211
8.4.2.46 M() . . . 211
8.4.2.47 resize() [1/2] . . . 211
8.4.2.48 resize() [2/2] . . . 211
8.4.2.49 return() . . . 211
8.4.2.50 ROW() [1/2] . . . 211
8.4.2.51 ROW() [2/2] . . . 211
8.4.3 Variable Documentation . . . 212
8.4.3.1 add . . . 212
8.4.3.2 ans . . . 212
8.4.3.3 Array_ . . . 212
8.4.3.4 check_bounds . . . 212
8.4.3.5 check_exists . . . 213
8.4.3.6 col0 . . . 213
8.4.3.7 const . . . 213
8.4.3.8 copy_data . . . 213
8.4.3.9 data . . . 213
8.4.3.10 delete_data . . . 214
8.4.3.11 delete_data_ . . . 214
8.4.3.12 else . . . 214
8.4.3.13 false . . . 214
8.4.3.14 first . . . 214
8.4.3.15 i1 . . . 215
8.4.3.16 j . . . 215
8.4.3.17 j0 . . . 215
8.4.3.18 j1 . . . 215
8.4.3.19 M . . . 215
8.4.3.20 M_ . . . 216
8.4.3.21 N . . . 216
8.4.3.22 NCells . . . 216
8.4.3.23 report . . . 216
8.4.3.24 return . . . 216
8.4.3.25 row0 . . . 217
8.4.3.26 search . . . 217
8.4.3.27 source . . . 217
8.4.3.28 target . . . 217

8.4.3.29 v . . . 217
8.4.3.30 value . . . 217
8.5 include/barry/barraycell-bones.hpp File Reference . . . 218
8.6 include/barry/barraycell-meat.hpp File Reference . . . 218
8.7 include/barry/barraydense-bones.hpp File Reference . . . 219
8.7.1 Macro Definition Documentation . . . 221
8.7.1.1 BARRY_BARRAYDENSE_BONES_HPP . . . 221
8.8 include/barry/barraydense-meat-operators.hpp File Reference . . . 221
8.8.1 Macro Definition Documentation . . . 222
8.8.1.1 BARRY_BARRAYDENSE_MEAT_OPERATORS_HPP . . . 222
8.8.1.2 BDENSE_TEMPLATE . . . 222
8.8.1.3 BDENSE_TEMPLATE_ARGS . . . 222
8.8.1.4 BDENSE_TYPE . . . 223
8.8.1.5 COL . . . 223
8.8.1.6 POS . . . 223
8.8.1.7 POS_N . . . 223
8.8.1.8 ROW . . . 223
8.8.2 Function Documentation . . . 223
8.8.2.1 BDENSE_TEMPLATE() [1/4] . . . 224
8.8.2.2 BDENSE_TEMPLATE() [2/4] . . . 224
8.8.2.3 BDENSE_TEMPLATE() [3/4] . . . 224
8.8.2.4 BDENSE_TEMPLATE() [4/4] . . . 224
8.8.2.5 BDENSE_TEMPLATE_ARGS() . . . 224
8.8.2.6 BDENSE_TYPE() . . . 224
8.9 include/barry/barraydense-meat.hpp File Reference . . . 225
8.9.1 Macro Definition Documentation . . . 227
8.9.1.1 BDENSE_TEMPLATE . . . 227
8.9.1.2 BDENSE_TEMPLATE_ARGS . . . 228
8.9.1.3 BDENSE_TYPE . . . 228
8.9.1.4 COL . . . 228
8.9.1.5 POS . . . 228
8.9.1.6 POS_N . . . 228
8.9.1.7 ROW . . . 229
8.9.1.8 ZERO_CELL . . . 229
8.9.2 Function Documentation . . . 229
8.9.2.1 ans() . . . 229
8.9.2.2 BDENSE_TEMPLATE() [1/37] . . . 229
8.9.2.3 BDENSE_TEMPLATE() [2/37] . . . 229
8.9.2.4 BDENSE_TEMPLATE() [3/37] . . . 229
8.9.2.5 BDENSE_TEMPLATE() [4/37] . . . 230
8.9.2.6 BDENSE_TEMPLATE() [5/37] . . . 230
8.9.2.7 BDENSE_TEMPLATE() [6/37] . . . 230

8.9.2.8 BDENSE_TEMPLATE() [7/37] . . . 230
8.9.2.9 BDENSE_TEMPLATE() [8/37] . . . 230
8.9.2.10 BDENSE_TEMPLATE() [9/37] . . . 231
8.9.2.11 BDENSE_TEMPLATE() [10/37] . . . 231
8.9.2.12 BDENSE_TEMPLATE() [11/37] . . . 231
8.9.2.13 BDENSE_TEMPLATE() [12/37] . . . 231
8.9.2.14 BDENSE_TEMPLATE() [13/37] . . . 231
8.9.2.15 BDENSE_TEMPLATE() [14/37] . . . 232
8.9.2.16 BDENSE_TEMPLATE() [15/37] . . . 232
8.9.2.17 BDENSE_TEMPLATE() [16/37] . . . 232
8.9.2.18 BDENSE_TEMPLATE() [17/37] . . . 232
8.9.2.19 BDENSE_TEMPLATE() [18/37] . . . 232
8.9.2.20 BDENSE_TEMPLATE() [19/37] . . . 233
8.9.2.21 BDENSE_TEMPLATE() [20/37] . . . 233
8.9.2.22 BDENSE_TEMPLATE() [21/37] . . . 233
8.9.2.23 BDENSE_TEMPLATE() [22/37] . . . 233
8.9.2.24 BDENSE_TEMPLATE() [23/37] . . . 233
8.9.2.25 BDENSE_TEMPLATE() [24/37] . . . 234
8.9.2.26 BDENSE_TEMPLATE() [25/37] . . . 234
8.9.2.27 BDENSE_TEMPLATE() [26/37] . . . 234
8.9.2.28 BDENSE_TEMPLATE() [27/37] . . . 234
8.9.2.29 BDENSE_TEMPLATE() [28/37] . . . 234
8.9.2.30 BDENSE_TEMPLATE() [29/37] . . . 234
8.9.2.31 BDENSE_TEMPLATE() [30/37] . . . 235
8.9.2.32 BDENSE_TEMPLATE() [31/37] . . . 235
8.9.2.33 BDENSE_TEMPLATE() [32/37] . . . 235
8.9.2.34 BDENSE_TEMPLATE() [33/37] . . . 235
8.9.2.35 BDENSE_TEMPLATE() [34/37] . . . 235
8.9.2.36 BDENSE_TEMPLATE() [35/37] . . . 235
8.9.2.37 BDENSE_TEMPLATE() [36/37] . . . 236
8.9.2.38 BDENSE_TEMPLATE() [37/37] . . . 236
8.9.2.39 for() . . . 236
8.9.2.40 if() [1/4] . . . 236
8.9.2.41 if() [2/4] . . . 236
8.9.2.42 if() [3/4] . . . 236
8.9.2.43 if() [4/4] . . . 237
8.9.2.44 insert_cell() [1/2] . . . 237
8.9.2.45 insert_cell() [2/2] . . . 237
8.9.2.46 M() . . . 237
8.9.2.47 resize() [1/6] . . . 237
8.9.2.48 resize() [2/6] . . . 238
8.9.2.49 resize() [3/6] . . . 238

8.9.2.50 resize() [4/6] . . . 238
8.9.2.51 resize() [5/6] . . . 238
8.9.2.52 resize() [6/6] . . . 238
8.9.2.53 rm_cell() [1/3] . . . 238
8.9.2.54 rm_cell() [2/3] . . . 239
8.9.2.55 rm_cell() [3/3] . . . 239
8.9.2.56 va_end() . . . 239
8.9.2.57 va_start() . . . 239
8.9.2.58 vprintf() . . . 239
8.9.3 Variable Documentation . . . 239
8.9.3.1 add . . . 240
8.9.3.2 ans . . . 240
8.9.3.3 check_bounds . . . 240
8.9.3.4 check_exists . . . 240
8.9.3.5 col . . . 241
8.9.3.6 const . . . 241
8.9.3.7 copy_data . . . 241
8.9.3.8 data . . . 241
8.9.3.9 delete_data . . . 241
8.9.3.10 delete_data_ . . . 242
8.9.3.11 el . . . 242
8.9.3.12 el_colsums . . . 242
8.9.3.13 el_rowsums . . . 242
8.9.3.14 else . . . 242
8.9.3.15 false . . . 243
8.9.3.16 i1 . . . 243
8.9.3.17 j . . . 243
8.9.3.18 j0 . . . 243
8.9.3.19 j1 . . . 243
8.9.3.20 M . . . 243
8.9.3.21 M_ . . . 244
8.9.3.22 N . . . 244
8.9.3.23 report . . . 244
8.9.3.24 return . . . 244
8.9.3.25 source . . . 244
8.9.3.26 target . . . 245
8.9.3.27 v . . . 245
8.9.3.28 val0 . . . 245
8.9.3.29 val1 . . . 245
8.9.3.30 value . . . 245
8.10 include/barry/barraydensecell-bones.hpp File Reference . . . 246
8.10.1 Macro Definition Documentation . . . 246

8.10.1.1 POS . . . 247
8.11 include/barry/barraydensecell-meat.hpp File Reference . . . 247
8.11.1 Macro Definition Documentation . . . 248
8.11.1.1 POS . . . 248
8.12 include/barry/barraydensecol-bones.hpp File Reference . . . 248
8.12.1 Macro Definition Documentation . . . 249
8.12.1.1 POS . . . 249
8.12.1.2 POS_N . . . 249
8.12.1.3 ZERO_CELL . . . 249
8.13 include/barry/barraydenserow-bones.hpp File Reference . . . 249
8.13.1 Macro Definition Documentation . . . 250
8.13.1.1 POS . . . 250
8.13.1.2 POS_N . . . 250
8.13.1.3 ZERO_CELL . . . 250
8.14 include/barry/barrayrow-bones.hpp File Reference . . . 251
8.15 include/barry/barrayrow-meat.hpp File Reference . . . 251
8.15.1 Macro Definition Documentation . . . 252
8.15.1.1 BARRY_BARRAYROW_MEAT_HPP . . . 252
8.15.1.2 BROW_TEMPLATE . . . 253
8.15.1.3 BROW_TEMPLATE_ARGS . . . 253
8.15.1.4 BROW_TYPE . . . 253
8.15.2 Function Documentation . . . 253
8.15.2.1 BROW_TEMPLATE() [1/5] . . . 253
8.15.2.2 BROW_TEMPLATE() [2/5] . . . 253
8.15.2.3 BROW_TEMPLATE() [3/5] . . . 254
8.15.2.4 BROW_TEMPLATE() [4/5] . . . 254
8.15.2.5 BROW_TEMPLATE() [5/5] . . . 254
8.16 include/barry/barrayvector-bones.hpp File Reference . . . 254
8.17 include/barry/barrayvector-meat.hpp File Reference . . . 255
8.17.1 Macro Definition Documentation . . . 256
8.17.1.1 BARRY_BARRAYVECTOR_MEAT_HPP . . . 256
8.18 include/barry/barry-configuration.hpp File Reference . . . 256
8.18.1 Macro Definition Documentation . . . 257
8.18.1.1 BARRY_CHECK_SUPPORT . . . 257
8.18.1.2 BARRY_ISFINITE . . . 257
8.18.1.3 BARRY_MAX_NUM_ELEMENTS . . . 257
8.18.1.4 BARRY_SAFE_EXP . . . 257
8.18.1.5 printf_barry . . . 257
8.18.2 Typedef Documentation . . . 257
8.18.2.1 Map . . . 258
8.19 include/barry/barry-debug.hpp File Reference . . . 258
8.19.1 Macro Definition Documentation . . . 258

8.19.1.1 BARRY_DEBUG_LEVEL . . . 258
8.20 include/barry/barry-macros.hpp File Reference . . . 258
8.20.1 Macro Definition Documentation . . . 259
8.20.1.1 BARRY_ONE . . . 259
8.20.1.2 BARRY_ONE_DENSE . . . 259
8.20.1.3 BARRY_UNUSED . . . 259
8.20.1.4 BARRY_ZERO . . . 259
8.20.1.5 BARRY_ZERO_DENSE . . . 260
8.21 include/barry/barry.hpp File Reference . . . 260
8.21.1 Macro Definition Documentation . . . 261
8.21.1.1 BARRY_HPP . . . 261
8.21.1.2 BARRY_VERSION . . . 261
8.21.1.3 COUNTER_FUNCTION . . . 262
8.21.1.4 COUNTER_LAMBDA . . . 262
8.21.1.5 RULE_FUNCTION . . . 262
8.21.1.6 RULE_LAMBDA . . . 262
8.22 include/barry/cell-bones.hpp File Reference . . . 263
8.23 include/barry/cell-meat.hpp File Reference . . . 263
8.24 include/barry/col-bones.hpp File Reference . . . 264
8.25 include/barry/counters-bones.hpp File Reference . . . 264
8.26 include/barry/counters-meat.hpp File Reference . . . 266
8.26.1 Macro Definition Documentation . . . 268
8.26.1.1 COUNTER_TEMPLATE . . . 268
8.26.1.2 COUNTER_TEMPLATE_ARGS . . . 268
8.26.1.3 COUNTER_TYPE . . . 268
8.26.1.4 COUNTERS_TEMPLATE . . . 268
8.26.1.5 COUNTERS_TEMPLATE_ARGS . . . 268
8.26.1.6 COUNTERS_TYPE . . . 269
8.26.2 Function Documentation . . . 269
8.26.2.1 count_fun() . . . 269
8.26.2.2 COUNTER_TEMPLATE() [1/7] . . . 269
8.26.2.3 COUNTER_TEMPLATE() [2/7] . . . 269
8.26.2.4 COUNTER_TEMPLATE() [3/7] . . . 269
8.26.2.5 COUNTER_TEMPLATE() [4/7] . . . 270
8.26.2.6 COUNTER_TEMPLATE() [5/7] . . . 270
8.26.2.7 COUNTER_TEMPLATE() [6/7] . . . 270
8.26.2.8 COUNTER_TEMPLATE() [7/7] . . . 270
8.26.2.9 COUNTERS_TEMPLATE() [1/8] . . . 270
8.26.2.10 COUNTERS_TEMPLATE() [2/8] . . . 271
8.26.2.11 COUNTERS_TEMPLATE() [3/8] . . . 271
8.26.2.12 COUNTERS_TEMPLATE() [4/8] . . . 271
8.26.2.13 COUNTERS_TEMPLATE() [5/8] . . . 271

8.26.2.14 COUNTERS_TEMPLATE() [6/8] . . . 271
8.26.2.15 COUNTERS_TEMPLATE() [7/8] . . . 272
8.26.2.16 COUNTERS_TEMPLATE() [8/8] . . . 272
8.26.2.17 data() . . . 272
8.26.2.18 delete_data() [1/3] . . . 272
8.26.2.19 delete_data() [2/3] . . . 272
8.26.2.20 delete_data() [3/3] . . . 272
8.26.2.21 delete_to_be_deleted() [1/2] . . . 273
8.26.2.22 delete_to_be_deleted() [2/2] . . . 273
8.26.2.23 desc() . . . 273
8.26.2.24 init_fun() [1/3] . . . 273
8.26.2.25 init_fun() [2/3] . . . 273
8.26.2.26 init_fun() [3/3] . . . 274
8.26.2.27 name() . . . 274
8.26.2.28 push_back() [1/2] . . . 274
8.26.2.29 push_back() [2/2] . . . 274
8.26.2.30 to_be_deleted() [1/2] . . . 274
8.26.2.31 to_be_deleted() [2/2] . . . 274
8.26.3 Variable Documentation . . . 274
8.26.3.1 count_fun_ . . . 275
8.26.3.2 counter . . . 275
8.26.3.3 counter_ . . . 275
8.26.3.4 data_ . . . 275
8.26.3.5 delete_data_ . . . 276
8.26.3.6 desc_ . . . 276
8.26.3.7 i . . . 276
8.26.3.8 init_fun_ . . . 276
8.26.3.9 j . . . 276
8.26.3.10 name_ . . . 277
8.26.3.11 noexcept . . . 277
8.26.3.12 return . . . 277
8.27 include/barry/counters/network-css.hpp File Reference . . . 278
8.27.1 Macro Definition Documentation . . . 279
8.27.1.1 CSS_APPEND . . . 279
8.27.1.2 CSS_CASE_ELSE . . . 280
8.27.1.3 CSS_CASE_PERCEIVED . . . 280
8.27.1.4 CSS_CASE_TRUTH . . . 280
8.27.1.5 CSS_CHECK_SIZE . . . 280
8.27.1.6 CSS_CHECK_SIZE_INIT . . . 280
8.27.1.7 CSS_NET_COUNTER_LAMBDA_INIT . . . 281
8.27.1.8 CSS_PERCEIVED_CELLS . . . 281
8.27.1.9 CSS_SIZE . . . 281

8.27.1.10 CSS_TRUE_CELLS . . . 281
8.27.2 Function Documentation . . . 281
8.27.2.1 counter_css_census01() . . . 282
8.27.2.2 counter_css_census02() . . . 282
8.27.2.3 counter_css_census03() . . . 282
8.27.2.4 counter_css_census04() . . . 282
8.27.2.5 counter_css_census05() . . . 283
8.27.2.6 counter_css_census06() . . . 283
8.27.2.7 counter_css_census07() . . . 283
8.27.2.8 counter_css_census08() . . . 283
8.27.2.9 counter_css_census09() . . . 284
8.27.2.10 counter_css_census10() . . . 284
8.27.2.11 counter_css_completely_false_recip_comiss() . . . 284
8.27.2.12 counter_css_completely_false_recip_omiss() . . . 284
8.27.2.13 counter_css_mixed_recip() . . . 285
8.27.2.14 counter_css_partially_false_recip_commi() . . . 285
8.27.2.15 counter_css_partially_false_recip_omiss() . . . 285
8.28 include/barry/counters/network.hpp File Reference . . . 286
8.28.1 Macro Definition Documentation . . . 288
8.28.1.1 BARRY_ZERO_NETWORK . . . 289
8.28.1.2 BARRY_ZERO_NETWORK_DENSE . . . 289
8.28.1.3 NET_C_DATA_IDX . . . 289
8.28.1.4 NET_C_DATA_NUM . . . 289
8.28.1.5 NETWORK_COUNTER . . . 289
8.28.1.6 NETWORK_COUNTER_LAMBDA . . . 290
8.28.1.7 NETWORK_RULE . . . 290
8.28.1.8 NETWORK_RULE_LAMBDA . . . 290
8.28.1.9 NETWORKDENSE_COUNTER_LAMBDA . . . 290
8.28.2 Typedef Documentation . . . 291
8.28.2.1 NetCounter . . . 291
8.28.2.2 NetCounters . . . 291
8.28.2.3 NetModel . . . 291
8.28.2.4 NetRule . . . 291
8.28.2.5 NetRules . . . 291
8.28.2.6 NetStatsCounter . . . 292
8.28.2.7 NetSupport . . . 292
8.28.2.8 Network . . . 292
8.28.2.9 NetworkDense . . . 292
8.28.3 Function Documentation . . . 292
8.28.3.1 rules_zerodiag() . . . 292
8.29 include/barry/counters/phylo.hpp File Reference . . . 293
8.29.1 Macro Definition Documentation . . . 295

8.29.1.1 DEFAULT_DUPLICATION . . . 295
8.29.1.2 DUPL_DUPL . . . 295
8.29.1.3 DUPL_EITH . . . 295
8.29.1.4 DUPL_SPEC . . . 296
8.29.1.5 IF_MATCHES . . . 296
8.29.1.6 IF_NOTMATCHES . . . 296
8.29.1.7 IS_DUPLICATION . . . 296
8.29.1.8 IS_EITHER . . . 296
8.29.1.9 IS_SPECIATION . . . 297
8.29.1.10 MAKE_DUPL_VARS . . . 297
8.29.1.11 PHYLO_CHECK_MISSING . . . 297
8.29.1.12 PHYLO_COUNTER_LAMBDA . . . 297
8.29.1.13 PHYLO_RULE_DYN_LAMBDA . . . 298
8.29.2 Typedef Documentation . . . 298
8.29.2.1 PhyloArray . . . 298
8.29.2.2 PhyloCounter . . . 298
8.29.2.3 PhyloCounters . . . 298
8.29.2.4 PhyloModel . . . 298
8.29.2.5 PhyloPowerSet . . . 299
8.29.2.6 PhyloRule . . . 299
8.29.2.7 PhyloRuleData . . . 299
8.29.2.8 PhyloRuleDyn . . . 299
8.29.2.9 PhyloRules . . . 299
8.29.2.10 PhyloRulesDyn . . . 299
8.29.2.11 PhyloStatsCounter . . . 300
8.29.2.12 PhyloSupport . . . 300
8.29.3 Function Documentation . . . 300
8.29.3.1 get_last_name() . . . 300
8.30 include/barry/model-bones.hpp File Reference . . . 300
8.30.1 Function Documentation . . . 301
8.30.1.1 keygen_default() . . . 301
8.31 include/barry/model-meat.hpp File Reference . . . 302
8.31.1 Macro Definition Documentation . . . 302
8.31.1.1 MODEL_TEMPLATE . . . 302
8.31.1.2 MODEL_TEMPLATE_ARGS . . . 303
8.31.1.3 MODEL_TYPE . . . 303
8.31.2 Function Documentation . . . 303
8.31.2.1 likelihood_() . . . 303
8.31.2.2 MODEL_TEMPLATE() [1/2] . . . 303
8.31.2.3 MODEL_TEMPLATE() [2/2] . . . 304
8.31.2.4 update_normalizing_constant() . . . 304
8.32 include/barry/models/geese.hpp File Reference . . . 304

8.33 include/barry/models/geese/flock-bones.hpp File Reference . . . 305
8.34 include/barry/models/geese/flock-meat.hpp File Reference . . . 305
8.35 include/barry/models/geese/geese-bones.hpp File Reference . . . 306
8.35.1 Macro Definition Documentation . . . 306
8.35.1.1 INITIALIZED . . . 307
8.35.2 Function Documentation . . . 307
8.35.2.1 keygen_full() . . . 307
8.35.2.2 RULE_FUNCTION() . . . 307
8.35.2.3 vec_diff() . . . 307
8.35.2.4 vector_caster() . . . 307
8.36 include/barry/models/geese/geese-meat-constructors.hpp File Reference . . . 308
8.37 include/barry/models/geese/geese-meat-likelihood.hpp File Reference . . . 308
8.38 include/barry/models/geese/geese-meat-likelihood_exhaust.hpp File Reference . . . 309
8.39 include/barry/models/geese/geese-meat-predict.hpp File Reference . . . 310
8.40 include/barry/models/geese/geese-meat-predict_exhaust.hpp File Reference . . . 310
8.41 include/barry/models/geese/geese-meat-predict_sim.hpp File Reference . . . 311
8.42 include/barry/models/geese/geese-meat-simulate.hpp File Reference . . . 311
8.43 include/barry/models/geese/geese-meat.hpp File Reference . . . 312
8.44 include/barry/models/geese/geese-node-bones.hpp File Reference . . . 312
8.45 include/barry/powerset-bones.hpp File Reference . . . 313
8.46 include/barry/powerset-meat.hpp File Reference . . . 314
8.47 include/barry/progress.hpp File Reference . . . 315
8.47.1 Macro Definition Documentation . . . 315
8.47.1.1 BARRY_PROGRESS_BAR_WIDTH . . . 315
8.48 include/barry/rules-bones.hpp File Reference . . . 315
8.48.1 Function Documentation . . . 316
8.48.1.1 rule_fun_default() . . . 316
8.49 include/barry/rules-meat.hpp File Reference . . . 317
8.50 include/barry/statscounter-bones.hpp File Reference . . . 317
8.51 include/barry/statscounter-meat.hpp File Reference . . . 319
8.51.1 Macro Definition Documentation . . . 320
8.51.1.1 STATSCOUNTER_TEMPLATE . . . 320
8.51.1.2 STATSCOUNTER_TEMPLATE_ARGS . . . 320
8.51.1.3 STATSCOUNTER_TYPE . . . 321
8.51.2 Function Documentation . . . 321
8.51.2.1 clear() . . . 321
8.51.2.2 for() . . . 321
8.51.2.3 resize() . . . 321
8.51.2.4 STATSCOUNTER_TEMPLATE() [1/9] . . . 321
8.51.2.5 STATSCOUNTER_TEMPLATE() [2/9] . . . 321
8.51.2.6 STATSCOUNTER_TEMPLATE() [3/9] . . . 322
8.51.2.7 STATSCOUNTER_TEMPLATE() [4/9] . . . 322

8.51.2.8 STATSCOUNTER_TEMPLATE() [5/9] . . . 322
8.51.2.9 STATSCOUNTER_TEMPLATE() [6/9] . . . 322
8.51.2.10 STATSCOUNTER_TEMPLATE() [7/9] . . . 322
8.51.2.11 STATSCOUNTER_TEMPLATE() [8/9] . . . 322
8.51.2.12 STATSCOUNTER_TEMPLATE() [9/9] . . . 323
8.51.3 Variable Documentation . . . 323
8.51.3.1 counter . . . 323
8.51.3.2 counter_deleted . . . 323
8.51.3.3 counters . . . 323
8.51.3.4 counters_ . . . 323
8.51.3.5 current_stats . . . 324
8.51.3.6 EmptyArray . . . 324
8.51.3.7 f_ . . . 324
8.51.3.8 j . . . 324
8.51.3.9 return . . . 324
8.52 include/barry/statsdb.hpp File Reference . . . 325
8.53 include/barry/support-bones.hpp File Reference . . . 325
8.54 include/barry/support-meat.hpp File Reference . . . 327
8.54.1 Macro Definition Documentation . . . 328
8.54.1.1 BARRY_SUPPORT_MEAT_HPP . . . 329
8.54.1.2 SUPPORT_TEMPLATE . . . 329
8.54.1.3 SUPPORT_TEMPLATE_ARGS . . . 329
8.54.1.4 SUPPORT_TYPE . . . 329
8.54.2 Function Documentation . . . 329
8.54.2.1 calc_backend_dense() . . . 330
8.54.2.2 calc_backend_sparse() . . . 330
8.54.2.3 for() . . . 330
8.54.2.4 if() [1/4] . . . 330
8.54.2.5 if() [2/4] . . . 330
8.54.2.6 if() [3/4] . . . 330
8.54.2.7 if() [4/4] . . . 331
8.54.2.8 insert_cell() [1/2] . . . 331
8.54.2.9 insert_cell() [2/2] . . . 331
8.54.2.10 rm_cell() . . . 331
8.54.2.11 SUPPORT_TEMPLATE() [1/17] . . . 331
8.54.2.12 SUPPORT_TEMPLATE() [2/17] . . . 332
8.54.2.13 SUPPORT_TEMPLATE() [3/17] . . . 332
8.54.2.14 SUPPORT_TEMPLATE() [4/17] . . . 332
8.54.2.15 SUPPORT_TEMPLATE() [5/17] . . . 332
8.54.2.16 SUPPORT_TEMPLATE() [6/17] . . . 332
8.54.2.17 SUPPORT_TEMPLATE() [7/17] . . . 333
8.54.2.18 SUPPORT_TEMPLATE() [8/17] . . . 333

8.54.2.19 SUPPORT_TEMPLATE() [9/17] . . . 333
8.54.2.20 SUPPORT_TEMPLATE() [10/17] . . . 333
8.54.2.21 SUPPORT_TEMPLATE() [11/17] . . . 333
8.54.2.22 SUPPORT_TEMPLATE() [12/17] . . . 333
8.54.2.23 SUPPORT_TEMPLATE() [13/17] . . . 334
8.54.2.24 SUPPORT_TEMPLATE() [14/17] . . . 334
8.54.2.25 SUPPORT_TEMPLATE() [15/17] . . . 334
8.54.2.26 SUPPORT_TEMPLATE() [16/17] . . . 334
8.54.2.27 SUPPORT_TEMPLATE() [17/17] . . . 334
8.54.3 Variable Documentation . . . 334
8.54.3.1 array_bank . . . 335
8.54.3.2 change_stats_different . . . 335
8.54.3.3 coord_i . . . 335
8.54.3.4 coord_j . . . 335
8.54.3.5 counters . . . 335
8.54.3.6 counters_ . . . 335
8.54.3.7 delete_counters . . . 336
8.54.3.8 delete_rules . . . 336
8.54.3.9 delete_rules_dyn . . . 336
8.54.3.10 else . . . 336
8.54.3.11 f_ . . . 336
8.54.3.12 hashes . . . 337
8.54.3.13 return . . . 337
8.54.3.14 rules . . . 337
8.54.3.15 rules_ . . . 337
8.54.3.16 rules_dyn . . . 337
8.54.3.17 stats_bank . . . 338
8.54.3.18 tmp_chng . . . 338
8.55 include/barry/typedefs.hpp File Reference . . . 338
8.55.1 Typedef Documentation . . . 340
8.55.1.1 Col_type . . . 340
8.55.1.2 Counter_fun_type . . . 340
8.55.1.3 Counts_type . . . 341
8.55.1.4 MapVec_type . . . 341
8.55.1.5 Row_type . . . 341
8.55.1.6 Rule_fun_type . . . 341
8.55.1.7 uint . . . 341
8.55.2 Function Documentation . . . 341
8.55.2.1 vec_equal() . . . 341
8.55.2.2 vec_equal_approx() . . . 342
8.55.2.3 vec_inner_prod() [1/2] . . . 342
8.55.2.4 vec_inner_prod() [2/2] . . . 342

8.56 README.md File Reference . . . . . 342

**Index** **343**

# Chapter 1

# Main Page

## Barry: your to-go motif accountant

`This repository` contains a C++ template library that essentially counts sufficient statistics on binary arrays. The goal of the library 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.

This 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 . . . . . 11
Statistical Models . . . . . 11
Network counters . . . . . 12
Phylo counters . . . . . 19
Phylo rules . . . . . 27

# Chapter 3

# Class Index

## 3.1 Class List

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


BArray< Cell_Type, Data_Type >
: Baseline class for binary arrays . . . 33
BArrayCell< Cell_Type, Data_Type > . . . 47
BArrayCell_const< Cell_Type, Data_Type > . . . 50
BArrayDense< Cell_Type, Data_Type >
: Baseline class for binary arrays . . . 52
BArrayDenseCell< Cell_Type, Data_Type > . . . 67
BArrayDenseCell_const< Cell_Type, Data_Type > . . . 71
BArrayDenseCol< Cell_Type, Data_Type > . . . 71
BArrayDenseCol_const< Cell_Type, Data_Type > . . . 73
BArrayDenseRow< Cell_Type, Data_Type > . . . 75
BArrayDenseRow_const< Cell_Type, Data_Type > . . . 78
BArrayRow< Cell_Type, Data_Type > . . . 80
BArrayRow_const< Cell_Type, Data_Type > . . . 82
BArrayVector< Cell_Type, Data_Type >
: Row or column of a `BArray` . . . 84
BArrayVector_const< Cell_Type, Data_Type > . . . 88
Cell< Cell_Type >
: Entries in BArray. For now, it only has two members: . . . 92
Cell_const< Cell_Type > . . . 97
ConstBArrayRowIter< Cell_Type, Data_Type > . . . 97
Counter< Array_Type, Data_Type >
: A counter function based on change statistics . . . 99
Counters< Array_Type, Data_Type >
: Vector of counters . . . 104
Entries< Cell_Type >
: A wrapper class to store `source`, `target`, `val` from a `BArray` object . . . 109
Flock
: A Flock is a group of Geese . . . 111
FreqTable< T >
: Database of statistics . . . 118
Geese
: Annotated Phylo Model . . . 121
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: . . . 133

NetCounterData
Data class used to store arbitrary uint or double vectors . . . 147
NetworkData
Data class for Networks . . . 149
Node
A single node for the model . . . 151
NodeData
Data definition for the PhyloArray class . . . 157
PhyloCounterData . . . 158
PhyloRuleDynData . . . 161
PowerSet< Array_Type, Data_Rule_Type >
Powerset of a binary array . . . 163
Progress
A simple progress bar . . . 170
Rule< Array_Type, Data_Type >
Rule for determining if a cell should be included in a sequence . . . 171
Rules< Array_Type, Data_Type >
Vector of objects of class Rule . . . 173
StatsCounter< Array_Type, Data_Type >
Count stats for a single Array . . . 177
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >
Compute the support of sufficient statistics . . . 181
vecHasher< T > . . . 191

# Chapter 4

# File Index

## 4.1 File List

Here is a list of all files with brief descriptions:


include/barry/barray-bones.hpp . . . 193
include/barry/barray-iterator.hpp . . . 194
include/barry/barray-meat-operators.hpp . . . 195
include/barry/barray-meat.hpp . . . 200
include/barry/barraycell-bones.hpp . . . 218
include/barry/barraycell-meat.hpp . . . 218
include/barry/barraydense-bones.hpp . . . 219
include/barry/barraydense-meat-operators.hpp . . . 221
include/barry/barraydense-meat.hpp . . . 225
include/barry/barraydensecell-bones.hpp . . . 246
include/barry/barraydensecell-meat.hpp . . . 247
include/barry/barraydensecol-bones.hpp . . . 248
include/barry/barraydenserow-bones.hpp . . . 249
include/barry/barrayrow-bones.hpp . . . 251
include/barry/barrayrow-meat.hpp . . . 251
include/barry/barrayvector-bones.hpp . . . 254
include/barry/barrayvector-meat.hpp . . . 255
include/barry/barry-configuration.hpp . . . 256
include/barry/barry-debug.hpp . . . 258
include/barry/barry-macros.hpp . . . 258
include/barry/barry.hpp . . . 260
include/barry/cell-bones.hpp . . . 263
include/barry/cell-meat.hpp . . . 263
include/barry/col-bones.hpp . . . 264
include/barry/counters-bones.hpp . . . 264
include/barry/counters-meat.hpp . . . 266
include/barry/model-bones.hpp . . . 300
include/barry/model-meat.hpp . . . 302
include/barry/powerset-bones.hpp . . . 313
include/barry/powerset-meat.hpp . . . 314
include/barry/progress.hpp . . . 315
include/barry/rules-bones.hpp . . . 315
include/barry/rules-meat.hpp . . . 317
include/barry/statscounter-bones.hpp . . . 317
include/barry/statscounter-meat.hpp . . . 319

include/barry/statsdb.hpp . . . 325
include/barry/support-bones.hpp . . . 325
include/barry/support-meat.hpp . . . 327
include/barry/typedefs.hpp . . . 338
include/barry/counters/network-css.hpp . . . 278
include/barry/counters/network.hpp . . . 286
include/barry/counters/phylo.hpp . . . 293
include/barry/models/geese.hpp . . . 304
include/barry/models/geese/flock-bones.hpp . . . 305
include/barry/models/geese/flock-meat.hpp . . . 305
include/barry/models/geese/geese-bones.hpp . . . 306
include/barry/models/geese/geese-meat-constructors.hpp . . . 308
include/barry/models/geese/geese-meat-likelihood.hpp . . . 308
include/barry/models/geese/geese-meat-likelihood_exhaust.hpp . . . 309
include/barry/models/geese/geese-meat-predict.hpp . . . 310
include/barry/models/geese/geese-meat-predict_exhaust.hpp . . . 310
include/barry/models/geese/geese-meat-predict_sim.hpp . . . 311
include/barry/models/geese/geese-meat-simulate.hpp . . . 311
include/barry/models/geese/geese-meat.hpp . . . 312
include/barry/models/geese/geese-node-bones.hpp . . . 312

# Chapter 5

# Module Documentation

## 5.1 Counting

### Classes

- class NetworkData

  *Data class for Networks.*
- class NodeData

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

  *A counter function based on change statistics.*

### 5.1.1 Detailed Description

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

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

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

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

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

## 5.2 Statistical Models

Statistical models available in `barry`.

### Classes

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

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

  *A Flock is a group of Geese.*
- class Geese

  *Annotated Phylo Model.*

#### 5.2.1 Detailed Description

Statistical models available in `barry`.

## 5.3 Network counters

Counters for network models.

### Functions

- 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, uint 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, uint attr_id, double alpha=1.0, double tail_head=true)

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

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

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

  *Counts number of vertices with a given out-degree.*

### 5.3.1 Detailed Description

Counters for network models.

**Parameters**

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

### 5.3.2 Function Documentation

#### 5.3.2.1 counter_absdiff()

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

Sum of absolute attribute difference between ego and alter.

Definition at line 852 of file network.hpp.

#### 5.3.2.2 counter_ctriads() [1/2]

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

Definition at line 631 of file network.hpp.

#### 5.3.2.3 counter_ctriads() [2/2]

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

Definition at line 578 of file network.hpp.

#### 5.3.2.4 counter_degree()

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

Counts number of vertices with a given out-degree.

Definition at line 1269 of file network.hpp.

#### 5.3.2.5 counter_density()

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

Definition at line 695 of file network.hpp.

#### 5.3.2.6 counter_diff()

```
template<typename Tnet = Network>
void counter_diff (
            NetCounters< Tnet > * counters,
            uint attr_id,
            double alpha = 1.0,
            double tail_head = true )  [inline]
```

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

Definition at line 897 of file network.hpp.

#### 5.3.2.7 counter_edges()

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

Number of edges.

Definition at line 152 of file network.hpp.

#### 5.3.2.8 counter_idegree() [1/2]

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

Definition at line 1113 of file network.hpp.

#### 5.3.2.9 counter_idegree() [2/2]

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

Counts number of vertices with a given in-degree.

Definition at line 1066 of file network.hpp.

#### 5.3.2.10 counter_idegree15() [1/2]

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

Definition at line 747 of file network.hpp.

#### 5.3.2.11 counter_idegree15() [2/2]

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

Definition at line 721 of file network.hpp.

#### 5.3.2.12 counter_isolates() [1/2]

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

Definition at line 211 of file network.hpp.

#### 5.3.2.13 counter_isolates() [2/2]

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

Number of isolated vertices.

Definition at line 174 of file network.hpp.

#### 5.3.2.14 counter_istar2() [1/2]

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

Definition at line 325 of file network.hpp.

#### 5.3.2.15 counter_istar2() [2/2]

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

Definition at line 304 of file network.hpp.

#### 5.3.2.16 counter_mutual()

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

Number of mutual ties.

Definition at line 250 of file network.hpp.

#### 5.3.2.17 counter_nodecov()

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

Definition at line 1010 of file network.hpp.

#### 5.3.2.18 counter_nodeicov()

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

Definition at line 960 of file network.hpp.

#### 5.3.2.19 counter_nodematch()

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

Definition at line 1035 of file network.hpp.

#### 5.3.2.20 counter_nodeocov()

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

Definition at line 985 of file network.hpp.

#### 5.3.2.21 counter_odegree() [1/2]

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

Definition at line 1214 of file network.hpp.

#### 5.3.2.22 counter_odegree() [2/2]

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

Counts number of vertices with a given out-degree.

Definition at line 1166 of file network.hpp.

#### 5.3.2.23 counter_odegree15() [1/2]

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

Definition at line 811 of file network.hpp.

#### 5.3.2.24 counter_odegree15() [2/2]

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

Definition at line 785 of file network.hpp.

#### 5.3.2.25 counter_ostar2() [1/2]

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

Definition at line 381 of file network.hpp.

#### 5.3.2.26 counter_ostar2() [2/2]

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

Definition at line 358 of file network.hpp.

#### 5.3.2.27 counter_ttriads() [1/2]

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

Definition at line 501 of file network.hpp.

#### 5.3.2.28 counter_ttriads() [2/2]

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

Definition at line 413 of file network.hpp.

#### 5.3.2.29 NETWORK_COUNTER()

```
NETWORK_COUNTER (
            init_single_attr  )
```

Definition at line 941 of file network.hpp.

## 5.4 Phylo counters

Counters for phylogenetic modeling.

### Functions

- void counter_overall_gains (PhyloCounters *counters, unsigned int duplication=DEFAULT_DUPLICATION)

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

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

  *k genes gain function nfun*
- void counter_genes_changing (PhyloCounters *counters, unsigned int duplication=DEFAULT_DUPLICATION)

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

  *Keeps track of how many pairs of genes preserve pseudostate.*
- void counter_prop_genes_changing (PhyloCounters *counters, unsigned int duplication=DEFAULT_DUPLICATION)

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

*Overall functional loss.*

- void counter_maxfuns (PhyloCounters *counters, uint lb, uint ub, unsigned int duplication=DEFAULT_DUPLICATION)

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

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

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

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

  *Co-evolution (joint gain or loss)*
- void counter_longest (PhyloCounters *counters, unsigned int duplication=DEFAULT_DUPLICATION)

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

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

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

  *Function co-opting.*
- void counter_k_genes_changing (PhyloCounters *counters, unsigned int k, unsigned int duplication=DEFAULT_DUPLICATION)

  *Indicator function. Equals to one if $k$ genes changed and zero otherwise.*
- void counter_less_than_p_prop_genes_changing (PhyloCounters *counters, double p, unsigned int duplication=DEFAULT_DUPLICATION)

  *Indicator function. Equals to one if $k$ genes changed and zero otherwise.*
- void counter_gains_from_0 (PhyloCounters *counters, std::vector< uint > nfun, unsigned int duplication=DEFAULT_DUPLICAT

  *Used when all the functions are in 0 (like the root node prob.)*
- void counter_overall_gains_from_0 (PhyloCounters *counters, unsigned int duplication=DEFAULT_DUPLICATION)

  *Used when all the functions are in 0 (like the root node prob.)*
- void counter_pairwise_overall_change (PhyloCounters *counters, unsigned int duplication=DEFAULT_DUPLICATION)

  *Used when all the functions are in 0 (like the root node prob.)*
- void counter_pairwise_preserving (PhyloCounters *counters, uint nfunA, uint nfunB, unsigned int duplication=DEFAULT_DUPLICATION)

  *Used when all the functions are in 0 (like the root node prob.)*
- void counter_pairwise_first_gain (PhyloCounters *counters, uint nfunA, uint nfunB, unsigned int duplication=DEFAULT_DUPLICATION)

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

### 5.4.1 Detailed Description

Counters for phylogenetic modeling.

**Parameters**

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

### 5.4.2 Function Documentation

#### 5.4.2.1 counter_co_opt()

```
void counter_co_opt (
            PhyloCounters * counters,
            uint nfunA,
            uint nfunB,
            unsigned int duplication = DEFAULT_DUPLICATION )  [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 1336 of file phylo.hpp.

#### 5.4.2.2 counter_cogain()

```
void counter_cogain (
            PhyloCounters * counters,
            uint nfunA,
            uint nfunB,
            unsigned int duplication = DEFAULT_DUPLICATION )  [inline]
```

Co-evolution (joint gain or loss)

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

Definition at line 907 of file phylo.hpp.

#### 5.4.2.3 counter_gains()

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

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

Definition at line 194 of file phylo.hpp.

#### 5.4.2.4 counter_gains_from_0()

```
void counter_gains_from_0 (
            PhyloCounters * counters,
            std::vector< uint > nfun,
            unsigned int duplication = DEFAULT_DUPLICATION )  [inline]
```

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

Needs to specify function a.

Definition at line 1673 of file phylo.hpp.

#### 5.4.2.5 counter_gains_k_offspring()

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

k genes gain function nfun

Definition at line 255 of file phylo.hpp.

#### 5.4.2.6 counter_genes_changing()

```
void counter_genes_changing (
            PhyloCounters * counters,
            unsigned int duplication = DEFAULT_DUPLICATION )  [inline]
```

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

Definition at line 328 of file phylo.hpp.

#### 5.4.2.7 counter_k_genes_changing()

```
void counter_k_genes_changing (
            PhyloCounters * counters,
            unsigned int k,
            unsigned int duplication = DEFAULT_DUPLICATION )  [inline]
```

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

Definition at line 1435 of file phylo.hpp.

#### 5.4.2.8 counter_less_than_p_prop_genes_changing()

```
void counter_less_than_p_prop_genes_changing (
            PhyloCounters * counters,
            double p,
            unsigned int duplication = DEFAULT_DUPLICATION )  [inline]
```

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

$<$ How many genes diverge the parent

Definition at line 1556 of file phylo.hpp.

#### 5.4.2.9 counter_longest()

```
void counter_longest (
            PhyloCounters * counters,
            unsigned int duplication = DEFAULT_DUPLICATION )  [inline]
```

Longest branch mutates (either by gain or by loss)

Definition at line 965 of file phylo.hpp.

#### 5.4.2.10 counter_loss()

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

Total count of losses for an specific function.

Definition at line 704 of file phylo.hpp.

#### 5.4.2.11 counter_maxfuns()

```
void counter_maxfuns (
            PhyloCounters * counters,
            uint lb,
            uint ub,
            unsigned int duplication = DEFAULT_DUPLICATION )  [inline]
```

Cap the number of functions per gene.

Definition at line 633 of file phylo.hpp.

#### 5.4.2.12 counter_neofun()

```
void counter_neofun (
            PhyloCounters * counters,
            uint nfunA,
            uint nfunB,
            unsigned int duplication = DEFAULT_DUPLICATION )  [inline]
```

Total number of neofunctionalization events.

Needs to specify pairs of function.

Definition at line 1136 of file phylo.hpp.

#### 5.4.2.13 counter_neofun_a2b()

```
void counter_neofun_a2b (
            PhyloCounters * counters,
            uint nfunA,
            uint nfunB,
            unsigned int duplication = DEFAULT_DUPLICATION )  [inline]
```

Total number of neofunctionalization events.

Needs to specify pairs of function.

Definition at line 1217 of file phylo.hpp.

#### 5.4.2.14 counter_overall_changes()

```
void counter_overall_changes (
            PhyloCounters * counters,
            unsigned int duplication = DEFAULT_DUPLICATION )  [inline]
```

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

Definition at line 757 of file phylo.hpp.

#### 5.4.2.15 counter_overall_gains()

```
void counter_overall_gains (
            PhyloCounters * counters,
            unsigned int duplication = DEFAULT_DUPLICATION )  [inline]
```

Overall functional gains.

Total number of gains (irrespective of the function).

Definition at line 155 of file phylo.hpp.

#### 5.4.2.16 counter_overall_gains_from_0()

```
void counter_overall_gains_from_0 (
            PhyloCounters * counters,
            unsigned int duplication = DEFAULT_DUPLICATION )  [inline]
```

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

Needs to specify function a.

Definition at line 1740 of file phylo.hpp.

#### 5.4.2.17 counter_overall_loss()

```
void counter_overall_loss (
            PhyloCounters * counters,
            unsigned int duplication = DEFAULT_DUPLICATION )  [inline]
```

Overall functional loss.

Definition at line 584 of file phylo.hpp.

#### 5.4.2.18 counter_pairwise_first_gain()

```
void counter_pairwise_first_gain (
            PhyloCounters * counters,
            uint nfunA,
            uint nfunB,
            unsigned int duplication = DEFAULT_DUPLICATION )  [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 1992 of file phylo.hpp.

#### 5.4.2.19 counter_pairwise_overall_change()

```
void counter_pairwise_overall_change (
            PhyloCounters * counters,
            unsigned int duplication = DEFAULT_DUPLICATION )  [inline]
```

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

Needs to specify function a.

Definition at line 1789 of file phylo.hpp.

#### 5.4.2.20 counter_pairwise_preserving()

```
void counter_pairwise_preserving (
            PhyloCounters * counters,
            uint nfunA,
            uint nfunB,
            unsigned int duplication = DEFAULT_DUPLICATION )  [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 1852 of file phylo.hpp.

#### 5.4.2.21 counter_preserve_pseudogene()

```
void counter_preserve_pseudogene (
            PhyloCounters * counters,
            unsigned int nfunA,
            unsigned int nfunB,
            unsigned int duplication = DEFAULT_DUPLICATION )  [inline]
```

Keeps track of how many pairs of genes preserve pseudostate.

Definition at line 398 of file phylo.hpp.

#### 5.4.2.22 counter_prop_genes_changing()

```
void counter_prop_genes_changing (
            PhyloCounters * counters,
            unsigned int duplication = DEFAULT_DUPLICATION )  [inline]
```

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

Definition at line 481 of file phylo.hpp.

#### 5.4.2.23 counter_subfun()

```
void counter_subfun (
            PhyloCounters * counters,
            uint nfunA,
            uint nfunB,
            unsigned int duplication = DEFAULT_DUPLICATION )  [inline]
```

Total count of Sub-functionalization events.

It requires to specify data = {funA, funB}

Definition at line 817 of file phylo.hpp.

## 5.5 Phylo rules

Rules for phylogenetic modeling.

### Classes

- class PhyloRuleDynData

### Functions

- void rule_dyn_limit_changes (PhyloSupport *support, uint pos, uint lb, uint ub, unsigned int duplication=DEFAULT_DUPLICATIO

  *Overall functional gains.*

### 5.5.1 Detailed Description

Rules for phylogenetic modeling.

**Parameters**

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

### 5.5.2 Function Documentation

#### 5.5.2.1 rule_dyn_limit_changes()

```
void rule_dyn_limit_changes (
            PhyloSupport * support,
            uint pos,
            uint lb,
            uint ub,
            unsigned int duplication = DEFAULT_DUPLICATION )  [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 2123 of file phylo.hpp.

# Chapter 6

# Namespace Documentation

## 6.1 barry Namespace Reference

barry: Your go-to motif accountant

### Namespaces

- counters

  *Tree class and TreeIterator class.*

### 6.1.1 Detailed Description

barry: Your go-to motif accountant

## 6.2 barry::counters Namespace Reference

Tree class and TreeIterator class.

### Namespaces

- network
- phylo

### 6.2.1 Detailed Description

Tree class and TreeIterator class.

## 6.3 barry::counters::network Namespace Reference

## 6.4 barry::counters::phylo Namespace Reference

## 6.5 CHECK Namespace Reference

Integer constants used to specify which cell should be check.

### Variables

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

### 6.5.1 Detailed Description

Integer constants used to specify which cell should be check.

### 6.5.2 Variable Documentation

#### 6.5.2.1 BOTH

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

Definition at line 28 of file typedefs.hpp.

#### 6.5.2.2 NONE

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

Definition at line 29 of file typedefs.hpp.

#### 6.5.2.3 ONE

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

Definition at line 30 of file typedefs.hpp.

#### 6.5.2.4 TWO

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

Definition at line 31 of file typedefs.hpp.

## 6.6 EXISTS Namespace Reference

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

### Variables

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

### 6.6.1 Detailed Description

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

### 6.6.2 Variable Documentation

#### 6.6.2.1 AS_ONE

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

Definition at line 46 of file typedefs.hpp.

#### 6.6.2.2 AS_ZERO

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

Definition at line 45 of file typedefs.hpp.

#### 6.6.2.3 BOTH

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

Definition at line 39 of file typedefs.hpp.

#### 6.6.2.4 NONE

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

Definition at line 40 of file typedefs.hpp.

#### 6.6.2.5 ONE

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

Definition at line 41 of file typedefs.hpp.

#### 6.6.2.6 TWO

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

Definition at line 42 of file typedefs.hpp.

#### 6.6.2.7 UKNOWN

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

Definition at line 44 of file typedefs.hpp.

# Chapter 7

# Class Documentation

## 7.1 BArray< Cell_Type, Data_Type > Class Template Reference

Baseline class for binary arrays.

`#include <barray-bones.hpp>`

### Public Member Functions

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

  *Get the edgelist.*
- void transpose ()
- void clear (bool hard=true)
- void resize (uint N_, uint M_)
- void reserve ()
- void print (const 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 unsigned int vector ranging from 0 to M_* |
| target | *When `true` tries to add repeated observations.* |

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

- void set_data (Data_Type *data_, bool delete_data_=false)
  *Set the data object.*
- Data_Type * D ()
- const Data_Type * D () const
- 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 (uint i, uint j, bool check_bounds=true) const
- uint nrow () const noexcept
- uint ncol () const noexcept
- uint nnozero () const noexcept
- Cell< Cell_Type > default_val () const

### Cell-wise insertion/deletion

***Parameters***

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

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

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

**Column/row wise interchange**

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

**Arithmetic operators**

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

## Public Attributes

- bool visited = false

## Friends

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

### 7.1.1 Detailed Description

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

Baseline class for binary arrays.

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

**Template Parameters**

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

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

### 7.1.2 Constructor & Destructor Documentation

#### 7.1.2.1 BArray() [1/6]

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

Zero-size array.

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

#### 7.1.2.2 BArray() [2/6]

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

Empty array.

Definition at line 66 of file barray-bones.hpp.

#### 7.1.2.3 BArray() [3/6]

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

Edgelist with data.

#### 7.1.2.4 BArray() [4/6]

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

Edgelist with no data (simpler)

#### 7.1.2.5 BArray() [5/6]

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

Copy constructor.

#### 7.1.2.6 BArray() [6/6]

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

Move operator.

#### 7.1.2.7 ~BArray()

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

### 7.1.3 Member Function Documentation

#### 7.1.3.1 clear()

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

#### 7.1.3.2 col()

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

#### 7.1.3.3 D() [1/2]

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

#### 7.1.3.4 D() [2/2]

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

#### 7.1.3.5 default_val()

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

#### 7.1.3.6 flush_data()

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

#### 7.1.3.7 get_cell()

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

#### 7.1.3.8 get_col_vec() [1/2]

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

#### 7.1.3.9 get_col_vec() [2/2]

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

#### 7.1.3.10 get_entries()

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

Get the edgelist.

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

**Returns**

Entries<Cell_Type>

#### 7.1.3.11 get_row_vec() [1/2]

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

#### 7.1.3.12 get_row_vec() [2/2]

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

#### 7.1.3.13 insert_cell() [1/3]

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

#### 7.1.3.14 insert_cell() [2/3]

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

#### 7.1.3.15 insert_cell() [3/3]

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

#### 7.1.3.16 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 232 of file barray-bones.hpp.

#### 7.1.3.17 is_empty()

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

#### 7.1.3.18 ncol()

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

#### 7.1.3.19 nnozero()

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

#### 7.1.3.20 nrow()

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

#### 7.1.3.21 operator()() [1/2]

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

#### 7.1.3.22 operator()() [2/2]

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

#### 7.1.3.23 operator*=()

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

#### 7.1.3.24 operator+=() [1/3]

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

#### 7.1.3.25 operator+=() [2/3]

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

#### 7.1.3.26 operator+=() [3/3]

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

#### 7.1.3.27 operator-=() [1/3]

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

#### 7.1.3.28 operator-=() [2/3]

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

#### 7.1.3.29 operator-=() [3/3]

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

#### 7.1.3.30 operator/=()

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

#### 7.1.3.31 operator=() [1/2]

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

Move assignment.

#### 7.1.3.32 operator=() [2/2]

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

Assignment constructor.

#### 7.1.3.33 operator==()

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

#### 7.1.3.34 out_of_range()

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

#### 7.1.3.35 print()

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

#### 7.1.3.36 reserve()

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

#### 7.1.3.37 resize()

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

#### 7.1.3.38 rm_cell()

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

#### 7.1.3.39 row()

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

#### 7.1.3.40 set_data()

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

Set the data object.

**Parameters**

| | |
|---|---|
| *data_* | |
| *delete_↩ data_* | |

#### 7.1.3.41 swap_cells()

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

#### 7.1.3.42 swap_cols()

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

#### 7.1.3.43 swap_rows()

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

#### 7.1.3.44 toggle_cell()

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

#### 7.1.3.45 toggle_lock()

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

#### 7.1.3.46 transpose()

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

#### 7.1.3.47 zero_col()

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

#### 7.1.3.48 zero_row()

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

### 7.1.4 Friends And Related Function Documentation

#### 7.1.4.1 BArrayCell< Cell_Type, Data_Type >

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

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

#### 7.1.4.2 BArrayCell_const< Cell_Type, Data_Type >

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

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

### 7.1.5 Member Data Documentation

#### 7.1.5.1 visited

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

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

Definition at line 48 of file barray-bones.hpp.

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

- include/barry/barray-bones.hpp

## 7.2 BArrayCell< Cell_Type, Data_Type > Class Template Reference

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

### Public Member Functions

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

### 7.2.1 Detailed Description

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

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

### 7.2.2 Constructor & Destructor Documentation

#### 7.2.2.1 BArrayCell()

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

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

#### 7.2.2.2 ~BArrayCell()

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

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

### 7.2.3 Member Function Documentation

#### 7.2.3.1 operator Cell_Type()

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

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

#### 7.2.3.2 operator*=()

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

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

#### 7.2.3.3 operator+=()

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

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

#### 7.2.3.4 operator-=()

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

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

#### 7.2.3.5 operator/=()

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

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

#### 7.2.3.6 operator=()

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

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

#### 7.2.3.7 operator==()

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

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

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

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

## 7.3 BArrayCell_const< Cell_Type, Data_Type > Class Template Reference

`#include <barraycell-bones.hpp>`

### Public Member Functions

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

### 7.3.1 Detailed Description

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

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

### 7.3.2 Constructor & Destructor Documentation

#### 7.3.2.1 BArrayCell_const()

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

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

#### 7.3.2.2 ~BArrayCell_const()

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

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

## 7.3.3 Member Function Documentation

### 7.3.3.1 operator Cell_Type()

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

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

### 7.3.3.2 operator"!=()

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

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

### 7.3.3.3 operator<()

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

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

### 7.3.3.4 operator<=()

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

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

### 7.3.3.5 operator==()

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

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

#### 7.3.3.6 operator>()

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

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

#### 7.3.3.7 operator>=()

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

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

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

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

## 7.4 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 (uint i, uint j) const
- Cell_Type get_cell (uint i, uint j, bool check_bounds=true) const
- std::vector< Cell_Type > get_col_vec (uint i, bool check_bounds=true) const
- std::vector< Cell_Type > get_row_vec (uint i, bool check_bounds=true) const
- void get_col_vec (std::vector< Cell_Type > *x, uint i, bool check_bounds=true) const
- void get_row_vec (std::vector< Cell_Type > *x, uint i, bool check_bounds=true) const
- BArrayDenseRow< Cell_Type, Data_Type > & row (uint i, bool check_bounds=true)
- const BArrayDenseRow_const< Cell_Type, Data_Type > row (uint i, bool check_bounds=true) const
- BArrayDenseCol< Cell_Type, Data_Type > & col (uint j, bool check_bounds=true)
- const BArrayDenseCol_const< Cell_Type, Data_Type > col (uint j, bool check_bounds=true) const
- Entries< Cell_Type > get_entries () const

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

#### Constructors

*Parameters*

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

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

### Queries

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

*Parameters*

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

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

### Cell-wise insertion/deletion

*Parameters*

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

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

**Column/row wise interchange**

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

**Arithmetic operators**

- 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 >

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

**Template Parameters**

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

Definition at line 34 of file barraydense-bones.hpp.

## 7.4.2 Constructor & Destructor Documentation

### 7.4.2.1 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.

### 7.4.2.2 BArrayDense() [2/6]

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

Empty array.

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

### 7.4.2.3 BArrayDense() [3/6]

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

Edgelist with data.

#### 7.4.2.4 BArrayDense() [4/6]

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

Edgelist with no data (simpler)

#### 7.4.2.5 BArrayDense() [5/6]

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

Copy constructor.

#### 7.4.2.6 BArrayDense() [6/6]

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

Move operator.

#### 7.4.2.7 ~BArrayDense()

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

### 7.4.3 Member Function Documentation

#### 7.4.3.1 clear()

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

#### 7.4.3.2 col() [1/2]

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

Definition at line 482 of file barraydense-meat.hpp.

#### 7.4.3.3 col() [2/2]

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

Definition at line 468 of file barraydense-meat.hpp.

#### 7.4.3.4 colsum()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
const Cell_Type BArrayDense< Cell_Type, Data_Type >::colsum (
            unsigned int i ) const
```

#### 7.4.3.5 D() [1/2]

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

#### 7.4.3.6 D() [2/2]

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

#### 7.4.3.7 default_val()

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

#### 7.4.3.8 get_cell()

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

#### 7.4.3.9 get_col_vec() [1/2]

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

#### 7.4.3.10 get_col_vec() [2/2]

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

#### 7.4.3.11 get_data()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
const std::vector< Cell_Type >& BArrayDense< Cell_Type, Data_Type >::get_data ( ) const
```

#### 7.4.3.12 get_entries()

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

Get the edgelist.

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

**Returns**

Entries<Cell_Type>

#### 7.4.3.13 get_row_vec() [1/2]

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

#### 7.4.3.14 get_row_vec() [2/2]

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

#### 7.4.3.15 insert_cell() [1/2]

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

#### 7.4.3.16 insert_cell() [2/2]

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

#### 7.4.3.17 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 254 of file barraydense-bones.hpp.

#### 7.4.3.18 is_empty()

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

#### 7.4.3.19 ncol()

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

#### 7.4.3.20 nnozero()

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

#### 7.4.3.21 nrow()

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

#### 7.4.3.22 operator()() [1/2]

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

#### 7.4.3.23 operator()() [2/2]

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

#### 7.4.3.24 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 )
```

#### 7.4.3.25 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 )
```

#### 7.4.3.26 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 )
```

#### 7.4.3.27 operator+=() [3/3]

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

#### 7.4.3.28 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 )
```

#### 7.4.3.29 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 )
```

#### 7.4.3.30 operator-=() [3/3]

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

#### 7.4.3.31 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 )
```

#### 7.4.3.32 operator=() [1/2]

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

Move assignment.

#### 7.4.3.33 operator=() [2/2]

```
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 > & Array_ )
```

Assignment constructor.

#### 7.4.3.34 operator==()

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

#### 7.4.3.35 out_of_range()

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

#### 7.4.3.36 print()

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

#### 7.4.3.37 reserve()

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

#### 7.4.3.38 resize()

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

#### 7.4.3.39 rm_cell()

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

#### 7.4.3.40 row() [1/2]

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

#### 7.4.3.41 row() [2/2]

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

#### 7.4.3.42 rowsum()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
const Cell_Type BArrayDense< Cell_Type, Data_Type >::rowsum (
            unsigned int i ) const
```

#### 7.4.3.43 set_data()

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

Set the data object.

**Parameters**

| | |
|---|---|
| *data_* | |
| *delete_↩ data_* | |

#### 7.4.3.44 swap_cells()

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

#### 7.4.3.45 swap_cols()

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

#### 7.4.3.46 swap_rows()

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

#### 7.4.3.47 toggle_cell()

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

#### 7.4.3.48 toggle_lock()

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

#### 7.4.3.49 transpose()

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

#### 7.4.3.50 zero_col()

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

#### 7.4.3.51 zero_row()

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

### 7.4.4 Friends And Related Function Documentation

#### 7.4.4.1 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.

#### 7.4.4.2 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.

#### 7.4.4.3 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.

#### 7.4.4.4 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.

#### 7.4.4.5 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.

### 7.4.5 Member Data Documentation

#### 7.4.5.1 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 64 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

## 7.5 BArrayDenseCell< Cell_Type, Data_Type > Class Template Reference

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

### Public Member Functions

- BArrayDenseCell (BArrayDense< Cell_Type, Data_Type > *Array_, uint i_, uint j_, bool check_bounds=true)
- ~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 >

### 7.5.1 Detailed Description

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

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

### 7.5.2 Constructor & Destructor Documentation

#### 7.5.2.1 BArrayDenseCell()

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

Definition at line 27 of file barraydensecell-bones.hpp.

#### 7.5.2.2 ~BArrayDenseCell()

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

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

### 7.5.3 Member Function Documentation

#### 7.5.3.1 operator Cell_Type()

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

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

#### 7.5.3.2 operator*=()

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

Definition at line 37 of file barraydensecell-meat.hpp.

#### 7.5.3.3 operator+=()

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

Definition at line 19 of file barraydensecell-meat.hpp.

#### 7.5.3.4 operator-=()

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

Definition at line 28 of file barraydensecell-meat.hpp.

#### 7.5.3.5 operator/=()

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

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

#### 7.5.3.6 operator=()

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

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

#### 7.5.3.7 operator==()

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

Definition at line 61 of file barraydensecell-meat.hpp.

### 7.5.4 Friends And Related Function Documentation

#### 7.5.4.1 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.

#### 7.5.4.2 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.

#### 7.5.4.3 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

## 7.6 BArrayDenseCell_const< Cell_Type, Data_Type > Class Template Reference

### 7.6.1 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

## 7.7 BArrayDenseCol< Cell_Type, Data_Type > Class Template Reference

`#include <barraydensecol-bones.hpp>`

### Public Member Functions

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

### Friends

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

### 7.7.1 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.

### 7.7.2 Constructor & Destructor Documentation

#### 7.7.2.1 BArrayDenseCol()

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

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

### 7.7.3 Member Function Documentation

#### 7.7.3.1 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.

#### 7.7.3.2 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.

#### 7.7.3.3 operator()()

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

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

#### 7.7.3.4 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.

### 7.7.4 Friends And Related Function Documentation

#### 7.7.4.1 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.

#### 7.7.4.2 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.

#### 7.7.4.3 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-meat.hpp
- include/barry/barraydensecol-bones.hpp

## 7.8 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_, unsigned int j)
- Col_type< Cell_Type >::iterator begin ()
- Col_type< Cell_Type >::iterator end ()
- size_t size () const noexcept
- const std::pair< unsigned int, Cell_Type * > operator() (unsigned int i) const

## Friends

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

### 7.8.1 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.

### 7.8.2 Constructor & Destructor Documentation

#### 7.8.2.1 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_,
            unsigned int j )  [inline]
```

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

### 7.8.3 Member Function Documentation

#### 7.8.3.1 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.

#### 7.8.3.2 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.

#### 7.8.3.3 operator()()

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

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

#### 7.8.3.4 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.

### 7.8.4 Friends And Related Function Documentation

#### 7.8.4.1 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.

#### 7.8.4.2 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-meat.hpp
- include/barry/barraydensecol-bones.hpp

## 7.9 BArrayDenseRow< Cell_Type, Data_Type > Class Template Reference

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

### Public Member Functions

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

### Friends

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

## 7.9.1 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.

## 7.9.2 Constructor & Destructor Documentation

### 7.9.2.1 BArrayDenseRow()

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

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

## 7.9.3 Member Function Documentation

### 7.9.3.1 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.

#### 7.9.3.2 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.

#### 7.9.3.3 operator()()

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

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

#### 7.9.3.4 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.

### 7.9.4 Friends And Related Function Documentation

#### 7.9.4.1 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.

#### 7.9.4.2 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.

#### 7.9.4.3 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

## 7.10 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_, unsigned int 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< unsigned int, Cell< Cell_Type > > operator() (unsigned int i) const

### Friends

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

### 7.10.1 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.

### 7.10.2 Constructor & Destructor Documentation

#### 7.10.2.1 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_,
            unsigned int i )  [inline]
```

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

### 7.10.3 Member Function Documentation

#### 7.10.3.1 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.

#### 7.10.3.2 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.

#### 7.10.3.3 operator()()

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

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

#### 7.10.3.4 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.

### 7.10.4 Friends And Related Function Documentation

#### 7.10.4.1 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.

#### 7.10.4.2 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

## 7.11 BArrayRow< Cell_Type, Data_Type > Class Template Reference

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

### Public Member Functions

- BArrayRow (BArray< Cell_Type, Data_Type > *Array_, uint 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

### 7.11.1 Detailed Description

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

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

## 7.11.2 Constructor & Destructor Documentation

### 7.11.2.1 BArrayRow()

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

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

### 7.11.2.2 ~BArrayRow()

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

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

## 7.11.3 Member Function Documentation

### 7.11.3.1 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
```

### 7.11.3.2 operator*=()

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

### 7.11.3.3 operator+=()

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

#### 7.11.3.4 operator-=()

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

#### 7.11.3.5 operator/=()

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

#### 7.11.3.6 operator=()

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

#### 7.11.3.7 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

## 7.12 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_, uint 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

### 7.12.1 Detailed Description

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

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

### 7.12.2 Constructor & Destructor Documentation

#### 7.12.2.1 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_,
            uint i_,
            bool check_bounds = true )  [inline]
```

Definition at line 51 of file barrayrow-bones.hpp.

#### 7.12.2.2 ~BArrayRow_const()

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

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

### 7.12.3 Member Function Documentation

#### 7.12.3.1 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
```

#### 7.12.3.2 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
```

#### 7.12.3.3 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
```

#### 7.12.3.4 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
```

#### 7.12.3.5 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
```

#### 7.12.3.6 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
```

#### 7.12.3.7 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

## 7.13 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_, uint &dim_ uint &i_, bool check_bounds=true)
  *Construct a new BArrayVector object.*
- ~BArrayVector ()
- bool is_row () const noexcept
- bool is_col () const noexcept
- uint 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

### 7.13.1 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 13 of file barrayvector-bones.hpp.

### 7.13.2 Constructor & Destructor Documentation

#### 7.13.2.1 BArrayVector()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
BArrayVector< Cell_Type, Data_Type >::BArrayVector (
            BArray< Cell_Type, Data_Type > * Array_,
            uint &dim_ uint & 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 34 of file barrayvector-bones.hpp.

#### 7.13.2.2 ~BArrayVector()

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

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

### 7.13.3 Member Function Documentation

#### 7.13.3.1 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 52 of file barrayvector-meat.hpp.

#### 7.13.3.2 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 66 of file barrayvector-meat.hpp.

#### 7.13.3.3 is_col()

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

Definition at line 36 of file barrayvector-meat.hpp.

#### 7.13.3.4 is_row()

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

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

#### 7.13.3.5 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 177 of file barrayvector-meat.hpp.

#### 7.13.3.6 operator*=()

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

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

#### 7.13.3.7 operator+=()

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

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

#### 7.13.3.8 operator-=()

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

Definition at line 114 of file barrayvector-meat.hpp.

#### 7.13.3.9 operator/=()

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

Definition at line 156 of file barrayvector-meat.hpp.

#### 7.13.3.10 operator=()

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

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

#### 7.13.3.11 operator==()

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

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

#### 7.13.3.12 size()

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

Definition at line 41 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

## 7.14 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_, uint &dim_ uint &i_, bool check_bounds=true)
- ~BArrayVector_const ()
- bool is_row () const noexcept
- bool is_col () const noexcept
- uint 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

### 7.14.1 Detailed Description

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

Definition at line 75 of file barrayvector-bones.hpp.

### 7.14.2 Constructor & Destructor Documentation

#### 7.14.2.1 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_,
            uint &dim_ uint & i_,
            bool check_bounds = true )  [inline]
```

Definition at line 88 of file barrayvector-bones.hpp.

#### 7.14.2.2 ~BArrayVector_const()

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

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

### 7.14.3 Member Function Documentation

#### 7.14.3.1 begin()

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

#### 7.14.3.2 end()

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

#### 7.14.3.3 is_col()

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

#### 7.14.3.4 is_row()

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

#### 7.14.3.5 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 214 of file barrayvector-meat.hpp.

#### 7.14.3.6 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 251 of file barrayvector-meat.hpp.

#### 7.14.3.7 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 256 of file barrayvector-meat.hpp.

#### 7.14.3.8 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 283 of file barrayvector-meat.hpp.

#### 7.14.3.9 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 224 of file barrayvector-meat.hpp.

#### 7.14.3.10 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 310 of file barrayvector-meat.hpp.

#### 7.14.3.11 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 317 of file barrayvector-meat.hpp.

#### 7.14.3.12 size()

```
template<typename Cell_Type = bool, typename Data_Type = bool>
uint 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

## 7.15 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 (unsigned int x)
- void add (int x)
- Cell ()
- Cell ()
- Cell ()

### Public Attributes

- Cell_Type value
- bool visited
- bool active

### 7.15.1 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 13 of file cell-bones.hpp.

### 7.15.2 Constructor & Destructor Documentation

#### 7.15.2.1 Cell() [1/7]

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

#### 7.15.2.2 Cell() [2/7]

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

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

#### 7.15.2.3 ~Cell()

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

Definition at line 21 of file cell-bones.hpp.

#### 7.15.2.4 Cell() [3/7]

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

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

#### 7.15.2.5 Cell() [4/7]

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

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

#### 7.15.2.6 Cell() [5/7]

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

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

#### 7.15.2.7 Cell() [6/7]

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

Definition at line 65 of file cell-meat.hpp.

#### 7.15.2.8 Cell() [7/7]

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

Definition at line 66 of file cell-meat.hpp.

### 7.15.3 Member Function Documentation

#### 7.15.3.1 add() [1/4]

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

#### 7.15.3.2 add() [2/4]

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

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

#### 7.15.3.3 add() [3/4]

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

Definition at line 54 of file cell-meat.hpp.

#### 7.15.3.4 add() [4/4]

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

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

#### 7.15.3.5 operator Cell_Type()

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

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

#### 7.15.3.6 operator"!=()

```
template<typename Cell_Type >
bool Cell< Cell_Type >::operator!= (
            const Cell< Cell_Type > & rhs ) const
```

Definition at line 33 of file cell-meat.hpp.

#### 7.15.3.7 operator=() [1/2]

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

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

#### 7.15.3.8 operator=() [2/2]

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

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

#### 7.15.3.9 operator==()

```
template<typename Cell_Type >
bool Cell< Cell_Type >::operator== (
            const Cell< Cell_Type > & rhs ) const
```

Definition at line 23 of file cell-meat.hpp.

### 7.15.4 Member Data Documentation

#### 7.15.4.1 active

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

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

#### 7.15.4.2 value

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

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

#### 7.15.4.3 visited

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

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

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

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

## 7.16 Cell_const< Cell_Type > Class Template Reference

### 7.16.1 Detailed Description

**template<typename Cell_Type>**
**class Cell_const< Cell_Type >**

Definition at line 8 of file barray-meat.hpp.

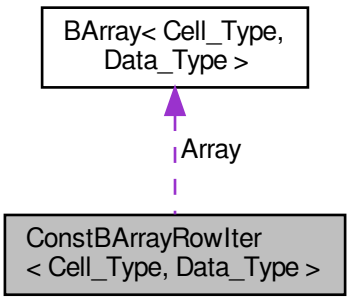
The documentation for this class was generated from the following file:

- include/barry/barray-meat.hpp

## 7.17 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

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

### 7.17.1 Detailed Description

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

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

### 7.17.2 Constructor & Destructor Documentation

#### 7.17.2.1 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.

#### 7.17.2.2 ~ConstBArrayRowIter()

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

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

### 7.17.3 Member Data Documentation

#### 7.17.3.1 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.

#### 7.17.3.2 current_col

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

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

#### 7.17.3.3 current_row

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

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

#### 7.17.3.4 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

## 7.18 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, uint i, uint j)
- double init (Array_Type &Array, uint i, uint 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_=nullptr, Data_Type *data_=nullptr, bool delete_data_=false, 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.*

## Public Attributes

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

### 7.18.1 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 38 of file counters-bones.hpp.

### 7.18.2 Constructor & Destructor Documentation

#### 7.18.2.1 Counter() [1/4]

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

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

#### 7.18.2.2 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_ = nullptr,
            Data_Type * data_ = nullptr,
            bool delete_data_ = false,
            std::string name_ = "",
            std::string desc_ = "" )  [inline]
```

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

#### 7.18.2.3 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.

#### 7.18.2.4 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.

#### 7.18.2.5 ~Counter()

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

Definition at line 77 of file counters-bones.hpp.

### 7.18.3 Member Function Documentation

#### 7.18.3.1 count()

```
template<typename Array_Type = BArray<>, typename Data_Type = bool>
double Counter< Array_Type, Data_Type >::count (
            Array_Type & Array,
            uint i,
            uint j )
```

#### 7.18.3.2 get_description()

```
template<typename Array_Type = BArray<>, typename Data_Type = bool>
std::string Counter< Array_Type, Data_Type >::get_description ( ) const
```

#### 7.18.3.3 get_name()

```
template<typename Array_Type = BArray<>, typename Data_Type = bool>
std::string Counter< Array_Type, Data_Type >::get_name ( ) const
```

#### 7.18.3.4 init()

```
template<typename Array_Type = BArray<>, typename Data_Type = bool>
double Counter< Array_Type, Data_Type >::init (
            Array_Type & Array,
            uint i,
            uint j )
```

#### 7.18.3.5 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.

#### 7.18.3.6 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.

### 7.18.4 Member Data Documentation

#### 7.18.4.1 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 41 of file counters-bones.hpp.

#### 7.18.4.2 data

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

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

#### 7.18.4.3 delete_data

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

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

#### 7.18.4.4 desc

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

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

#### 7.18.4.5 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 42 of file counters-bones.hpp.

#### 7.18.4.6 name

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

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

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

- include/barry/counters-bones.hpp

## 7.19 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[ ] (uint idx)

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

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

### 7.19.1 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 101 of file counters-bones.hpp.

### 7.19.2 Constructor & Destructor Documentation

#### 7.19.2.1 Counters() [1/3]

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

#### 7.19.2.2 ~Counters()

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

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

#### 7.19.2.3 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_* | |

#### 7.19.2.4 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_* | |

### 7.19.3 Member Function Documentation

#### 7.19.3.1 add_counter() [1/3]

```
template<typename Array_Type = BArray<>, typename Data_Type = bool>
void Counters< Array_Type, Data_Type >::add_counter (
            Counter< Array_Type, Data_Type > & counter )
```

#### 7.19.3.2 add_counter() [2/3]

```
template<typename Array_Type = BArray<>, typename Data_Type = bool>
void Counters< Array_Type, Data_Type >::add_counter (
            Counter< Array_Type, Data_Type > * counter )
```

#### 7.19.3.3 add_counter() [3/3]

```
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_ = nullptr,
            Data_Type * data_ = nullptr,
            bool delete_data_ = false,
            std::string name_ = "",
            std::string desc_ = "" )
```

#### 7.19.3.4 clear()

```
template<typename Array_Type = BArray<>, typename Data_Type = bool>
void Counters< Array_Type, Data_Type >::clear ( )
```

#### 7.19.3.5 get_descriptions()

```
template<typename Array_Type = BArray<>, typename Data_Type = bool>
std::vector< std::string > Counters< Array_Type, Data_Type >::get_descriptions ( ) const
```

#### 7.19.3.6 get_names()

```
template<typename Array_Type = BArray<>, typename Data_Type = bool>
std::vector< std::string > Counters< Array_Type, Data_Type >::get_names ( ) const
```

#### 7.19.3.7 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>

#### 7.19.3.8 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>&

#### 7.19.3.9 operator[]()

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

Returns a pointer to a particular counter.

**Parameters**

| | |
|---|---|
| *idx* | Id of the counter |

**Returns**

Counter<Array_Type,Data_Type>*

#### 7.19.3.10 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**

uint

Definition at line 161 of file counters-bones.hpp.

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

- include/barry/counters-bones.hpp

# 7.20 Entries< Cell_Type > Class Template Reference

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

`#include <typedefs.hpp>`

## Public Member Functions

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

## Public Attributes

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

### 7.20.1 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 79 of file typedefs.hpp.

### 7.20.2 Constructor & Destructor Documentation

#### 7.20.2.1 Entries() [1/2]

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

Definition at line 85 of file typedefs.hpp.

#### 7.20.2.2 Entries() [2/2]

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

Definition at line 86 of file typedefs.hpp.

#### 7.20.2.3 ~Entries()

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

Definition at line 93 of file typedefs.hpp.

### 7.20.3 Member Function Documentation

#### 7.20.3.1 resize()

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

Definition at line 95 of file typedefs.hpp.

### 7.20.4 Member Data Documentation

#### 7.20.4.1 source

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

Definition at line 81 of file typedefs.hpp.

#### 7.20.4.2 target

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

Definition at line 82 of file typedefs.hpp.

#### 7.20.4.3 val

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

Definition at line 83 of file typedefs.hpp.

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

- include/barry/typedefs.hpp

## 7.21 Flock Class Reference

A Flock is a group of Geese.

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

### Public Member Functions

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

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

  *Set the seed of the model.*
- void init (unsigned int bar_width=BARRY_PROGRESS_BAR_WIDTH)
- phylocounters::PhyloCounters * get_counters ()
- phylocounters::PhyloSupport * get_support_fun ()
- std::vector< std::vector< double > > * get_stats_support ()
- std::vector< std::vector< double > > * get_stats_target ()
- phylocounters::PhyloModel * get_model ()
- double likelihood_joint (const std::vector< double > &par, bool as_log=false, bool use_reduced_sequence=true)

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

  *Access the i-th geese element.*

#### Information about the model

- unsigned int nfuns () const noexcept
- unsigned int ntrees () const noexcept
- std::vector< unsigned int > nnodes () const noexcept
- std::vector< unsigned int > nleafs () const noexcept
- unsigned int nterms () const
- unsigned int support_size () const noexcept
- std::vector< std::string > colnames () const
- unsigned int 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
- unsigned int nfunctions = 0u
- bool initialized = false
- std::mt19937 rengine
- phylocounters::PhyloModel model = phylocounters::PhyloModel()

## 7.21.1 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.

## 7.21.2 Constructor & Destructor Documentation

### 7.21.2.1 Flock()

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

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

### 7.21.2.2 ~Flock()

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

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

## 7.21.3 Member Function Documentation

### 7.21.3.1 add_data()

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

Add a tree to the flock.

**Parameters**

| | |
|---|---|
| *annotations* | see Geese::Geese. |
| *geneid* | see Geese. |
| *parent* | see Geese. |
| *duplication* | see Geese. |

**Returns**

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

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

#### 7.21.3.2 colnames()

```
std::vector< std::string > Flock::colnames ( ) const  [inline]
```

Definition at line 224 of file flock-meat.hpp.

#### 7.21.3.3 get_counters()

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

Definition at line 100 of file flock-meat.hpp.

#### 7.21.3.4 get_model()

```
phylocounters::PhyloModel * Flock::get_model ( )  [inline]
```

Definition at line 131 of file flock-meat.hpp.

#### 7.21.3.5 get_stats_support()

```
std::vector< std::vector< double > > * Flock::get_stats_support ( )  [inline]
```

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

#### 7.21.3.6 get_stats_target()

```
std::vector< std::vector< double > > * Flock::get_stats_target ( )  [inline]
```

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

#### 7.21.3.7 get_support_fun()

```
phylocounters::PhyloSupport * Flock::get_support_fun ( )  [inline]
```

Definition at line 110 of file flock-meat.hpp.

#### 7.21.3.8 init()

```
void Flock::init (
            unsigned int bar_width = BARRY_PROGRESS_BAR_WIDTH )  [inline]
```

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

#### 7.21.3.9 likelihood_joint()

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

Returns the joint likelihood of the model.

**Parameters**

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

**Returns**

double

Definition at line 138 of file flock-meat.hpp.

#### 7.21.3.10 nfuns()

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

Definition at line 167 of file flock-meat.hpp.

#### 7.21.3.11 nleafs()

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

Definition at line 195 of file flock-meat.hpp.

#### 7.21.3.12 nnodes()

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

Definition at line 181 of file flock-meat.hpp.

#### 7.21.3.13 nterms()

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

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

#### 7.21.3.14 ntrees()

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

Definition at line 174 of file flock-meat.hpp.

#### 7.21.3.15 operator()()

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

Access the i-th geese element.

**Parameters**

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

**Returns**

Geese*

Definition at line 302 of file flock-meat.hpp.

#### 7.21.3.16 parse_polytomies()

```
unsigned int Flock::parse_polytomies (
            bool verb = true,
            std::vector< size_t > * dist = nullptr ) const  [inline], [noexcept]
```

Check polytomies and return the largest.

Definition at line 231 of file flock-meat.hpp.

#### 7.21.3.17 print()

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

Definition at line 258 of file flock-meat.hpp.

#### 7.21.3.18 set_seed()

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

Set the seed of the model.

**Parameters**

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

Definition at line 42 of file flock-meat.hpp.

#### 7.21.3.19 support_size()

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

Definition at line 217 of file flock-meat.hpp.

### 7.21.4 Member Data Documentation

#### 7.21.4.1 dat

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

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

#### 7.21.4.2 initialized

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

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

#### 7.21.4.3 model

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

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

#### 7.21.4.4 nfunctions

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

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

#### 7.21.4.5 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

## 7.22 FreqTable< T > Class Template Reference

Database of statistics.

```
#include <statsdb.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< double > &x) const

### 7.22.1 Detailed Description

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

Database of statistics.

This is mostly used in `Support`.

Definition at line 16 of file statsdb.hpp.

### 7.22.2 Constructor & Destructor Documentation

#### 7.22.2.1 FreqTable()

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

Definition at line 28 of file statsdb.hpp.

#### 7.22.2.2 ~FreqTable()

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

Definition at line 29 of file statsdb.hpp.

### 7.22.3 Member Function Documentation

#### 7.22.3.1 add()

```
template<typename T >
size_t FreqTable< T >::add (
            const std::vector< T > & x,
            size_t * h_precomp )  [inline]
```

Definition at line 53 of file statsdb.hpp.

#### 7.22.3.2 as_vector()

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

Definition at line 133 of file statsdb.hpp.

#### 7.22.3.3 clear()

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

Definition at line 162 of file statsdb.hpp.

#### 7.22.3.4 get_data()

```
template<typename T = double>
const std::vector< double >& FreqTable< T >::get_data ( ) const  [inline]
```

Definition at line 34 of file statsdb.hpp.

#### 7.22.3.5 get_index()

```
template<typename T = double>
const std::unordered_map<size_t,size_t>& FreqTable< T >::get_index ( ) const  [inline]
```

Definition at line 35 of file statsdb.hpp.

#### 7.22.3.6 make_hash()

```
template<typename T >
size_t FreqTable< T >::make_hash (
            const std::vector< double > & x ) const  [inline]
```

Definition at line 233 of file statsdb.hpp.

#### 7.22.3.7 print()

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

Definition at line 198 of file statsdb.hpp.

#### 7.22.3.8 reserve()

```
template<typename T >
void FreqTable< T >::reserve (
            size_t n,
            size_t k )  [inline]
```

Definition at line 176 of file statsdb.hpp.

#### 7.22.3.9 size()

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

Number of unique elements in the table. (.

**Returns**

size_t

Definition at line 225 of file statsdb.hpp.

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

- include/barry/statsdb.hpp

## 7.23 Geese Class Reference

Annotated Phylo Model.

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

### Public Member Functions

- ~Geese ()
- void init (unsigned int 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)
- double likelihood_exhaust (const std::vector< double > &par)
- std::vector< double > get_probabilities () const
- void set_seed (const unsigned int &s)
- std::vector< std::vector< unsigned int > > simulate (const std::vector< double > &par)
- std::vector< std::vector< double > > observed_counts ()
- void print_observed_counts ()
- void print () const

  *Prints information about the GEESE.*
- void init_node (Node &n)
- void update_annotations (unsigned int nodeid, std::vector< unsigned int > newann)
- std::vector< std::vector< bool > > get_states () const

  *Powerset of a gene's possible states.*
- std::vector< unsigned int > get_annotated_nodes () const

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

### Information about the model

*Parameters*

| | |
|---|---|
| verb | *When* `true` *it will print out information about the encountered polytomies.* |

- unsigned int nfuns () const noexcept
  *Number of functions analyzed.*
- unsigned int nnodes () const noexcept
  *Number of nodes (interior + leaf)*
- unsigned int nleafs () const noexcept
  *Number of leaf.*
- unsigned int nterms () const
  *Number of terms included.*
- unsigned int support_size () const noexcept
  *Number of unique sets of sufficient stats.*
- std::vector< unsigned int > nannotations () const noexcept
  *Number of annotations.*
- std::vector< std::string > colnames () const
  *Names of the terms in the model.*
- unsigned int 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< uint > &preorder)
- std::vector< std::vector< double > > predict_exhaust_backend (const std::vector< double > &par, const std::vector< uint > &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, unsigned int 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::PhyloCounters * get_counters ()
- phylocounters::PhyloModel * get_model ()
- phylocounters::PhyloSupport * get_support_fun ()

## Public Attributes

- unsigned int nfunctions
- std::map< unsigned int, Node > nodes
- barry::MapVec_type< unsigned int > map_to_nodes
- std::vector< std::vector< std::vector< size_t > > > pset_loc

  *Locations of columns.*
- std::vector< unsigned int > sequence
- std::vector< unsigned int > reduced_sequence
- bool initialized = false
- bool delete_rengine = false
- bool delete_support = false

### 7.23.1 Detailed Description

Annotated Phylo Model.

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

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

### 7.23.2 Constructor & Destructor Documentation

#### 7.23.2.1 Geese() [1/4]

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

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

#### 7.23.2.2 Geese() [2/4]

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

Definition at line 20 of file geese-meat-constructors.hpp.

#### 7.23.2.3 Geese() [3/4]

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

Definition at line 214 of file geese-meat-constructors.hpp.

#### 7.23.2.4 Geese() [4/4]

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

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

#### 7.23.2.5 ~Geese()

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

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

## 7.23.3 Member Function Documentation

### 7.23.3.1 calc_reduced_sequence()

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

Definition at line 324 of file geese-meat.hpp.

### 7.23.3.2 calc_sequence()

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

Definition at line 280 of file geese-meat.hpp.

### 7.23.3.3 colnames()

```
std::vector< std::string > Geese::colnames ( ) const  [inline]
```

Names of the terms in the model.

Definition at line 446 of file geese-meat.hpp.

### 7.23.3.4 get_annotated_nodes()

```
std::vector< unsigned int > Geese::get_annotated_nodes ( ) const  [inline]
```

Returns the ids of the nodes with at least one annotation.

Definition at line 662 of file geese-meat.hpp.

### 7.23.3.5 get_counters()

```
phylocounters::PhyloCounters * Geese::get_counters ( )  [inline]
```

Definition at line 645 of file geese-meat.hpp.

#### 7.23.3.6 get_model()

```
phylocounters::PhyloModel * Geese::get_model ( )  [inline]
```

Definition at line 650 of file geese-meat.hpp.

#### 7.23.3.7 get_probabilities()

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

Definition at line 372 of file geese-meat.hpp.

#### 7.23.3.8 get_rengine()

```
std::mt19937 * Geese::get_rengine ( )  [inline]
```

Definition at line 640 of file geese-meat.hpp.

#### 7.23.3.9 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 658 of file geese-meat.hpp.

#### 7.23.3.10 get_support_fun()

```
phylocounters::PhyloSupport * Geese::get_support_fun ( )  [inline]
```

Definition at line 654 of file geese-meat.hpp.

#### 7.23.3.11 inherit_support()

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

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

#### 7.23.3.12 init()

```
void Geese::init (
            unsigned int bar_width = BARRY_PROGRESS_BAR_WIDTH )  [inline]
```

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

#### 7.23.3.13 init_node()

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

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

#### 7.23.3.14 likelihood()

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

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

#### 7.23.3.15 likelihood_exhaust()

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

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

#### 7.23.3.16 nannotations()

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

Number of annotations.

Definition at line 437 of file geese-meat.hpp.

#### 7.23.3.17 nfuns()

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

Number of functions analyzed.

Definition at line 393 of file geese-meat.hpp.

#### 7.23.3.18 nleafs()

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

Number of leaf.

Definition at line 407 of file geese-meat.hpp.

#### 7.23.3.19 nnodes()

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

Number of nodes (interior + leaf)

Definition at line 400 of file geese-meat.hpp.

#### 7.23.3.20 nterms()

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

Number of terms included.

Definition at line 419 of file geese-meat.hpp.

#### 7.23.3.21 observed_counts()

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

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

#### 7.23.3.22 operator=() [1/2]

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

#### 7.23.3.23 operator=() [2/2]

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

#### 7.23.3.24 parse_polytomies()

```
unsigned int Geese::parse_polytomies (
            bool verb = true,
            std::vector< size_t > * dist = nullptr ) const  [inline], [noexcept]
```

Check polytomies and return the largest.

Definition at line 453 of file geese-meat.hpp.

#### 7.23.3.25 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 240 of file geese-meat-predict.hpp.

#### 7.23.3.26 predict_backend()

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

< True if the array belongs to the set

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

#### 7.23.3.27 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.

#### 7.23.3.28 predict_exhaust_backend()

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

Definition at line 47 of file geese-meat-predict_exhaust.hpp.

#### 7.23.3.29 predict_sim()

```
std::vector< std::vector< double > > Geese::predict_sim (
            const std::vector< double > & par,
            bool only_annotated = false,
            unsigned int nsims = 10000u )  [inline]
```

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

#### 7.23.3.30 print()

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

Prints information about the GEESE.

Definition at line 622 of file geese-meat.hpp.

#### 7.23.3.31 print_observed_counts()

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

Definition at line 559 of file geese-meat.hpp.

#### 7.23.3.32 set_seed()

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

Definition at line 4 of file geese-meat-simulate.hpp.

#### 7.23.3.33 simulate()

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

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

#### 7.23.3.34 support_size()

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

Number of unique sets of sufficient stats.

Definition at line 427 of file geese-meat.hpp.

#### 7.23.3.35 update_annotations()

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

Definition at line 251 of file geese-meat.hpp.

### 7.23.4 Member Data Documentation

#### 7.23.4.1 delete_rengine

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

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

#### 7.23.4.2 delete_support

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

Definition at line 119 of file geese-bones.hpp.

#### 7.23.4.3 initialized

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

Definition at line 117 of file geese-bones.hpp.

#### 7.23.4.4 map_to_nodes

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

Definition at line 109 of file geese-bones.hpp.

#### 7.23.4.5 nfunctions

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

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

#### 7.23.4.6 nodes

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

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

#### 7.23.4.7 pset_loc

```
std::vector< std::vector< std::vector< size_t > > > Geese::pset_loc
```

Locations of columns.

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

#### 7.23.4.8 reduced_sequence

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

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

#### 7.23.4.9 sequence

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

Definition at line 113 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

## 7.24 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>
```

## Public Member Functions

- void set_rengine (std::mt19937 *rengine_, bool delete_=false)
- void set_seed (unsigned int s)
- Model ()
- Model (uint size_)
- Model (const Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type > &Model_)
- Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type > & operator= (const Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type > &Model_)
- ~Model ()
- void store_psets () noexcept
- void set_keygen (std::function< std::vector< double >(const Array_Type &)> keygen_)
- std::vector< double > gen_key (const Array_Type &Array_)
- uint add_array (const Array_Type &Array_, bool force_new=false)

  *Adds an array to the support of not already included.*
- void print_stats (uint i) const
- void print () const

  *Prints information about the model.*
- Array_Type sample (const Array_Type &Array_, const std::vector< double > &params={})
- Array_Type sample (const uint &i, const std::vector< double > &params)
- double conditional_prob (const Array_Type &Array_, const std::vector< double > &params, unsigned int i, unsigned int 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< Array_Type, Data_Counter_Type > *counter)
- void add_counter (Counter_fun_type< Array_Type, Data_Counter_Type > count_fun_, Counter_fun_type< Array_Type, Data_Counter_Type > init_fun_=nullptr, Data_Counter_Type *data_=nullptr, bool delete_data_=false)
- void set_counters (Counters< Array_Type, Data_Counter_Type > *counters_)

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

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

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

### Likelihood functions.

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

***Parameters***

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

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

### Extract elements by index

***Parameters***

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

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

- unsigned int size () const noexcept
- unsigned int size_unique () const noexcept
- unsigned int nterms () const noexcept
- unsigned int 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< std::vector< double > > * get_stats_support ()
- std::vector< unsigned int > * get_arrays2support ()
- std::vector< std::vector< Array_Type > > * get_pset_arrays ()
- std::vector< std::vector< std::vector< double > > > * get_pset_stats ()

  *Statistics of the support(s)*
- std::vector< std::vector< double > > * get_pset_probs ()

### 7.24.1 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 46 of file model-bones.hpp.

### 7.24.2 Constructor & Destructor Documentation

#### 7.24.2.1 Model() [1/3]

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

#### 7.24.2.2 Model() [2/3]

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::Model (
            uint size_ )
```

#### 7.24.2.3 Model() [3/3]

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::Model (
            const Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >
& Model_ )
```

#### 7.24.2.4 ~Model()

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

Definition at line 147 of file model-bones.hpp.

### 7.24.3 Member Function Documentation

#### 7.24.3.1 add_array()

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
uint Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::add_array (
            const Array_Type & Array_,
            bool force_new = false )
```

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.

#### 7.24.3.2 add_counter() [1/3]

```
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_counter (
            Counter< Array_Type, Data_Counter_Type > & counter )
```

#### 7.24.3.3 add_counter() [2/3]

```
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_counter (
            Counter< Array_Type, Data_Counter_Type > * counter )
```

#### 7.24.3.4 add_counter() [3/3]

```
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_counter (
            Counter_fun_type< Array_Type, Data_Counter_Type > count_fun_,
            Counter_fun_type< Array_Type, Data_Counter_Type > init_fun_ = nullptr,
            Data_Counter_Type * data_ = nullptr,
            bool delete_data_ = false )
```

#### 7.24.3.5 add_rule() [1/3]

```
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< Array_Type, Data_Rule_Type > & rule )
```

#### 7.24.3.6 add_rule() [2/3]

```
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< Array_Type, Data_Rule_Type > * rule )
```

#### 7.24.3.7 add_rule() [3/3]

```
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_ = nullptr,
            bool delete_data_ = false )
```

#### 7.24.3.8 add_rule_dyn() [1/3]

```
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_dyn
(
            Rule< Array_Type, Data_Rule_Dyn_Type > & rule )
```

#### 7.24.3.9 add_rule_dyn() [2/3]

```
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_dyn
(
            Rule< Array_Type, Data_Rule_Dyn_Type > * rule )
```

#### 7.24.3.10 add_rule_dyn() [3/3]

```
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_dyn
(
            Rule_fun_type< Array_Type, Data_Rule_Dyn_Type > count_fun_,
            Data_Rule_Dyn_Type * data_ = nullptr,
            bool delete_data_ = false )
```

#### 7.24.3.11 colnames()

```
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 >::colnames ( ) const
```

#### 7.24.3.12 conditional_prob()

```
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 >::conditional↩
_prob (
            const Array_Type & Array_,
            const std::vector< double > & params,
            unsigned int i,
            unsigned int j )
```

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

#### 7.24.3.13 gen_key()

```
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 >::gen_key (
            const Array_Type & Array_ )
```

#### 7.24.3.14 get_arrays2support()

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
std::vector< unsigned int >* Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_↩
Rule_Dyn_Type >::get_arrays2support ( )
```

#### 7.24.3.15 get_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 >::get_counters ( )
```

#### 7.24.3.16 get_norm_const()

```
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 >::get_norm_↩
const (
            const std::vector< double > & params,
            const uint & i,
            bool as_log = false )
```

#### 7.24.3.17 get_pset()

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
const std::vector< Array_Type >* Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data↩
_Rule_Dyn_Type >::get_pset (
            const uint & i )
```

#### 7.24.3.18 get_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 >::get_pset_arrays ( )
```

#### 7.24.3.19 get_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 >::get_pset_probs ( )
```

#### 7.24.3.20 get_pset_stats() [1/2]

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

Statistics of the support(s)

#### 7.24.3.21 get_pset_stats() [2/2]

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
const std::vector< std::vector< double > >* Model< Array_Type, Data_Counter_Type, Data_Rule↩
_Type, Data_Rule_Dyn_Type >::get_pset_stats (
            const uint & i )
```

#### 7.24.3.22 get_rengine()

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
const std::mt19937* Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type
>::get_rengine ( ) const
```

#### 7.24.3.23 get_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 >::get_rules ( )
```

#### 7.24.3.24 get_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 >::get_rules_dyn ( )
```

#### 7.24.3.25 get_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< std::vector< double > >* Model< Array_Type, Data_Counter_Type, Data_Rule_Type,
Data_Rule_Dyn_Type >::get_stats_support ( )
```

#### 7.24.3.26 get_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 >::get_stats_target ( )
```

Raw pointers to the support and target statistics.

#### 7.24.3.27 get_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 >::get_support_fun ( )
```

#### 7.24.3.28 likelihood() [1/3]

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

#### 7.24.3.29 likelihood() [2/3]

```
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
(
            const std::vector< double > & params,
            const std::vector< double > & target_,
            const uint & i,
            bool as_log = false )
```

#### 7.24.3.30 likelihood() [3/3]

```
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
(
            const std::vector< double > & params,
            const uint & i,
            bool as_log = false )
```

#### 7.24.3.31 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 )
```

#### 7.24.3.32 nterms()

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
unsigned int Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >↩
::nterms ( ) const  [noexcept]
```

#### 7.24.3.33 operator=()

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
Model<Array_Type,Data_Counter_Type,Data_Rule_Type,Data_Rule_Dyn_Type>& Model< 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_ )
```

#### 7.24.3.34 print()

```
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 >::print ( )
const
```

Prints information about the model.

#### 7.24.3.35 print_stats()

```
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 >::print_stats (
            uint i ) const
```

#### 7.24.3.36 sample() [1/2]

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

#### 7.24.3.37 sample() [2/2]

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

#### 7.24.3.38 set_counters()

```
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_counters
(
            Counters< Array_Type, Data_Counter_Type > * counters_ )
```

#### 7.24.3.39 set_keygen()

```
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_keygen (
            std::function< std::vector< double >(const Array_Type &)> keygen_ )
```

#### 7.24.3.40 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 117 of file model-bones.hpp.

#### 7.24.3.41 set_rules()

```
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_rules (
            Rules< Array_Type, Data_Rule_Type > * rules_ )
```

#### 7.24.3.42 set_rules_dyn()

```
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_rules_dyn
(
            Rules< Array_Type, Data_Rule_Dyn_Type > * rules_ )
```

#### 7.24.3.43 set_seed()

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

Definition at line 127 of file model-bones.hpp.

#### 7.24.3.44 size()

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
unsigned int Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::size
( ) const  [noexcept]
```

#### 7.24.3.45 size_unique()

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
unsigned int Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >↩
::size_unique ( ) const  [noexcept]
```

#### 7.24.3.46 store_psets()

```
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 >::store_psets (
)  [noexcept]
```

#### 7.24.3.47 support_size()

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
unsigned int Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >↩
::support_size ( ) const  [noexcept]
```

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

- include/barry/model-bones.hpp

## 7.25 NetCounterData Class Reference

Data class used to store arbitrary uint or double vectors.

`#include <network.hpp>`

### Public Member Functions

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

### Public Attributes

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

### 7.25.1 Detailed Description

Data class used to store arbitrary uint or double vectors.

Definition at line 56 of file network.hpp.

### 7.25.2 Constructor & Destructor Documentation

#### 7.25.2.1 NetCounterData() [1/2]

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

Definition at line 62 of file network.hpp.

#### 7.25.2.2 NetCounterData() [2/2]

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

Definition at line 63 of file network.hpp.

#### 7.25.2.3 ~NetCounterData()

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

Definition at line 68 of file network.hpp.

### 7.25.3 Member Data Documentation

#### 7.25.3.1 indices

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

Definition at line 59 of file network.hpp.

#### 7.25.3.2 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

## 7.26 NetworkData Class Reference

Data class for Networks.

`#include <network.hpp>`

### Public Member Functions

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

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

  *Constructor using multiple attributes.*
- ~NetworkData ()

### Public Attributes

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

### 7.26.1 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.

### 7.26.2 Constructor & Destructor Documentation

#### 7.26.2.1 NetworkData() [1/3]

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

Definition at line 25 of file network.hpp.

#### 7.26.2.2 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.

#### 7.26.2.3 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.

#### 7.26.2.4 ~NetworkData()

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

Definition at line 51 of file network.hpp.

### 7.26.3 Member Data Documentation

#### 7.26.3.1 directed

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

Definition at line 22 of file network.hpp.

#### 7.26.3.2 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:

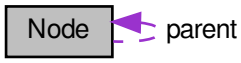
- include/barry/counters/network.hpp

## 7.27 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
- unsigned int noffspring () const noexcept
- bool is_leaf () const noexcept

#### Construct a new Node object

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

## Public Attributes

- unsigned int id

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

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

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

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

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

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

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

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

  *The probability of observing each state.*

### 7.27.1 Detailed Description

A single node for the model.

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

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

### 7.27.2 Constructor & Destructor Documentation

#### 7.27.2.1 Node() [1/5]

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

Definition at line 36 of file geese-node-bones.hpp.

#### 7.27.2.2 Node() [2/5]

```
Node::Node (
            unsigned int id_,
            unsigned int ord_,
            bool duplication_ )  [inline]
```

Definition at line 56 of file geese-node-bones.hpp.

#### 7.27.2.3 Node() [3/5]

```
Node::Node (
            unsigned int id_,
            unsigned int ord_,
            std::vector< unsigned int > annotations_,
            bool duplication_ )  [inline]
```

Definition at line 62 of file geese-node-bones.hpp.

#### 7.27.2.4 Node() [4/5]

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

Definition at line 69 of file geese-node-bones.hpp.

#### 7.27.2.5 Node() [5/5]

```
Node::Node (
            const Node & x )  [inline]
```

Definition at line 83 of file geese-node-bones.hpp.

#### 7.27.2.6 ~Node()

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

Definition at line 47 of file geese-node-bones.hpp.

### 7.27.3 Member Function Documentation

#### 7.27.3.1 get_parent()

```
int Node::get_parent ( ) const  [inline]
```

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

#### 7.27.3.2 is_leaf()

```
bool Node::is_leaf ( ) const  [inline], [noexcept]
```

Definition at line 109 of file geese-node-bones.hpp.

#### 7.27.3.3 noffspring()

```
unsigned int Node::noffspring ( ) const  [inline], [noexcept]
```

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

### 7.27.4 Member Data Documentation

#### 7.27.4.1 annotations

```
std::vector< unsigned int > Node::annotations
```

Observed annotations (only defined for Geese)

Definition at line 18 of file geese-node-bones.hpp.

#### 7.27.4.2 array

```
phylocounters::PhyloArray Node::array
```

Definition at line 17 of file geese-node-bones.hpp.

#### 7.27.4.3 arrays

```
std::vector< phylocounters::PhyloArray > Node::arrays = {}
```

Arrays given all possible states.

Definition at line 21 of file geese-node-bones.hpp.

#### 7.27.4.4 duplication

```
bool Node::duplication
```

Definition at line 19 of file geese-node-bones.hpp.

#### 7.27.4.5 id

```
unsigned int Node::id
```

Id of the node (as specified in the input)

Definition at line 14 of file geese-node-bones.hpp.

#### 7.27.4.6 narray

```
std::vector< unsigned int > Node::narray = {}
```

ID of the array in the model.

Definition at line 24 of file geese-node-bones.hpp.

#### 7.27.4.7 offspring

```
std::vector< Node* > Node::offspring = {}
```

Offspring nodes.

Definition at line 23 of file geese-node-bones.hpp.

#### 7.27.4.8 ord

```
unsigned int Node::ord
```

Order in which the node was created.

Definition at line 15 of file geese-node-bones.hpp.

#### 7.27.4.9 parent

```
Node* Node::parent = nullptr
```

Parent node.

Definition at line 22 of file geese-node-bones.hpp.

#### 7.27.4.10 probability

```
std::vector< double > Node::probability
```

The probability of observing each state.

Definition at line 28 of file geese-node-bones.hpp.

#### 7.27.4.11 subtree_prob

```
std::vector< double > Node::subtree_prob
```

Induced subtree probabilities.

Definition at line 27 of file geese-node-bones.hpp.

#### 7.27.4.12 visited

```
bool Node::visited = false
```

Definition at line 25 of file geese-node-bones.hpp.

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

- include/barry/models/geese/geese-node-bones.hpp

## 7.28 NodeData Class Reference

Data definition for the `PhyloArray` class.

`#include <phylo.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

### 7.28.1 Detailed Description

Data definition for the `PhyloArray` class.

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

This holds basic information about a given node.

Definition at line 38 of file phylo.hpp.

### 7.28.2 Constructor & Destructor Documentation

#### 7.28.2.1 NodeData()

```
NodeData::NodeData (
            const std::vector< double > & blengths_,
            const std::vector< bool > & states_,
            bool duplication_ = true )  [inline]
```

Definition at line 58 of file phylo.hpp.

### 7.28.3 Member Data Documentation

#### 7.28.3.1 blengths

```
std::vector< double > NodeData::blengths = {}
```

Branch length.

Definition at line 44 of file phylo.hpp.

#### 7.28.3.2 duplication

```
bool NodeData::duplication = true
```

Definition at line 54 of file phylo.hpp.

#### 7.28.3.3 states

```
std::vector< bool > NodeData::states = {}
```

State of the parent node.

Definition at line 49 of file phylo.hpp.

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

- include/barry/counters/phylo.hpp

## 7.29 PhyloCounterData Class Reference

```
#include <phylo.hpp>
```

### Public Member Functions

- PhyloCounterData (std::vector< uint > data_, std::vector< double > *counters_=nullptr)
- uint at (uint d)
- uint operator() (uint d)
- uint operator[] (uint d)
- void reserve (uint x)
- void push_back (uint x)
- void shrink_to_fit ()
- uint size ()
- std::vector< uint >::iterator begin ()
- std::vector< uint >::iterator end ()
- bool empty ()
- std::vector< double > * get_counters ()

### 7.29.1 Detailed Description

Definition at line 69 of file phylo.hpp.

### 7.29.2 Constructor & Destructor Documentation

#### 7.29.2.1 PhyloCounterData()

```
PhyloCounterData::PhyloCounterData (
            std::vector< uint > data_,
            std::vector< double > * counters_ = nullptr )  [inline]
```

Definition at line 75 of file phylo.hpp.

### 7.29.3 Member Function Documentation

#### 7.29.3.1 at()

```
uint PhyloCounterData::at (
            uint d )  [inline]
```

Definition at line 80 of file phylo.hpp.

#### 7.29.3.2 begin()

```
std::vector< uint >::iterator PhyloCounterData::begin ( )  [inline]
```

Definition at line 88 of file phylo.hpp.

#### 7.29.3.3 empty()

```
bool PhyloCounterData::empty ( )  [inline]
```

Definition at line 91 of file phylo.hpp.

#### 7.29.3.4 end()

```
std::vector< uint >::iterator PhyloCounterData::end ( )  [inline]
```

Definition at line 89 of file phylo.hpp.

#### 7.29.3.5 get_counters()

```
std::vector< double >* PhyloCounterData::get_counters ( )  [inline]
```

Definition at line 92 of file phylo.hpp.

#### 7.29.3.6 operator()()

```
uint PhyloCounterData::operator() (
            uint d )  [inline]
```

Definition at line 81 of file phylo.hpp.

#### 7.29.3.7 operator[]()

```
uint PhyloCounterData::operator[] (
            uint d )  [inline]
```

Definition at line 82 of file phylo.hpp.

#### 7.29.3.8 push_back()

```
void PhyloCounterData::push_back (
            uint x )  [inline]
```

Definition at line 84 of file phylo.hpp.

#### 7.29.3.9 reserve()

```
void PhyloCounterData::reserve (
            uint x )  [inline]
```

Definition at line 83 of file phylo.hpp.

#### 7.29.3.10 shrink_to_fit()

```
void PhyloCounterData::shrink_to_fit ( )  [inline]
```

Definition at line 85 of file phylo.hpp.

#### 7.29.3.11 size()

```
uint PhyloCounterData::size ( )  [inline]
```

Definition at line 86 of file phylo.hpp.

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

- include/barry/counters/phylo.hpp

## 7.30 PhyloRuleDynData Class Reference

`#include <phylo.hpp>`

### Public Member Functions

- PhyloRuleDynData (const std::vector< double > *counts_, uint pos_, uint lb_, uint ub_, uint duplication_)
- ~PhyloRuleDynData ()

### Public Attributes

- const std::vector< double > * counts
- uint pos
- uint lb
- uint ub
- uint duplication

### 7.30.1 Detailed Description

Definition at line 2095 of file phylo.hpp.

### 7.30.2 Constructor & Destructor Documentation

#### 7.30.2.1 PhyloRuleDynData()

```
PhyloRuleDynData::PhyloRuleDynData (
            const std::vector< double > * counts_,
            uint pos_,
            uint lb_,
            uint ub_,
            uint duplication_ )  [inline]
```

Definition at line 2102 of file phylo.hpp.

#### 7.30.2.2 ~PhyloRuleDynData()

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

Definition at line 2111 of file phylo.hpp.

### 7.30.3 Member Data Documentation

#### 7.30.3.1 counts

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

Definition at line 2097 of file phylo.hpp.

#### 7.30.3.2 duplication

```
uint PhyloRuleDynData::duplication
```

Definition at line 2101 of file phylo.hpp.

#### 7.30.3.3 lb

```
uint PhyloRuleDynData::lb
```

Definition at line 2099 of file phylo.hpp.

#### 7.30.3.4 pos

```
uint PhyloRuleDynData::pos
```

Definition at line 2098 of file phylo.hpp.

#### 7.30.3.5 ub

```
uint PhyloRuleDynData::ub
```

Definition at line 2100 of file phylo.hpp.

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

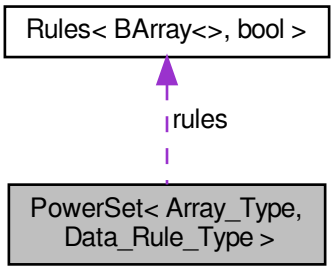
- include/barry/counters/phylo.hpp

## 7.31 PowerSet< Array_Type, Data_Rule_Type > Class Template Reference

Powerset of a binary array.

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

Collaboration diagram for PowerSet< Array_Type, Data_Rule_Type >:



## Public Member Functions

- void init_support ()
- void calc ()
- void reset (uint N_, uint M_)

### Construct and destroy a PowerSet object

- PowerSet ()
- PowerSet (uint N_, uint M_)
- PowerSet (const Array_Type &array)
- ~PowerSet ()

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

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

- void add_rule (Rule< Array_Type, Data_Rule_Type > &rule)
- void add_rule (Rule< Array_Type, Data_Rule_Type > *rule)
- void add_rule (Rule_fun_type< Array_Type, Data_Rule_Type > count_fun_, Data_Rule_Type *data_=nullptr, bool delete_data_=false)

### Getter functions

- const std::vector< Array_Type > * get_data_ptr () const
- std::vector< Array_Type > get_data () const
- std::vector< Array_Type >::iterator begin ()
- std::vector< Array_Type >::iterator end ()
- std::size_t size () const noexcept
- const Array_Type & operator[] (const unsigned int &i) const

## Public Attributes

- Array_Type EmptyArray
- std::vector< Array_Type > data
- Rules< Array_Type, Data_Rule_Type > * rules
- uint N
- uint M
- bool rules_deleted = false
- std::vector< size_t > coordinates_free
- std::vector< size_t > coordinates_locked
- size_t n_free
- size_t n_locked

## 7.31.1 Detailed Description

**template<typename Array_Type = BArray<>, typename Data_Rule_Type = bool>**
**class PowerSet< Array_Type, Data_Rule_Type >**

Powerset of a binary array.

**Template Parameters**

| | |
|---|---|
| *Array_Type* | |
| *Data_Rule_Type* | |

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

## 7.31.2 Constructor & Destructor Documentation

### 7.31.2.1 PowerSet() [1/3]

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

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

### 7.31.2.2 PowerSet() [2/3]

```
template<typename Array_Type = BArray<>, typename Data_Rule_Type = bool>
PowerSet< Array_Type, Data_Rule_Type >::PowerSet (
            uint N_,
            uint M_ )  [inline]
```

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

### 7.31.2.3 PowerSet() [3/3]

```
template<typename Array_Type , typename Data_Rule_Type >
PowerSet< Array_Type, Data_Rule_Type >::PowerSet (
            const Array_Type & array )  [inline]
```

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

### 7.31.2.4 ~PowerSet()

```
template<typename Array_Type , typename Data_Rule_Type >
PowerSet< Array_Type, Data_Rule_Type >::~PowerSet  [inline]
```

Definition at line 15 of file powerset-meat.hpp.

### 7.31.3 Member Function Documentation

#### 7.31.3.1 add_rule() [1/3]

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

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

#### 7.31.3.2 add_rule() [2/3]

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

Definition at line 184 of file powerset-meat.hpp.

#### 7.31.3.3 add_rule() [3/3]

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

Definition at line 194 of file powerset-meat.hpp.

#### 7.31.3.4 begin()

```
template<typename Array_Type = BArray<>, typename Data_Rule_Type = bool>
std::vector< Array_Type >::iterator PowerSet< Array_Type, Data_Rule_Type >::begin ( )  [inline]
```

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

#### 7.31.3.5 calc()

```
template<typename Array_Type , typename Data_Rule_Type >
void PowerSet< Array_Type, Data_Rule_Type >::calc  [inline]
```

Definition at line 147 of file powerset-meat.hpp.

#### 7.31.3.6 end()

```
template<typename Array_Type = BArray<>, typename Data_Rule_Type = bool>
std::vector< Array_Type >::iterator PowerSet< Array_Type, Data_Rule_Type >::end ( )  [inline]
```

Definition at line 77 of file powerset-bones.hpp.

#### 7.31.3.7 get_data()

```
template<typename Array_Type = BArray<>, typename Data_Rule_Type = bool>
std::vector< Array_Type > PowerSet< Array_Type, Data_Rule_Type >::get_data ( ) const  [inline]
```

Definition at line 75 of file powerset-bones.hpp.

#### 7.31.3.8 get_data_ptr()

```
template<typename Array_Type = BArray<>, typename Data_Rule_Type = bool>
const std::vector< Array_Type >* PowerSet< Array_Type, Data_Rule_Type >::get_data_ptr ( )
const  [inline]
```

Definition at line 74 of file powerset-bones.hpp.

#### 7.31.3.9 init_support()

```
template<typename Array_Type , typename Data_Rule_Type >
void PowerSet< Array_Type, Data_Rule_Type >::init_support  [inline]
```

Definition at line 21 of file powerset-meat.hpp.

#### 7.31.3.10 operator[]()

```
template<typename Array_Type = BArray<>, typename Data_Rule_Type = bool>
const Array_Type& PowerSet< Array_Type, Data_Rule_Type >::operator[] (
            const unsigned int & i ) const  [inline]
```

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

#### 7.31.3.11 reset()

```
template<typename Array_Type , typename Data_Rule_Type >
void PowerSet< Array_Type, Data_Rule_Type >::reset (
            uint N_,
            uint M_ )  [inline]
```

Definition at line 163 of file powerset-meat.hpp.

#### 7.31.3.12 size()

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

Definition at line 78 of file powerset-bones.hpp.

### 7.31.4 Member Data Documentation

#### 7.31.4.1 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 32 of file powerset-bones.hpp.

#### 7.31.4.2 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 33 of file powerset-bones.hpp.

#### 7.31.4.3 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 25 of file powerset-bones.hpp.

#### 7.31.4.4 EmptyArray

```
template<typename Array_Type = BArray<>, typename Data_Rule_Type = bool>
Array_Type PowerSet< Array_Type, Data_Rule_Type >::EmptyArray
```

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

#### 7.31.4.5 M

```
template<typename Array_Type = BArray<>, typename Data_Rule_Type = bool>
uint PowerSet< Array_Type, Data_Rule_Type >::M
```

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

#### 7.31.4.6 N

```
template<typename Array_Type = BArray<>, typename Data_Rule_Type = bool>
uint PowerSet< Array_Type, Data_Rule_Type >::N
```

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

#### 7.31.4.7 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 34 of file powerset-bones.hpp.

#### 7.31.4.8 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 35 of file powerset-bones.hpp.

#### 7.31.4.9 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 26 of file powerset-bones.hpp.

#### 7.31.4.10 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 29 of file powerset-bones.hpp.

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

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

## 7.32 Progress Class Reference

A simple progress bar.

```
#include <progress.hpp>
```

### Public Member Functions

- Progress (int n_, int width_)
- ~Progress ()
- void next ()
- void end ()

### 7.32.1 Detailed Description

A simple progress bar.

Definition at line 11 of file progress.hpp.

### 7.32.2 Constructor & Destructor Documentation

#### 7.32.2.1 Progress()

```
Progress::Progress (
            int n_,
            int width_ )  [inline]
```

Definition at line 30 of file progress.hpp.

#### 7.32.2.2 ~Progress()

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

Definition at line 23 of file progress.hpp.

### 7.32.3 Member Function Documentation

#### 7.32.3.1 end()

```
void Progress::end ( )  [inline]
```

Definition at line 52 of file progress.hpp.

#### 7.32.3.2 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

## 7.33 Rule< Array_Type, Data_Type > Class Template Reference

Rule for determining if a cell should be included in a sequence.

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

### Public Member Functions

- ~Rule ()
- Data_Type * D ()

  *Read/Write access to the data.*
- bool operator() (const Array_Type &a, uint i, uint j)

**Construct a new Rule object**

*Construct a new Rule object*

***Parameters***

| | |
|---|---|
| fun_ | *A function of type* `Rule_fun_type`. |
| dat_ | *Data pointer to be passed to* `fun_` |
| delete_↩ dat_ | *When* `true`, *the* `Rule` *destructor will delete the pointer, if defined.* |

- Rule ()
- Rule (Rule_fun_type< Array_Type, Data_Type > fun_, Data_Type *dat_=nullptr, bool delete_dat_=false)

## 7.33.1 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 22 of file rules-bones.hpp.

## 7.33.2 Constructor & Destructor Documentation

### 7.33.2.1 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.

### 7.33.2.2 Rule() [2/2]

```
template<typename Array_Type = BArray<>, typename Data_Type = bool>
Rule< Array_Type, Data_Type >::Rule (
            Rule_fun_type< Array_Type, Data_Type > fun_,
            Data_Type * dat_ = nullptr,
            bool delete_dat_ = false )  [inline]
```

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

#### 7.33.2.3 ~Rule()

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

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

### 7.33.3 Member Function Documentation

#### 7.33.3.1 D()

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

Read/Write access to the data.

#### 7.33.3.2 operator()()

```
template<typename Array_Type , typename Data_Type >
bool Rule< Array_Type, Data_Type >::operator() (
            const Array_Type & a,
            uint i,
            uint j )  [inline]
```

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

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

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

## 7.34 Rules< Array_Type, Data_Type > Class Template Reference

Vector of objects of class Rule.

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

### Public Member Functions

- Rules ()
- Rules (const Rules< Array_Type, Data_Type > &rules_)
- Rules< Array_Type, Data_Type > operator= (const Rules< Array_Type, Data_Type > &rules_)
- ~Rules ()
- uint size () const noexcept
- bool operator() (const Array_Type &a, uint i, uint j)

  *Check whether a given cell is free or locked.*
- void clear ()
- void get_seq (const Array_Type &a, std::vector< size_t > *free, std::vector< size_t > *locked=nullptr)

  *Computes the sequence of free and locked cells in an BArray.*

**Rule adding**

***Parameters***

| | |
|---|---|
| rule | |

- void add_rule (Rule< Array_Type, Data_Type > &rule)
- void add_rule (Rule< Array_Type, Data_Type > *rule)
- void add_rule (Rule_fun_type< Array_Type, Data_Type > rule_, Data_Type *data_=nullptr, bool delete_data_=false)

## 7.34.1 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 68 of file rules-bones.hpp.

## 7.34.2 Constructor & Destructor Documentation

### 7.34.2.1 Rules() [1/2]

```
template<typename Array_Type , typename Data_Type >
Rules< Array_Type, Data_Type >::Rules ( )  [inline]
```

Definition at line 75 of file rules-bones.hpp.

### 7.34.2.2 Rules() [2/2]

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

Definition at line 10 of file rules-meat.hpp.

#### 7.34.2.3 ~Rules()

```
template<typename Array_Type , typename Data_Type >
Rules< Array_Type, Data_Type >::~Rules ( )  [inline]
```

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

### 7.34.3 Member Function Documentation

#### 7.34.3.1 add_rule() [1/3]

```
template<typename Array_Type , typename Data_Type >
void Rules< Array_Type, Data_Type >::add_rule (
            Rule< Array_Type, Data_Type > & rule )  [inline]
```

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

#### 7.34.3.2 add_rule() [2/3]

```
template<typename Array_Type , typename Data_Type >
void Rules< Array_Type, Data_Type >::add_rule (
            Rule< Array_Type, Data_Type > * rule )  [inline]
```

Definition at line 79 of file rules-meat.hpp.

#### 7.34.3.3 add_rule() [3/3]

```
template<typename Array_Type , typename Data_Type >
void Rules< Array_Type, Data_Type >::add_rule (
            Rule_fun_type< Array_Type, Data_Type > rule_,
            Data_Type * data_ = nullptr,
            bool delete_data_ = false )  [inline]
```

Definition at line 89 of file rules-meat.hpp.

#### 7.34.3.4 clear()

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

Definition at line 127 of file rules-meat.hpp.

#### 7.34.3.5 get_seq()

```
template<typename Array_Type , typename Data_Type >
void Rules< Array_Type, Data_Type >::get_seq (
            const Array_Type & a,
            std::vector< 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 140 of file rules-meat.hpp.

#### 7.34.3.6 operator()()

```
template<typename Array_Type , typename Data_Type >
bool Rules< Array_Type, Data_Type >::operator() (
            const Array_Type & a,
            uint i,
            uint j )  [inline]
```

Check whether a given cell is free or locked.

**Parameters**

| | |
|---|---|
| *a* | A BArray object |
| *i* | row position |
| *j* | col position |

**Returns**

true If the cell is locked

false If the cell is free

Definition at line 111 of file rules-meat.hpp.

#### 7.34.3.7 operator=()

```
template<typename Array_Type , typename Data_Type >
Rules< Array_Type, Data_Type > Rules< Array_Type, Data_Type >::operator= (
            const Rules< Array_Type, Data_Type > & rules_ )
```

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

#### 7.34.3.8 size()

```
template<typename Array_Type , typename Data_Type >
uint Rules< Array_Type, Data_Type >::size ( ) const  [inline], [noexcept]
```

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

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

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

## 7.35 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 add_counter (Counter< Array_Type, Data_Type > f_)
- void set_counters (Counters< Array_Type, Data_Type > *counters_)
- void count_init (uint i, uint j)
  *Counter functions This function recurses through the entries of `Array` and at each step of adding a new cell it uses the functions to list the statistics.*
- void count_current (uint i, uint j)
- std::vector< double > count_all ()
- Counters< Array_Type, Data_Type > * get_counters ()
- std::vector< std::string > get_names () const
- std::vector< std::string > get_descriptions () const

### 7.35.1 Detailed Description

**template<typename Array_Type = BArray<>, typename Data_Type = bool>**
**class StatsCounter< Array_Type, Data_Type >**

Count stats for a single Array.

Users can a list of functions that can be used with this. The baseline set of arguments is a pointer to a binary array and a dataset to add the counts to.

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

## 7.35.2 Constructor & Destructor Documentation

### 7.35.2.1 StatsCounter() [1/3]

```
template<typename Array_Type = BArray<>, typename Data_Type = bool>
StatsCounter< Array_Type, Data_Type >::StatsCounter (
            const Array_Type * Array_ )  [inline]
```

Creator of a StatsCounter

**Parameters**

| | |
|---|---|
| *Array_* | A const pointer to a BArray. |

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

### 7.35.2.2 StatsCounter() [2/3]

```
template<typename Array_Type = BArray<>, typename Data_Type = bool>
StatsCounter< Array_Type, Data_Type >::StatsCounter (
            const StatsCounter< Array_Type, Data_Type > & counter )
```

Copy constructor.

**Parameters**

| | |
|---|---|
| *counter* | |

### 7.35.2.3 StatsCounter() [3/3]

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

Can be created without setting the array.

Definition at line 64 of file statscounter-bones.hpp.

### 7.35.2.4 ~StatsCounter()

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

## 7.35.3 Member Function Documentation

### 7.35.3.1 add_counter() [1/2]

```
template<typename Array_Type = BArray<>, typename Data_Type = bool>
void StatsCounter< Array_Type, Data_Type >::add_counter (
            Counter< Array_Type, Data_Type > * f_ )
```

### 7.35.3.2 add_counter() [2/2]

```
template<typename Array_Type = BArray<>, typename Data_Type = bool>
void StatsCounter< Array_Type, Data_Type >::add_counter (
            Counter< Array_Type, Data_Type > f_ )
```

### 7.35.3.3 count_all()

```
template<typename Array_Type , typename Data_Type >
std::vector< double > StatsCounter< Array_Type, Data_Type >::count_all  [inline]
```

Definition at line 110 of file statscounter-meat.hpp.

### 7.35.3.4 count_current()

```
template<typename Array_Type = BArray<>, typename Data_Type = bool>
void StatsCounter< Array_Type, Data_Type >::count_current (
            uint i,
            uint j )
```

### 7.35.3.5 count_init()

```
template<typename Array_Type = BArray<>, typename Data_Type = bool>
void StatsCounter< Array_Type, Data_Type >::count_init (
            uint i,
            uint j )
```

Counter functions This function recurses through the entries of `Array` and at each step of adding a new cell it uses the functions to list the statistics.

#### 7.35.3.6 get_counters()

```
template<typename Array_Type = BArray<>, typename Data_Type = bool>
Counters<Array_Type,Data_Type>* StatsCounter< Array_Type, Data_Type >::get_counters ( )
```

#### 7.35.3.7 get_descriptions()

```
template<typename Array_Type = BArray<>, typename Data_Type = bool>
std::vector< std::string > StatsCounter< Array_Type, Data_Type >::get_descriptions ( ) const
```

#### 7.35.3.8 get_names()

```
template<typename Array_Type = BArray<>, typename Data_Type = bool>
std::vector< std::string > StatsCounter< Array_Type, Data_Type >::get_names ( ) const
```

#### 7.35.3.9 reset_array()

```
template<typename Array_Type = BArray<>, typename Data_Type = bool>
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`. |

#### 7.35.3.10 set_counters()

```
template<typename Array_Type = BArray<>, typename Data_Type = bool>
void StatsCounter< Array_Type, Data_Type >::set_counters (
            Counters< Array_Type, Data_Type > * counters_ )
```

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

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

# 7.36 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 (uint N_, uint M_)
  *Constructor specifying the dimensions of the array (empty).*
- Support ()
- ~Support ()
- void init_support (std::vector< Array_Type > *array_bank=nullptr, std::vector< std::vector< double > > *stats_bank=nullptr)
- void calc (std::vector< Array_Type > *array_bank=nullptr, std::vector< std::vector< double > > *stats_bank=nullptr, unsigned int max_num_elements_=0u)
  *Computes the entire support.*
- std::vector< double > get_counts () const
- std::vector< double > * get_current_stats ()
  *List current statistics.*
- void print () const
- const FreqTable & 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 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 uint &i, const uint &j)

## Public Attributes

- uint N
- uint M
- bool delete_counters = true
- bool delete_rules = true
- bool delete_rules_dyn = true
- uint max_num_elements = BARRY_MAX_NUM_ELEMENTS
- std::vector< double > current_stats
- std::vector< 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

### 7.36.1 Detailed Description

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

Compute the support of sufficient statistics.

Given an array and a set of counters, this object iterates throughout the support set of the Array while at the same time computing the support of the sufficient statitics.

The members `rule` and `rule_dyn` allow constraining the support. The first will establish which cells of the array will be used to iterate, for example, in the case of social networks, self-loops are not allowed, so the entire diagonal would be fixed to zero, reducing the size of the support.

In the case of `rule_dyn`, the function will stablish dynamically whether the current state will be included in the counts or not. For example, this set of rules can be used to constrain the support to networks that have a prescribed degree sequence.

Definition at line 35 of file support-bones.hpp.

## 7.36.2 Constructor & Destructor Documentation

### 7.36.2.1 Support() [1/3]

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::Support (
            const Array_Type & Array_ )  [inline]
```

Constructor passing a reference Array.

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

### 7.36.2.2 Support() [2/3]

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::Support (
            uint N_,
            uint M_ )  [inline]
```

Constructor specifying the dimensions of the array (empty).

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

### 7.36.2.3 Support() [3/3]

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

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

### 7.36.2.4 ~Support()

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

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

## 7.36.3 Member Function Documentation

### 7.36.3.1 add_counter() [1/2]

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
void Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::add_counter
(
            Counter< Array_Type, Data_Counter_Type > * f_ )
```

### 7.36.3.2 add_counter() [2/2]

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
void Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::add_counter
(
            Counter< Array_Type, Data_Counter_Type > f_ )
```

### 7.36.3.3 add_rule() [1/2]

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
void Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::add_rule (
            Rule< Array_Type, Data_Rule_Type > * f_ )
```

### 7.36.3.4 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 Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::add_rule (
            Rule< Array_Type, Data_Rule_Type > f_ )
```

### 7.36.3.5 add_rule_dyn() [1/2]

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
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_ )
```

#### 7.36.3.6 add_rule_dyn() [2/2]

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
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_ )
```

#### 7.36.3.7 calc()

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
void Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::calc (
            std::vector< Array_Type > * array_bank = nullptr,
            std::vector< std::vector< double > > * stats_bank = nullptr,
            unsigned int max_num_elements_ = 0u )
```

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. |

#### 7.36.3.8 eval_rules_dyn()

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
bool Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::eval_↩
rules_dyn (
            const std::vector< double > & counts,
            const uint & i,
            const uint & j )
```

#### 7.36.3.9 get_counters()

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
Counters<Array_Type,Data_Counter_Type>* Support< Array_Type, Data_Counter_Type, Data_Rule_↩
Type, Data_Rule_Dyn_Type >::get_counters ( )
```

Vector of couter functions.

#### 7.36.3.10 get_counts()

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
std::vector< double > Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn↩
_Type >::get_counts ( ) const
```

#### 7.36.3.11 get_current_stats()

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
std::vector< double >* Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_↩
Dyn_Type >::get_current_stats ( )
```

List current statistics.

#### 7.36.3.12 get_data()

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
const FreqTable& Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type
>::get_data ( ) const
```

#### 7.36.3.13 get_rules()

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
Rules<Array_Type,Data_Rule_Type>* Support< Array_Type, Data_Counter_Type, Data_Rule_Type,
Data_Rule_Dyn_Type >::get_rules ( )
```

Vector of static rules (cells to iterate).

#### 7.36.3.14 get_rules_dyn()

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
Rules<Array_Type,Data_Rule_Dyn_Type>* Support< Array_Type, Data_Counter_Type, Data_Rule_Type,
Data_Rule_Dyn_Type >::get_rules_dyn ( )
```

Vector of dynamic rules (to include/exclude a realizaton).

#### 7.36.3.15 init_support()

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
void Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::init_↩
support (
            std::vector< Array_Type > * array_bank = nullptr,
            std::vector< std::vector< double > > * stats_bank = nullptr )
```

#### 7.36.3.16 print()

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
void Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::print ( )
const
```

#### 7.36.3.17 reset_array() [1/2]

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
void Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::reset_array
( )
```

#### 7.36.3.18 reset_array() [2/2]

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
void Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::reset_array
(
            const Array_Type & Array_ )
```

#### 7.36.3.19 set_counters()

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
void Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::set_↩
counters (
            Counters< Array_Type, Data_Counter_Type > * counters_ )
```

#### 7.36.3.20 set_rules()

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
void Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::set_rules (
            Rules< Array_Type, Data_Rule_Type > * rules_ )
```

#### 7.36.3.21 set_rules_dyn()

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
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_ )
```

### 7.36.4 Member Data Documentation

#### 7.36.4.1 change_stats

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
std::vector< double > Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn↩
_Type >::change_stats
```

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

#### 7.36.4.2 coordiantes_n_free

```
template<typename Array_Type = BArray<>, 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 71 of file support-bones.hpp.

#### 7.36.4.3 coordiantes_n_locked

```
template<typename Array_Type = BArray<>, 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 72 of file support-bones.hpp.

#### 7.36.4.4 coordinates_free

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
std::vector< size_t > Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn↩
_Type >::coordinates_free
```

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

#### 7.36.4.5 coordinates_locked

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
std::vector< size_t > Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn↩
_Type >::coordinates_locked
```

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

#### 7.36.4.6 current_stats

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
std::vector< double > Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn↩
_Type >::current_stats
```

Definition at line 68 of file support-bones.hpp.

#### 7.36.4.7 delete_counters

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
bool Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::delete_↩
counters = true
```

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

#### 7.36.4.8 delete_rules

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
bool Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::delete_↩
rules = true
```

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

#### 7.36.4.9 delete_rules_dyn

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
bool Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::delete_↩
rules_dyn = true
```

Definition at line 64 of file support-bones.hpp.

#### 7.36.4.10 hashes

```
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 > Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn↩
_Type >::hashes
```

Definition at line 74 of file support-bones.hpp.

#### 7.36.4.11 hashes_initialized

```
template<typename Array_Type = BArray<>, 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 75 of file support-bones.hpp.

#### 7.36.4.12 M

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

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

#### 7.36.4.13 max_num_elements

```
template<typename Array_Type = BArray<>, typename Data_Counter_Type = bool, typename Data_↩
Rule_Type = bool, typename Data_Rule_Dyn_Type = bool>
uint Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >::max_num_↩
elements = BARRY_MAX_NUM_ELEMENTS
```

Definition at line 65 of file support-bones.hpp.

#### 7.36.4.14 N

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

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

#### 7.36.4.15 n_counters

```
template<typename Array_Type = BArray<>, 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 76 of file support-bones.hpp.

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

- include/barry/support-bones.hpp

## 7.37 vecHasher< T > Struct Template Reference

```
#include <typedefs.hpp>
```

### Public Member Functions

- std::size_t operator() (std::vector< T > const &dat) const noexcept

### 7.37.1 Detailed Description

**template<typename T>**
**struct vecHasher< T >**

Definition at line 106 of file typedefs.hpp.

### 7.37.2 Member Function Documentation

#### 7.37.2.1 operator()()

```
template<typename T >
std::size_t vecHasher< T >::operator() (
            std::vector< T > const & dat ) const  [inline], [noexcept]
```

Definition at line 109 of file typedefs.hpp.

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

- include/barry/typedefs.hpp

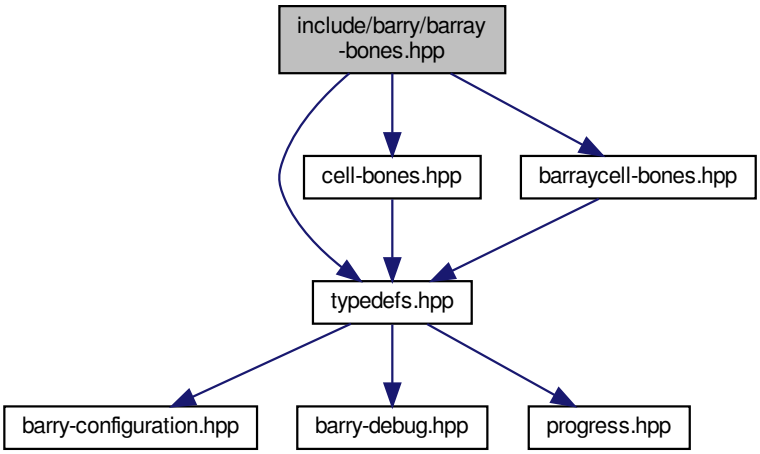
# Chapter 8

# File Documentation

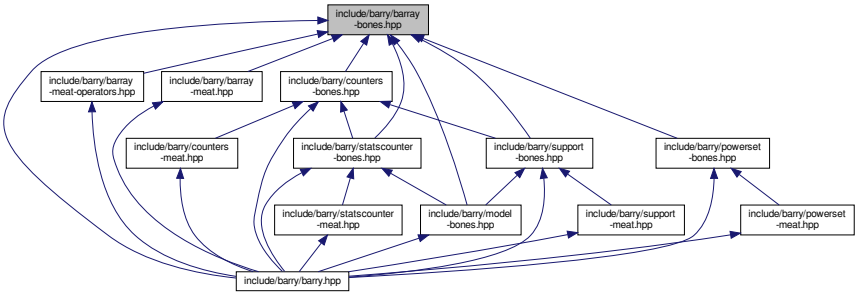
## 8.1 include/barry/barray-bones.hpp File Reference

```
#include "typedefs.hpp"
#include "cell-bones.hpp"
#include "barraycell-bones.hpp"
```

Include dependency graph for barray-bones.hpp:



This graph shows which files directly or indirectly include this file:




## Classes

- class BArray< Cell_Type, Data_Type >

  *Baseline class for binary arrays.*

## Macros

- #define BARRAY_BONES_HPP 1

### 8.1.1 Macro Definition Documentation

#### 8.1.1.1 BARRAY_BONES_HPP

```
#define BARRAY_BONES_HPP 1
```

Definition at line 8 of file barray-bones.hpp.

## 8.2 include/barry/barray-iterator.hpp File Reference

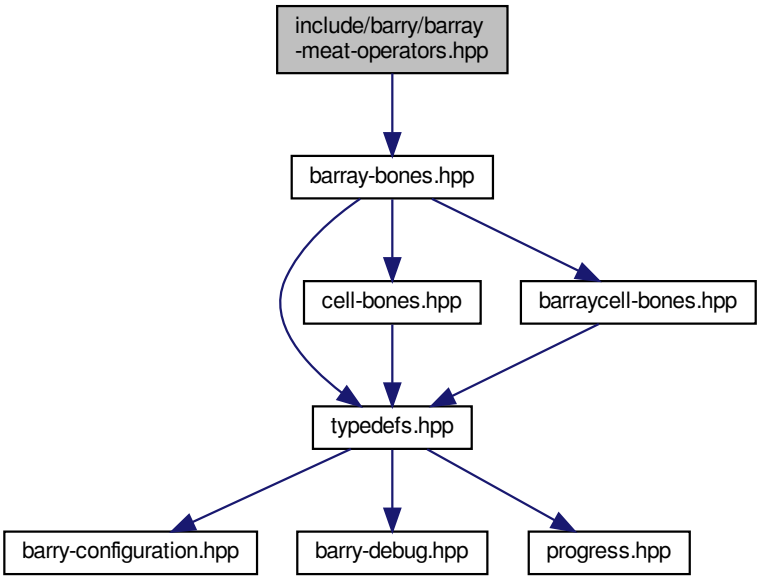
## Classes

- class ConstBArrayRowIter< Cell_Type, Data_Type >

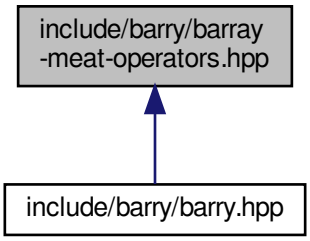
## 8.3 include/barry/barray-meat-operators.hpp File Reference

```
#include "barray-bones.hpp"
```

Include dependency graph for barray-meat-operators.hpp:




This graph shows which files directly or indirectly include this file:




### Macros

- #define BARRY_BARRAY_MEAT_OPERATORS_HPP 1
- #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 (uint i=0u;i< nrow();++i) for(uint j=0u = el[POS(i, j)]
- 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

### 8.3.1 Macro Definition Documentation

#### 8.3.1.1 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.

#### 8.3.1.2 BARRAY_TEMPLATE_ARGS

```
template BARRAY_TEMPLATE_ARGS( ) <typename Cell_Type, typename Data_Type>
```

Definition at line 9 of file barray-meat-operators.hpp.

#### 8.3.1.3 BARRAY_TYPE

```
template Data_Type BARRAY_TYPE( ) BArray<Cell_Type, Data_Type>
```

Definition at line 7 of file barray-meat-operators.hpp.

#### 8.3.1.4 BARRY_BARRAY_MEAT_OPERATORS_HPP

```
#define BARRY_BARRAY_MEAT_OPERATORS_HPP 1
```

Definition at line 5 of file barray-meat-operators.hpp.

#### 8.3.1.5 COL

```
#define COL(
              a ) this->el_ji[a]
```

Definition at line 15 of file barray-meat-operators.hpp.

#### 8.3.1.6 ROW

```
#define ROW(
              a ) this->el_ij[a]
```

Definition at line 14 of file barray-meat-operators.hpp.

### 8.3.2 Function Documentation

#### 8.3.2.1 BARRAY_TEMPLATE() [1/6]

```
BARRAY_TEMPLATE (
             BARRAY_TYPE()& ,
             operator*  ) const &
```

Definition at line 88 of file barray-meat-operators.hpp.

#### 8.3.2.2 BARRAY_TEMPLATE() [2/6]

```
BARRAY_TEMPLATE (
             BARRAY_TYPE()& ,
             operator+  ) const
```

#### 8.3.2.3 BARRAY_TEMPLATE() [3/6]

```
BARRAY_TEMPLATE (
            BARRAY_TYPE()& ,
            operator+  ) const &
```

Definition at line 46 of file barray-meat-operators.hpp.

#### 8.3.2.4 BARRAY_TEMPLATE() [4/6]

```
BARRAY_TEMPLATE (
            BARRAY_TYPE()& ,
            operator-  ) const
```

#### 8.3.2.5 BARRAY_TEMPLATE() [5/6]

```
BARRAY_TEMPLATE (
            BARRAY_TYPE()& ,
            operator-  ) const &
```

Definition at line 75 of file barray-meat-operators.hpp.

#### 8.3.2.6 BARRAY_TEMPLATE() [6/6]

```
BARRAY_TEMPLATE (
            BARRAY_TYPE()& ,
            operator/  ) const &
```

Definition at line 105 of file barray-meat-operators.hpp.

#### 8.3.2.7 BARRAY_TEMPLATE_ARGS()

```
template BARRAY_TEMPLATE_ARGS ( ) const &
```

#### 8.3.2.8 BARRAY_TYPE()

```
template const BARRAY_TYPE ( ) &
```

Definition at line 20 of file barray-meat-operators.hpp.

#### 8.3.2.9 for()

```
for ( ) = el[POS(i, j)]  [pure virtual]
```

Definition at line 66 of file barray-meat-operators.hpp.

#### 8.3.2.10 operator()()

```
j< ncol(); ++j) this-> operator() (
             i ,
             j  )
```

### 8.3.3 Variable Documentation

#### 8.3.3.1 rhs

```
Data_Type & rhs
```

**Initial value:**

```
{
    checkdim_(*this, rhs)
```

Definition at line 33 of file barray-meat-operators.hpp.

#### 8.3.3.2 this

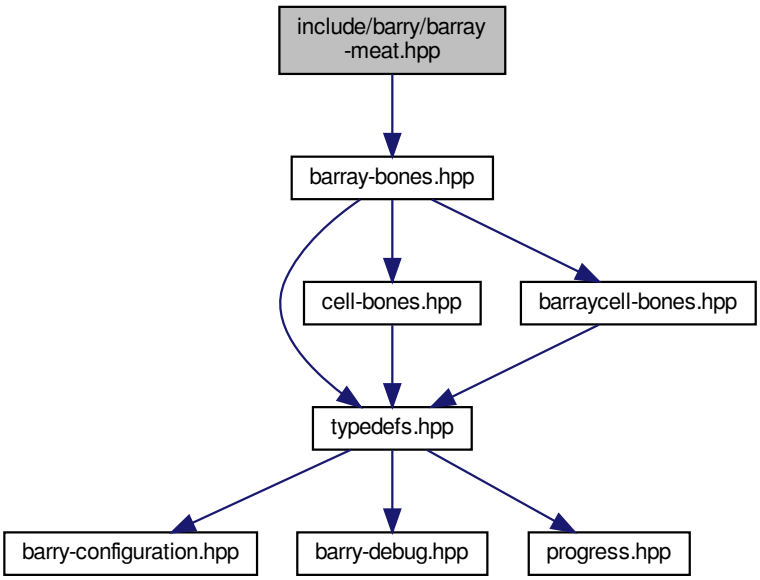
```
return * this
```

Definition at line 43 of file barray-meat-operators.hpp.

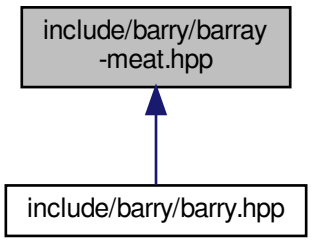
## 8.4 include/barry/barray-meat.hpp File Reference

```
#include "barray-bones.hpp"
```

Include dependency graph for barray-meat.hpp:




This graph shows which files directly or indirectly include this file:




### Macros

- #define 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

- BARRAY_TEMPLATE (, BArray)(uint N_
- el_ij resize (N)
- el_ji resize (M)
- for (uint i=0u;i< source.size();++i)
- Data_Type bool M (Array_.M)
- BARRAY_TEMPLATE (BARRAY_TYPE() &, operator=)(const BArray< Cell_Type
- BARRAY_TEMPLATE (, BArray)(BARRAY_TYPE() &&x) noexcept
- BARRAY_TEMPLATE (BARRAY_TYPE() &, operator=)(BARRAY_TYPE() &&x) noexcept
- BARRAY_TEMPLATE (bool, operator==)(const BARRAY_TYPE() &Array_)
- BARRAY_TEMPLATE (,~BArray)()
- BARRAY_TEMPLATE (void, set_data)(Data_Type *data_
- BARRAY_TEMPLATE (Data_Type *, D)()
- BARRAY_TEMPLATE (void, out_of_range)(uint i
- BARRAY_TEMPLATE (Cell_Type, get_cell)(uint i
- if (ROW(i).size()==0u) return(Cell_Type) 0.0
- if (search !=ROW(i).end()) return search -> second.value
- return (Cell_Type) 0.0
- BARRAY_TEMPLATE (std::vector< Cell_Type >, get_row_vec)(uint i
- std::vector< Cell_Type > ans (ncol(),(Cell_Type) false)
- for (const auto &iter :row(i, false)) ans[iter.first]
- BARRAY_TEMPLATE (void, get_row_vec)(std
- BARRAY_TEMPLATE (BARRAY_TYPE() &, operator-=)(const std
- BARRAY_TEMPLATE (void, insert_cell)(uint i
- if (check_exists)
- COL (j).emplace(i
- & ROW (i)[j])
- BARRAY_TEMPLATE (void, swap_cells)(uint i0
- if (report !=nullptr)(*report)
- if (check0 &check1)
- else if (!check0 &check1)
- else if (check0 &!check1)
- BARRAY_TEMPLATE (void, toggle_cell)(uint i
- BARRAY_TEMPLATE (void, swap_rows)(uint i0
- if (ROW(i0).size()==0u) move0
- if (ROW(i1).size()==0u) move1
- if (!move0 &&!move1) return
- ROW (i0).swap(ROW(i1))
- BARRAY_TEMPLATE (void, swap_cols)(uint j0
- if (COL(j0).size()==0u) check0
- if (COL(j1).size()==0u) check1
- if (check0 &&check1)
- else if (check0 &&!check1)
- else if (!check0 &&check1)
- BARRAY_TEMPLATE (void, zero_row)(uint i
- for (auto row=row0.begin();row !=row0.end();++row) rm_cell(i
- BARRAY_TEMPLATE (void, zero_col)(uint j
- if (COL(j).size()==0u) return
- BARRAY_TEMPLATE (void, transpose)()
- BARRAY_TEMPLATE (void, clear)(bool hard)
- BARRAY_TEMPLATE (void, resize)(uint N_
- if (M_< M) for(uint j = N_

## Variables

- uint M_
- uint const std::vector< uint > & source
- uint const std::vector< uint > const std::vector< uint > & target
- uint const std::vector< uint > const std::vector< uint > const std::vector< Cell_Type > & value
- uint const std::vector< uint > const std::vector< uint > const std::vector< Cell_Type > bool add
- if(source.size() !=value.size()) throw std N = N_
- M = M_
- return
- Data_Type & Array_
- Data_Type bool copy_data
- bool delete_data_
- data = data_
- delete_data = delete_data_
- uint j const
- uint j
- auto search = ROW(i).find(j)
- return ans
- uint const Cell< Cell_Type > & v
- uint const Cell< Cell_Type > bool check_bounds
- uint const Cell< Cell_Type > bool bool check_exists
- else
- NCells
- uint j0
- uint uint i1
- uint uint uint j1
- uint uint uint bool int int * report
- auto row0 = ROW(i)
- row first
- row false
- auto col0 = COL(j)

### 8.4.1 Macro Definition Documentation

#### 8.4.1.1 BARRAY_TEMPLATE

```
#define BARRAY_TEMPLATE(
              a,
              b )  template BARRAY_TEMPLATE_ARGS() inline a BARRAY_TYPE()::b
```

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

#### 8.4.1.2 BARRAY_TEMPLATE_ARGS

```
#define BARRAY_TEMPLATE_ARGS( ) <typename Cell_Type, typename Data_Type>
```

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

#### 8.4.1.3 BARRAY_TYPE

```
#define BARRAY_TYPE( ) BArray<Cell_Type, Data_Type>
```

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

#### 8.4.1.4 COL

```
#define COL(
              a ) this->el_ji[a]
```

Definition at line 21 of file barray-meat.hpp.

#### 8.4.1.5 ROW

```
#define ROW(
              a ) this->el_ij[a]
```

Definition at line 20 of file barray-meat.hpp.

### 8.4.2 Function Documentation

#### 8.4.2.1 ans()

```
std::vector< Cell_Type > ans (
             ncol() ,
             (Cell_Type) false )
```

#### 8.4.2.2 BARRAY_TEMPLATE() [1/23]

```
BARRAY_TEMPLATE (
             BArray  ) &&  [noexcept]
```

Definition at line 230 of file barray-meat.hpp.

#### 8.4.2.3 BARRAY_TEMPLATE() [2/23]

```
BARRAY_TEMPLATE (
            BArray  )
```

#### 8.4.2.4 BARRAY_TEMPLATE() [3/23]

```
BARRAY_TEMPLATE (
            ~ BArray )
```

Definition at line 339 of file barray-meat.hpp.

#### 8.4.2.5 BARRAY_TEMPLATE() [4/23]

```
BARRAY_TEMPLATE (
            BARRAY_TYPE() & ,
            operator-  ) const
```

Definition at line 586 of file barray-meat.hpp.

#### 8.4.2.6 BARRAY_TEMPLATE() [5/23]

```
BARRAY_TEMPLATE (
            BARRAY_TYPE() & ,
            operator  ) &&  [noexcept]
```

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

#### 8.4.2.7 BARRAY_TEMPLATE() [6/23]

```
BARRAY_TEMPLATE (
            BARRAY_TYPE() & ,
            operator  ) const
```

#### 8.4.2.8 BARRAY_TEMPLATE() [7/23]

```
BARRAY_TEMPLATE (
            bool ,
            operator  = = ) const &
```

Definition at line 321 of file barray-meat.hpp.

#### 8.4.2.9 BARRAY_TEMPLATE() [8/23]

```
BARRAY_TEMPLATE (
            Cell_Type ,
            get_cell  )
```

#### 8.4.2.10 BARRAY_TEMPLATE() [9/23]

```
BARRAY_TEMPLATE (
            Data_Type * ,
            D  )
```

Definition at line 361 of file barray-meat.hpp.

#### 8.4.2.11 BARRAY_TEMPLATE() [10/23]

```
BARRAY_TEMPLATE (
            std::vector< Cell_Type > ,
            get_row_vec  )
```

#### 8.4.2.12 BARRAY_TEMPLATE() [11/23]

```
BARRAY_TEMPLATE (
            void ,
            clear  )
```

Definition at line 1119 of file barray-meat.hpp.

#### 8.4.2.13 BARRAY_TEMPLATE() [12/23]

```
BARRAY_TEMPLATE (
            void ,
            get_row_vec  )
```

Definition at line 441 of file barray-meat.hpp.

#### 8.4.2.14 BARRAY_TEMPLATE() [13/23]

```
BARRAY_TEMPLATE (
            void ,
            insert_cell  )
```

#### 8.4.2.15 BARRAY_TEMPLATE() [14/23]

```
BARRAY_TEMPLATE (
            void ,
            out_of_range  )
```

#### 8.4.2.16 BARRAY_TEMPLATE() [15/23]

```
BARRAY_TEMPLATE (
            void ,
            resize  )
```

#### 8.4.2.17 BARRAY_TEMPLATE() [16/23]

```
BARRAY_TEMPLATE (
            void ,
            set_data  )
```

#### 8.4.2.18 BARRAY_TEMPLATE() [17/23]

```
BARRAY_TEMPLATE (
            void ,
            swap_cells  )
```

#### 8.4.2.19 BARRAY_TEMPLATE() [18/23]

```
BARRAY_TEMPLATE (
            void ,
            swap_cols  )
```

#### 8.4.2.20 BARRAY_TEMPLATE() [19/23]

```
BARRAY_TEMPLATE (
            void ,
            swap_rows  )
```

#### 8.4.2.21 BARRAY_TEMPLATE() [20/23]

```
BARRAY_TEMPLATE (
            void ,
            toggle_cell  )
```

#### 8.4.2.22 BARRAY_TEMPLATE() [21/23]

```
BARRAY_TEMPLATE (
            void ,
            transpose  )
```

Definition at line 1058 of file barray-meat.hpp.

#### 8.4.2.23 BARRAY_TEMPLATE() [22/23]

```
BARRAY_TEMPLATE (
            void ,
            zero_col  )
```

#### 8.4.2.24 BARRAY_TEMPLATE() [23/23]

```
BARRAY_TEMPLATE (
            void ,
            zero_row  )
```

#### 8.4.2.25 COL()

```
COL (
            j  )
```

#### 8.4.2.26 for() [1/3]

```
for (
            auto row = row0.begin();row !=row0.end();++row )
```

#### 8.4.2.27 for() [2/3]

```
for (
            const auto &iter  :rowi, false )
```

#### 8.4.2.28 for() [3/3]

```
for ( )
```

Definition at line 51 of file barray-meat.hpp.

#### 8.4.2.29 if() [1/17]

```
else if (
            !check0 && check1 )
```

Definition at line 997 of file barray-meat.hpp.

#### 8.4.2.30 if() [2/17]

```
else if (
            !check0 & check1 )
```

Definition at line 845 of file barray-meat.hpp.

#### 8.4.2.31 if() [3/17]

```
if (
            !move0 &&!  move1 )
```

#### 8.4.2.32 if() [4/17]

```
else if (
             check0 &!  check1 )
```

Definition at line 853 of file barray-meat.hpp.

#### 8.4.2.33 if() [5/17]

```
else if (
             check0 &&!  check1 )
```

Definition at line 988 of file barray-meat.hpp.

#### 8.4.2.34 if() [6/17]

```
if (
             check0 && check1 )
```

Definition at line 961 of file barray-meat.hpp.

#### 8.4.2.35 if() [7/17]

```
if (
             check0 & check1 )
```

Definition at line 827 of file barray-meat.hpp.

#### 8.4.2.36 if() [8/17]

```
else if (
             check_exists  = = CHECK::BOTH )
```

Definition at line 668 of file barray-meat.hpp.

#### 8.4.2.37 if() [9/17]

```
if (
             COL(j).size()  = =0u )
```

#### 8.4.2.38 if() [10/17]

```
if (
             COL(j0).size()   = =0u )
```

#### 8.4.2.39 if() [11/17]

```
if (
             COL(j1).size()   = =0u )
```

#### 8.4.2.40 if() [12/17]

```
else if ( ) = N_
```

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

#### 8.4.2.41 if() [13/17]

```
if (
             report !    = nullptr )
```

#### 8.4.2.42 if() [14/17]

```
if (
             ROW(i).size()   = =0u )
```

#### 8.4.2.43 if() [15/17]

```
if (
             ROW(i0).size()   = =0u )
```

#### 8.4.2.44 if() [16/17]

```
if (
             ROW(i1).size()   = =0u )
```

#### 8.4.2.45 if() [17/17]

```
if (
             search !     = ROW(i).end() ) -> second.value
```

#### 8.4.2.46 M()

```
Data_Type bool M (
             Array_.  M )
```

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

#### 8.4.2.47 resize() [1/2]

```
el_ji resize (
             M  )
```

#### 8.4.2.48 resize() [2/2]

```
el_ij resize (
             N  )
```

#### 8.4.2.49 return()

```
return (
             Cell_Type  )
```

#### 8.4.2.50 ROW() [1/2]

```
& ROW (
             i  )
```

#### 8.4.2.51 ROW() [2/2]

```
ROW (
             i0  )
```

### 8.4.3 Variable Documentation

#### 8.4.3.1 add

```
uint const std::vector< uint > const std::vector< uint > bool add
```

**Initial value:**

```
{
    if (source.size() != target.size())
        throw std::length_error("-source- and -target- don't match on length.")
```

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

#### 8.4.3.2 ans

```
return ans
```

Definition at line 438 of file barray-meat.hpp.

#### 8.4.3.3 Array_

```
Data_Type & Array_
```

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

#### 8.4.3.4 check_bounds

```
bool check_bounds
```

**Initial value:**

```
{
    if (check_bounds) {
        out_of_range(i0,0u);
        out_of_range(i1,0u);
    }

    bool move0=true, move1=true
```

Definition at line 661 of file barray-meat.hpp.

#### 8.4.3.5 check_exists

```
uint bool int check_exists
```

**Initial value:**

```
{
    if (check_bounds)
        out_of_range(i,j)
```

Definition at line 662 of file barray-meat.hpp.

#### 8.4.3.6 col0

```
auto col0 = COL(j)
```

Definition at line 1050 of file barray-meat.hpp.

#### 8.4.3.7 const

```
uint bool check_bounds const
```

**Initial value:**

```
{
    if (i >= N)
        throw std::range_error("The row is out of range.")
```

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

#### 8.4.3.8 copy_data

```
Data_Type bool copy_data
```

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

#### 8.4.3.9 data

```
data = data_
```

Definition at line 354 of file barray-meat.hpp.

#### 8.4.3.10 delete_data

```
delete_data = delete_data_
```

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

#### 8.4.3.11 delete_data_

```
bool delete_data_
```

**Initial value:**

```
{
    if ((data != nullptr) && delete_data)
        delete data
```

Definition at line 348 of file barray-meat.hpp.

#### 8.4.3.12 else

```
else (
             void  )
```

**Initial value:**

```
{
        ROW(i).insert(std::pair< uint, Cell<Cell_Type>>(j, v))
```

Definition at line 692 of file barray-meat.hpp.

#### 8.4.3.13 false

```
row false
```

Definition at line 1031 of file barray-meat.hpp.

#### 8.4.3.14 first

```
row first
```

Definition at line 1031 of file barray-meat.hpp.

#### 8.4.3.15 i1

```
uint i1
```

Definition at line 765 of file barray-meat.hpp.

#### 8.4.3.16 j

```
uint j
```

**Initial value:**

```
{
    if (init_fun == nullptr)
        return 0.0
```

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

#### 8.4.3.17 j0

```
uint j0
```

Definition at line 764 of file barray-meat.hpp.

#### 8.4.3.18 j1

```
uint j1
```

Definition at line 765 of file barray-meat.hpp.

#### 8.4.3.19 M

```
M = M_
```

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

#### 8.4.3.20 M_

```
uint M_
```

**Initial value:**

```
{

    if (N_ < N)
        for (uint i = N_; i < N; ++i)
            zero_row(i, false)
```

Definition at line 30 of file barray-meat.hpp.

#### 8.4.3.21 N

```
if (source.size() != target.size()) throw std if (source.size() != value.size()) throw std N =
N_
```

Definition at line 43 of file barray-meat.hpp.

#### 8.4.3.22 NCells

```
NCells
```

Definition at line 696 of file barray-meat.hpp.

#### 8.4.3.23 report

```
uint uint uint bool int int* report
```

Definition at line 768 of file barray-meat.hpp.

#### 8.4.3.24 return

```
return
```

Definition at line 66 of file barray-meat.hpp.

#### 8.4.3.25 row0

```
auto row0 = ROW(i)
```

Definition at line 1029 of file barray-meat.hpp.

#### 8.4.3.26 search

```
auto search = ROW(i).find(j)
```

Definition at line 415 of file barray-meat.hpp.

#### 8.4.3.27 source

```
uint const std::vector< uint > & source
```

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

#### 8.4.3.28 target

```
uint const std::vector< uint > const std::vector< uint > & target
```

Definition at line 32 of file barray-meat.hpp.

#### 8.4.3.29 v

```
uint Cell_Type v
```

Definition at line 660 of file barray-meat.hpp.

#### 8.4.3.30 value

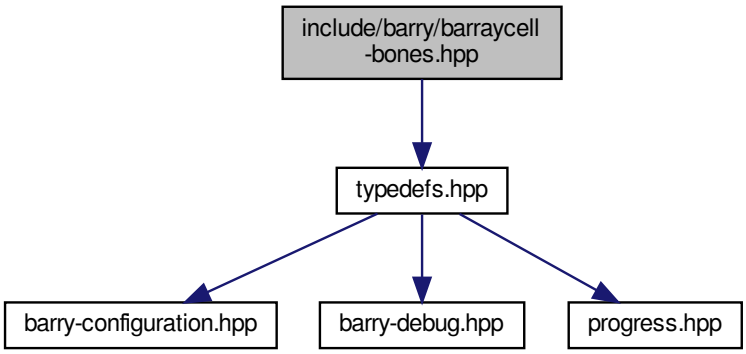
```
uint const std::vector< uint > const std::vector< uint > const std::vector< Cell_Type >&
value
```

Definition at line 33 of file barray-meat.hpp.

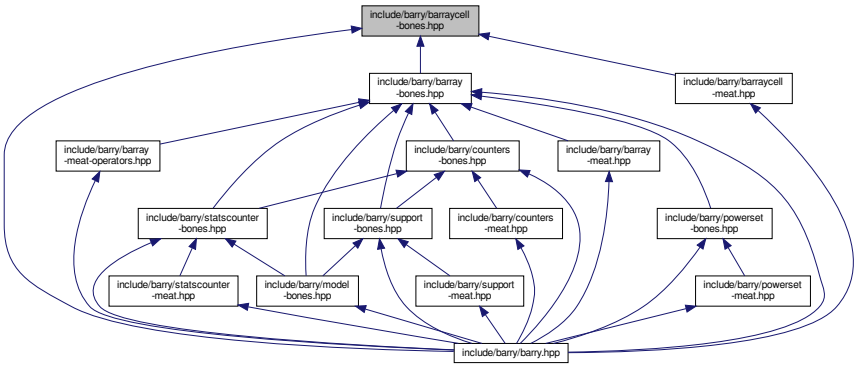
## 8.5 include/barry/barraycell-bones.hpp File Reference

```
#include "typedefs.hpp"
```

Include dependency graph for barraycell-bones.hpp:




This graph shows which files directly or indirectly include this file:



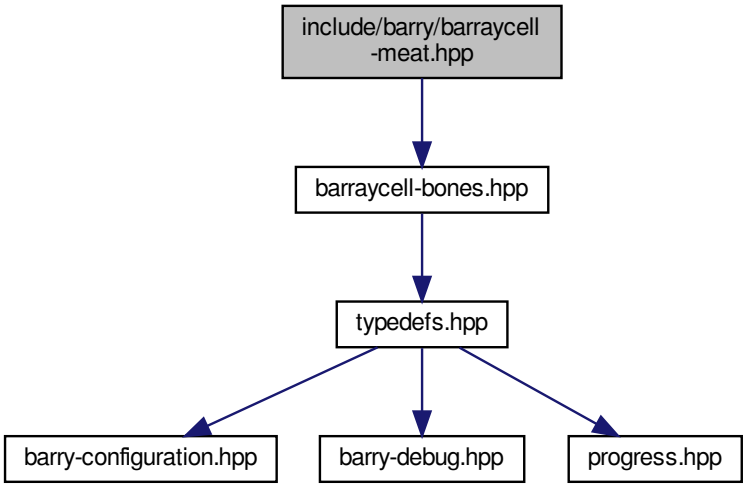

### Classes

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

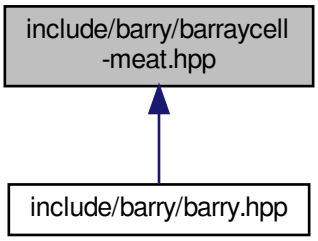
## 8.6 include/barry/barraycell-meat.hpp File Reference

```
#include "barraycell-bones.hpp"
```

Include dependency graph for barraycell-meat.hpp:



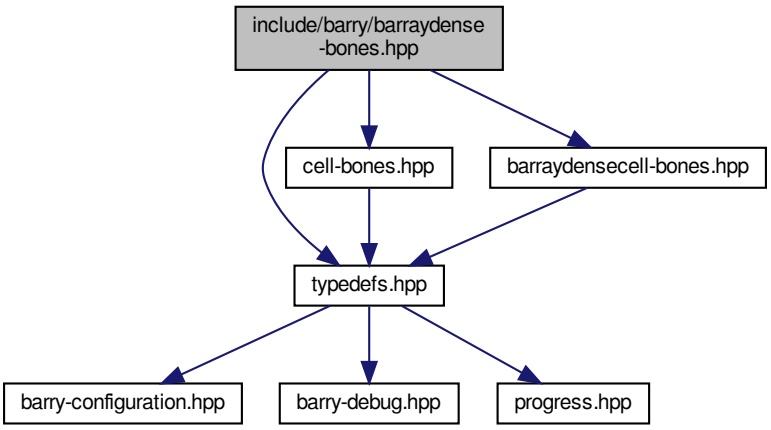

This graph shows which files directly or indirectly include this file:



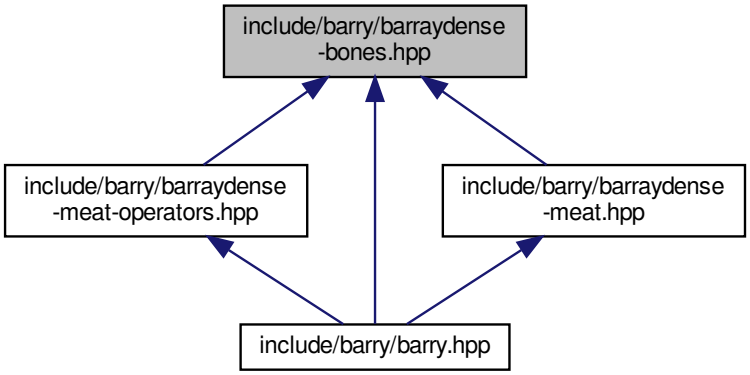

## 8.7 include/barry/barraydense-bones.hpp File Reference

```
#include "typedefs.hpp"
#include "cell-bones.hpp"
#include "barraydensecell-bones.hpp"
```

Include dependency graph for barraydense-bones.hpp:




This graph shows which files directly or indirectly include this file:




## Classes

- class BArrayDense< Cell_Type, Data_Type >

  *Baseline class for binary arrays.*

## Macros

- #define BARRY_BARRAYDENSE_BONES_HPP 1

### 8.7.1 Macro Definition Documentation

#### 8.7.1.1 BARRY_BARRAYDENSE_BONES_HPP

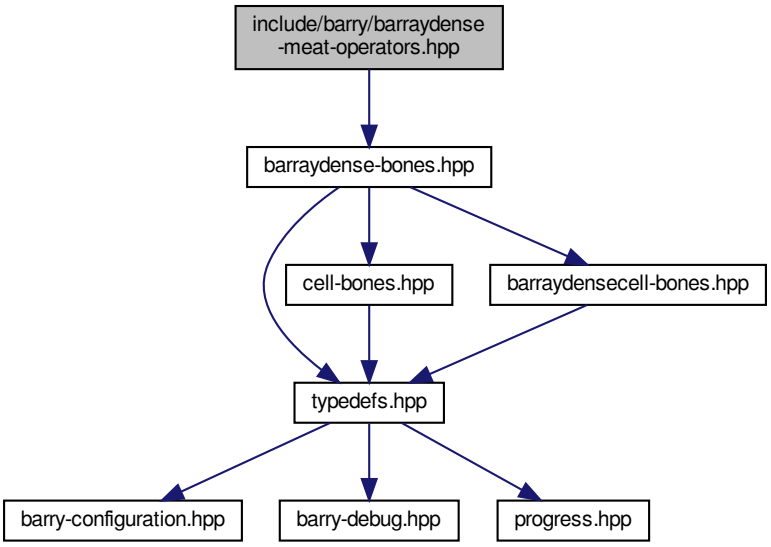
```
#define BARRY_BARRAYDENSE_BONES_HPP 1
```

Definition at line 8 of file barraydense-bones.hpp.

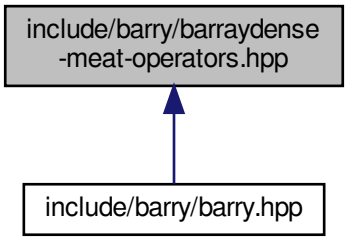
## 8.8 include/barry/barraydense-meat-operators.hpp File Reference

```
#include "barraydense-bones.hpp"
```

Include dependency graph for barraydense-meat-operators.hpp:




This graph shows which files directly or indirectly include this file:



## Macros

- #define BARRY_BARRAYDENSE_MEAT_OPERATORS_HPP 1
- #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)

### 8.8.1 Macro Definition Documentation

#### 8.8.1.1 BARRY_BARRAYDENSE_MEAT_OPERATORS_HPP

```
#define BARRY_BARRAYDENSE_MEAT_OPERATORS_HPP 1
```

Definition at line 5 of file barraydense-meat-operators.hpp.

#### 8.8.1.2 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.

#### 8.8.1.3 BDENSE_TEMPLATE_ARGS

```
template BDENSE_TEMPLATE_ARGS( ) <typename Cell_Type, typename Data_Type>
```

Definition at line 9 of file barraydense-meat-operators.hpp.

#### 8.8.1.4 BDENSE_TYPE

```
template Data_Type BDENSE_TYPE( ) BArrayDense<Cell_Type, Data_Type>
```

Definition at line 7 of file barraydense-meat-operators.hpp.

#### 8.8.1.5 COL

```
#define COL(
              a ) this->el_ji[a]
```

Definition at line 15 of file barraydense-meat-operators.hpp.

#### 8.8.1.6 POS

```
#define POS(
              a,
              b ) (b)*N + (a)
```

Definition at line 16 of file barraydense-meat-operators.hpp.

#### 8.8.1.7 POS_N

```
#define POS_N(
              a,
              b,
              c ) (b)*(c) + (a)
```

Definition at line 17 of file barraydense-meat-operators.hpp.

#### 8.8.1.8 ROW

```
#define ROW(
              a ) this->el_ij[a]
```

Definition at line 14 of file barraydense-meat-operators.hpp.

### 8.8.2 Function Documentation

#### 8.8.2.1 BDENSE_TEMPLATE() [1/4]

```
BDENSE_TEMPLATE (
            BDENSE_TYPE()& ,
            operator*  ) const &
```

Definition at line 90 of file barraydense-meat-operators.hpp.

#### 8.8.2.2 BDENSE_TEMPLATE() [2/4]

```
BDENSE_TEMPLATE (
            BDENSE_TYPE()& ,
            operator+  ) const &
```

Definition at line 34 of file barraydense-meat-operators.hpp.

#### 8.8.2.3 BDENSE_TEMPLATE() [3/4]

```
BDENSE_TEMPLATE (
            BDENSE_TYPE()& ,
            operator-  ) const &
```

Definition at line 61 of file barraydense-meat-operators.hpp.

#### 8.8.2.4 BDENSE_TEMPLATE() [4/4]

```
BDENSE_TEMPLATE (
            BDENSE_TYPE()& ,
            operator/  ) const &
```

Definition at line 101 of file barraydense-meat-operators.hpp.

#### 8.8.2.5 BDENSE_TEMPLATE_ARGS()

```
template BDENSE_TEMPLATE_ARGS ( ) const &
```

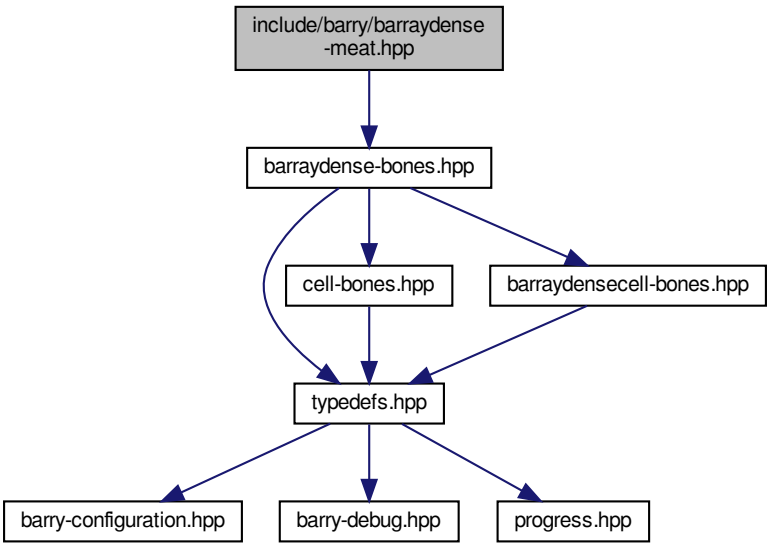
#### 8.8.2.6 BDENSE_TYPE()

```
template const BDENSE_TYPE ( ) &
```

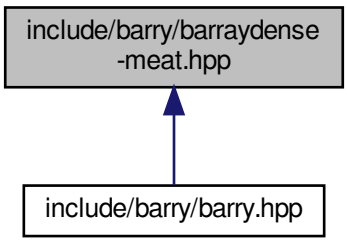
Definition at line 22 of file barraydense-meat-operators.hpp.

## 8.9 include/barry/barraydense-meat.hpp File Reference

`#include "barraydense-bones.hpp"`
Include dependency graph for barraydense-meat.hpp:




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)
- #define ZERO_CELL static_cast<Cell_Type>(0.0)

## Functions

- BDENSE_TEMPLATE (, BArrayDense)(uint N_
- el resize (N ∗M, ZERO_CELL)
- el_rowsums resize (N, ZERO_CELL)
- el_colsums resize (M, ZERO_CELL)
- for (uint i=0u;i< source.size();++i)
- BDENSE_TEMPLATE (, BArrayDense)(const BDENSE_TYPE() &Array_
- bool M (Array_.M)
- BDENSE_TEMPLATE (BDENSE_TYPE() &, operator=)(const BDENSE_TYPE() &Array_)
- BDENSE_TEMPLATE (, BArrayDense)(BDENSE_TYPE() &&x) noexcept
- BDENSE_TEMPLATE (BDENSE_TYPE() &, operator=)(BDENSE_TYPE() &&x) noexcept
- BDENSE_TEMPLATE (bool, operator==)(const BDENSE_TYPE() &Array_)
- BDENSE_TEMPLATE (, ∼BArrayDense)()
- BDENSE_TEMPLATE (void, set_data)(Data_Type ∗data_
- BDENSE_TEMPLATE (Data_Type ∗, D)()
- BDENSE_TEMPLATE (const Data_Type ∗, D)() const
- BDENSE_TEMPLATE (void, out_of_range)(uint i
- BDENSE_TEMPLATE (Cell_Type, get_cell)(uint i
- BDENSE_TEMPLATE (std::vector< Cell_Type >, get_row_vec)(uint i
- std::vector< Cell_Type > ans (ncol(), static_cast< Cell_Type >(false))
- BDENSE_TEMPLATE (void, get_row_vec)(std
- BDENSE_TEMPLATE (Entries< Cell_Type >, get_entries)() const
- BDENSE_TEMPLATE (bool, is_empty)(uint i
- BDENSE_TEMPLATE (unsigned int, nrow)() const noexcept
- BDENSE_TEMPLATE (unsigned int, ncol)() const noexcept
- BDENSE_TEMPLATE (unsigned int, nnozero)() const noexcept
- BDENSE_TEMPLATE (Cell< Cell_Type >, default_val)() const
- BDENSE_TEMPLATE (BDENSE_TYPE() &, operator+=)(const std
- BDENSE_TEMPLATE (BDENSE_TYPE() &, operator-=)(const std
- BDENSE_TEMPLATE (void, insert_cell)(uint i
- if (el[POS(i, j)]==BARRY_ZERO_DENSE)
- BDENSE_TEMPLATE (void, swap_cells)(uint i0
- if ((i0==i1) &&(j0==j1)) return
- rm_cell (i0, j0, false, false)
- rm_cell (i1, j1, false, false)
- insert_cell (i0, j0, val1, false, false)
- insert_cell (i1, j1, val0, false, false)
- BDENSE_TEMPLATE (void, toggle_cell)(uint i
- else rm_cell (i, j, false, false)
- BDENSE_TEMPLATE (void, swap_rows)(uint i0
- BDENSE_TEMPLATE (void, swap_cols)(uint j0
- BDENSE_TEMPLATE (void, zero_row)(uint i
- if (el_rowsums[i]==ZERO_CELL) return
- BDENSE_TEMPLATE (void, zero_col)(uint j
- if (el_colsums[j]==ZERO_CELL) return
- BDENSE_TEMPLATE (void, transpose)()
- BDENSE_TEMPLATE (void, clear)(bool hard)
- BDENSE_TEMPLATE (void, resize)(uint N_
- el resize (N_ ∗M_, ZERO_CELL)
- el_rowsums resize (N_, ZERO_CELL)
- el_colsums resize (M_, ZERO_CELL)
- BDENSE_TEMPLATE (void, reserve)()
- BDENSE_TEMPLATE (void, print)(const char ∗fmt
- va_start (args, fmt)

- vprintf (fmt, args)
- va_end (args)
- BDENSE_TEMPLATE (const std::vector< Cell_Type > &, get_data)() const
- BDENSE_TEMPLATE (const Cell_Type, rowsum)(unsigned int i) const
- BDENSE_TEMPLATE (const Cell_Type, colsum)(unsigned int j) const

## Variables

- uint M_
- uint const std::vector< uint > & source
- uint const std::vector< uint > const std::vector< uint > & target
- uint const std::vector< uint > const std::vector< uint > const std::vector< Cell_Type > & value
- uint const std::vector< uint > const std::vector< uint > const std::vector< Cell_Type > bool add
- if(source.size() !=value.size()) throw std N = N_
- M = M_
- return
- bool copy_data
- bool delete_data_
- data = data_
- delete_data = delete_data_
- uint j const
- uint j
- return el [POS(i, j)] == ZERO_CELL
- return ans
- uint const Cell< Cell_Type > & v
- uint const Cell< Cell_Type > bool check_bounds
- uint const Cell< Cell_Type > bool bool check_exists
- else
- el_rowsums [i] = (v.value - old)
- el_colsums [j] = (v.value - old)
- uint j0
- uint uint i1
- uint uint uint j1
- uint uint uint bool int int * report
- Cell_Type val0 = el[POS(i0,j0)]
- Cell_Type val1 = el[POS(i1,j1)]
- false
- col

### 8.9.1 Macro Definition Documentation

#### 8.9.1.1 BDENSE_TEMPLATE

```
#define BDENSE_TEMPLATE(
              a,
              b )  template BDENSE_TEMPLATE_ARGS() inline a BDENSE_TYPE()::b
```

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

#### 8.9.1.2 BDENSE_TEMPLATE_ARGS

```
#define BDENSE_TEMPLATE_ARGS( ) <typename Cell_Type, typename Data_Type>
```

Definition at line 25 of file barraydense-meat.hpp.

#### 8.9.1.3 BDENSE_TYPE

```
#define BDENSE_TYPE( ) BArrayDense<Cell_Type, Data_Type>
```

Definition at line 23 of file barraydense-meat.hpp.

#### 8.9.1.4 COL

```
#define COL(
              a ) this->el_ji[a]
```

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

#### 8.9.1.5 POS

```
#define POS(
              a,
              b ) (b)*N + (a)
```

Definition at line 32 of file barraydense-meat.hpp.

#### 8.9.1.6 POS_N

```
#define POS_N(
              a,
              b,
              c ) (b)*(c) + (a)
```

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

#### 8.9.1.7 ROW

```
#define ROW(
              a ) this->el_ij[a]
```

Definition at line 30 of file barraydense-meat.hpp.

#### 8.9.1.8 ZERO_CELL

```
#define ZERO_CELL static_cast<Cell_Type>(0.0)
```

Definition at line 38 of file barraydense-meat.hpp.

### 8.9.2 Function Documentation

#### 8.9.2.1 ans()

```
std::vector< Cell_Type > ans (
             ncol() ,
             static_cast< Cell_Type > false )
```

#### 8.9.2.2 BDENSE_TEMPLATE() [1/37]

```
BDENSE_TEMPLATE (
             BArrayDense  ) &&  [noexcept]
```

Definition at line 240 of file barraydense-meat.hpp.

#### 8.9.2.3 BDENSE_TEMPLATE() [2/37]

```
BDENSE_TEMPLATE (
             BArrayDense  ) const &
```

#### 8.9.2.4 BDENSE_TEMPLATE() [3/37]

```
BDENSE_TEMPLATE (
             BArrayDense  )
```

#### 8.9.2.5 BDENSE_TEMPLATE() [4/37]

```
BDENSE_TEMPLATE (
            ~ BArrayDense )
```

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

#### 8.9.2.6 BDENSE_TEMPLATE() [5/37]

```
BDENSE_TEMPLATE (
            BDENSE_TYPE() & ,
            operator+  ) const
```

Definition at line 558 of file barraydense-meat.hpp.

#### 8.9.2.7 BDENSE_TEMPLATE() [6/37]

```
BDENSE_TEMPLATE (
            BDENSE_TYPE() & ,
            operator-  ) const
```

Definition at line 576 of file barraydense-meat.hpp.

#### 8.9.2.8 BDENSE_TEMPLATE() [7/37]

```
BDENSE_TEMPLATE (
            BDENSE_TYPE() & ,
            operator  ) &&  [noexcept]
```

Definition at line 257 of file barraydense-meat.hpp.

#### 8.9.2.9 BDENSE_TEMPLATE() [8/37]

```
BDENSE_TEMPLATE (
            BDENSE_TYPE() & ,
            operator  ) const &
```

Definition at line 194 of file barraydense-meat.hpp.

#### 8.9.2.10 BDENSE_TEMPLATE() [9/37]

```
BDENSE_TEMPLATE (
            bool ,
            is_empty  )
```

#### 8.9.2.11 BDENSE_TEMPLATE() [10/37]

```
BDENSE_TEMPLATE (
            bool ,
            operator  = = ) const &
```

Definition at line 300 of file barraydense-meat.hpp.

#### 8.9.2.12 BDENSE_TEMPLATE() [11/37]

```
BDENSE_TEMPLATE (
            Cell< Cell_Type > ,
            default_val  ) const
```

Definition at line 554 of file barraydense-meat.hpp.

#### 8.9.2.13 BDENSE_TEMPLATE() [12/37]

```
BDENSE_TEMPLATE (
            Cell_Type ,
            get_cell  )
```

#### 8.9.2.14 BDENSE_TEMPLATE() [13/37]

```
BDENSE_TEMPLATE (
            const Cell_Type,
            colsum  ) const
```

Definition at line 991 of file barraydense-meat.hpp.

#### 8.9.2.15 BDENSE_TEMPLATE() [14/37]

```
BDENSE_TEMPLATE (
            const Cell_Type,
            rowsum  ) const
```

Definition at line 986 of file barraydense-meat.hpp.

#### 8.9.2.16 BDENSE_TEMPLATE() [15/37]

```
BDENSE_TEMPLATE (
            const Data_Type * ,
            D  ) const
```

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

#### 8.9.2.17 BDENSE_TEMPLATE() [16/37]

```
BDENSE_TEMPLATE (
            const std::vector< Cell_Type > & ,
            get_data  ) const
```

Definition at line 981 of file barraydense-meat.hpp.

#### 8.9.2.18 BDENSE_TEMPLATE() [17/37]

```
BDENSE_TEMPLATE (
            Data_Type * ,
            D  )
```

Definition at line 341 of file barraydense-meat.hpp.

#### 8.9.2.19 BDENSE_TEMPLATE() [18/37]

```
BDENSE_TEMPLATE (
            Entries< Cell_Type > ,
            get_entries  ) const
```

Definition at line 494 of file barraydense-meat.hpp.

#### 8.9.2.20 BDENSE_TEMPLATE() [19/37]

```
BDENSE_TEMPLATE (
            std::vector< Cell_Type > ,
            get_row_vec  )
```

#### 8.9.2.21 BDENSE_TEMPLATE() [20/37]

```
BDENSE_TEMPLATE (
            unsigned int ,
            ncol  ) const  [noexcept]
```

Definition at line 540 of file barraydense-meat.hpp.

#### 8.9.2.22 BDENSE_TEMPLATE() [21/37]

```
BDENSE_TEMPLATE (
            unsigned int ,
            nnozero  ) const  [noexcept]
```

Definition at line 544 of file barraydense-meat.hpp.

#### 8.9.2.23 BDENSE_TEMPLATE() [22/37]

```
BDENSE_TEMPLATE (
            unsigned int ,
            nrow  ) const  [noexcept]
```

Definition at line 536 of file barraydense-meat.hpp.

#### 8.9.2.24 BDENSE_TEMPLATE() [23/37]

```
BDENSE_TEMPLATE (
            void ,
            clear  )
```

Definition at line 888 of file barraydense-meat.hpp.

#### 8.9.2.25 BDENSE_TEMPLATE() [24/37]

```
BDENSE_TEMPLATE (
            void ,
            get_row_vec  )
```

Definition at line 394 of file barraydense-meat.hpp.

#### 8.9.2.26 BDENSE_TEMPLATE() [25/37]

```
BDENSE_TEMPLATE (
            void ,
            insert_cell  )
```

#### 8.9.2.27 BDENSE_TEMPLATE() [26/37]

```
BDENSE_TEMPLATE (
            void ,
            out_of_range  )
```

#### 8.9.2.28 BDENSE_TEMPLATE() [27/37]

```
BDENSE_TEMPLATE (
            void ,
            print  ) const
```

#### 8.9.2.29 BDENSE_TEMPLATE() [28/37]

```
BDENSE_TEMPLATE (
            void ,
            reserve  )
```

Definition at line 938 of file barraydense-meat.hpp.

#### 8.9.2.30 BDENSE_TEMPLATE() [29/37]

```
BDENSE_TEMPLATE (
            void ,
            resize  )
```

#### 8.9.2.31 BDENSE_TEMPLATE() [30/37]

```
BDENSE_TEMPLATE (
            void ,
            set_data  )
```

#### 8.9.2.32 BDENSE_TEMPLATE() [31/37]

```
BDENSE_TEMPLATE (
            void ,
            swap_cells  )
```

#### 8.9.2.33 BDENSE_TEMPLATE() [32/37]

```
BDENSE_TEMPLATE (
            void ,
            swap_cols  )
```

#### 8.9.2.34 BDENSE_TEMPLATE() [33/37]

```
BDENSE_TEMPLATE (
            void ,
            swap_rows  )
```

#### 8.9.2.35 BDENSE_TEMPLATE() [34/37]

```
BDENSE_TEMPLATE (
            void ,
            toggle_cell  )
```

#### 8.9.2.36 BDENSE_TEMPLATE() [35/37]

```
BDENSE_TEMPLATE (
            void ,
            transpose  )
```

Definition at line 860 of file barraydense-meat.hpp.

#### 8.9.2.37 BDENSE_TEMPLATE() [36/37]

```
BDENSE_TEMPLATE (
             void ,
             zero_col  )
```

#### 8.9.2.38 BDENSE_TEMPLATE() [37/37]

```
BDENSE_TEMPLATE (
             void ,
             zero_row  )
```

#### 8.9.2.39 for()

```
for ( )
```

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

#### 8.9.2.40 if() [1/4]

```
if (
             (i0==i1) &&(j0==j1)  )
```

#### 8.9.2.41 if() [2/4]

```
if (
             el [POS(i, j)] = = BARRY_ZERO_DENSE )
```

Definition at line 655 of file barraydense-meat.hpp.

#### 8.9.2.42 if() [3/4]

```
if (
             el_colsums [j] = =ZERO_CELL )
```

#### 8.9.2.43 if() [4/4]

```
if (
             el_rowsums [i] = =ZERO_CELL )
```

#### 8.9.2.44 insert_cell() [1/2]

```
insert_cell (
             i0 ,
             j0 ,
             val1 ,
             false ,
             false  )
```

#### 8.9.2.45 insert_cell() [2/2]

```
insert_cell (
             i1 ,
             j1 ,
             val0 ,
             false ,
             false  )
```

#### 8.9.2.46 M()

```
bool M (
             Array_.  M )
```

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

#### 8.9.2.47 resize() [1/6]

```
el_colsums resize (
             M ,
             ZERO_CELL  )
```

#### 8.9.2.48 resize() [2/6]

```
el_colsums resize (
            M_ ,
            ZERO_CELL  )
```

#### 8.9.2.49 resize() [3/6]

```
el resize (
            N * M,
            ZERO_CELL  )
```

#### 8.9.2.50 resize() [4/6]

```
el_rowsums resize (
            N ,
            ZERO_CELL  )
```

#### 8.9.2.51 resize() [5/6]

```
el resize (
            N_ * M_,
            ZERO_CELL  )
```

#### 8.9.2.52 resize() [6/6]

```
el_rowsums resize (
            N_ ,
            ZERO_CELL  )
```

#### 8.9.2.53 rm_cell() [1/3]

```
else rm_cell (
            i ,
            j ,
            false ,
            false  )
```

#### 8.9.2.54 rm_cell() [2/3]

```
rm_cell (
            i0 ,
            j0 ,
            false ,
            false  )
```

#### 8.9.2.55 rm_cell() [3/3]

```
rm_cell (
            i1 ,
            j1 ,
            false ,
            false  )
```

#### 8.9.2.56 va_end()

```
va_end (
            args  )
```

#### 8.9.2.57 va_start()

```
va_start (
            args ,
            fmt  )
```

#### 8.9.2.58 vprintf()

```
vprintf (
            fmt ,
            args  )
```

### 8.9.3 Variable Documentation

#### 8.9.3.1 add

```
uint const std::vector< uint > const std::vector< uint > bool add
```

**Initial value:**

```
{
    if (source.size() != target.size())
        throw std::length_error("-source- and -target- don't match on length.")
```

Definition at line 47 of file barraydense-meat.hpp.

#### 8.9.3.2 ans

```
return ans
```

Definition at line 390 of file barraydense-meat.hpp.

#### 8.9.3.3 check_bounds

```
bool check_bounds
```

**Initial value:**

```
{
    if (check_bounds)
    {
        out_of_range(i0,0u);
        out_of_range(i1,0u);
    }



    for (uint j = 0u; j < M; ++j)
        std::swap(el[POS(i0, j)], el[POS(i1, j)])
```

Definition at line 646 of file barraydense-meat.hpp.

#### 8.9.3.4 check_exists

```
uint bool int check_exists
```

**Initial value:**

```
{
    if (check_bounds)
        out_of_range(i,j)
```

Definition at line 647 of file barraydense-meat.hpp.

#### 8.9.3.5 col

```
col
```

Definition at line 835 of file barraydense-meat.hpp.

#### 8.9.3.6 const

```
const
```

**Initial value:**

```
{
    if (i >= N)
        throw std::range_error("The row is out of range.")
```

Definition at line 352 of file barraydense-meat.hpp.

#### 8.9.3.7 copy_data

```
bool copy_data
```

Definition at line 156 of file barraydense-meat.hpp.

#### 8.9.3.8 data

```
data = data_
```

Definition at line 334 of file barraydense-meat.hpp.

#### 8.9.3.9 delete_data

```
delete_data = delete_data_
```

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

#### 8.9.3.10 delete_data_

```
bool delete_data_
```

**Initial value:**

```
{
    if ((data != nullptr) && delete_data)
        delete data
```

Definition at line 328 of file barraydense-meat.hpp.

#### 8.9.3.11 el

```
return el == ZERO_CELL
```

Definition at line 373 of file barraydense-meat.hpp.

#### 8.9.3.12 el_colsums

```
el_colsums[j] = (v.value - old)
```

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

#### 8.9.3.13 el_rowsums

```
el_rowsums[i] = (v.value - old)
```

Definition at line 666 of file barraydense-meat.hpp.

#### 8.9.3.14 else

```
else (
             void  )
```

**Initial value:**

```
{
        Cell_Type old = el[POS(i,j)]
```

Definition at line 662 of file barraydense-meat.hpp.

#### 8.9.3.15 false

```
false
```

Definition at line 759 of file barraydense-meat.hpp.

#### 8.9.3.16 i1

```
uint i1
```

Definition at line 713 of file barraydense-meat.hpp.

#### 8.9.3.17 j

```
j
```

Definition at line 365 of file barraydense-meat.hpp.

#### 8.9.3.18 j0

```
uint j0
```

Definition at line 712 of file barraydense-meat.hpp.

#### 8.9.3.19 j1

```
uint j1
```

Definition at line 713 of file barraydense-meat.hpp.

#### 8.9.3.20 M

```
M = M_
```

Definition at line 57 of file barraydense-meat.hpp.

#### 8.9.3.21 M_

```
uint M_
```

**Initial value:**

```
{
    std::vector< Cell_Type > el_tmp(el)
```

Definition at line 43 of file barraydense-meat.hpp.

#### 8.9.3.22 N

```
N = N_
```

Definition at line 56 of file barraydense-meat.hpp.

#### 8.9.3.23 report

```
uint uint uint bool int int* report
```

**Initial value:**

```
{
    if (check_bounds) {
        out_of_range(i0,j0);
        out_of_range(i1,j1);
    }


    if (report != nullptr)
        (*report) = EXISTS::BOTH
```

Definition at line 716 of file barraydense-meat.hpp.

#### 8.9.3.24 return

```
return
```

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

#### 8.9.3.25 source

```
uint const std::vector< uint >& source
```

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

#### 8.9.3.26 target

```
uint const std::vector< uint > const std::vector< uint >& target
```

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

#### 8.9.3.27 v

```
uint Cell_Type v
```

Definition at line 645 of file barraydense-meat.hpp.

#### 8.9.3.28 val0

```
Cell_Type val0 = el[POS(i0,j0)]
```

Definition at line 734 of file barraydense-meat.hpp.

#### 8.9.3.29 val1

```
Cell_Type val1 = el[POS(i1,j1)]
```

Definition at line 735 of file barraydense-meat.hpp.

#### 8.9.3.30 value

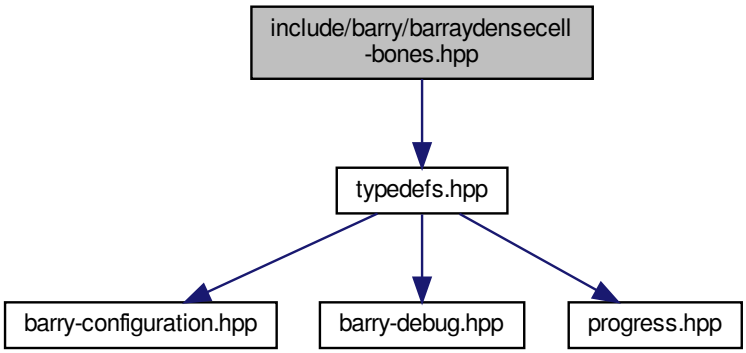
```
uint const std::vector< uint > const std::vector< uint > const std::vector< Cell_Type >&
value
```

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

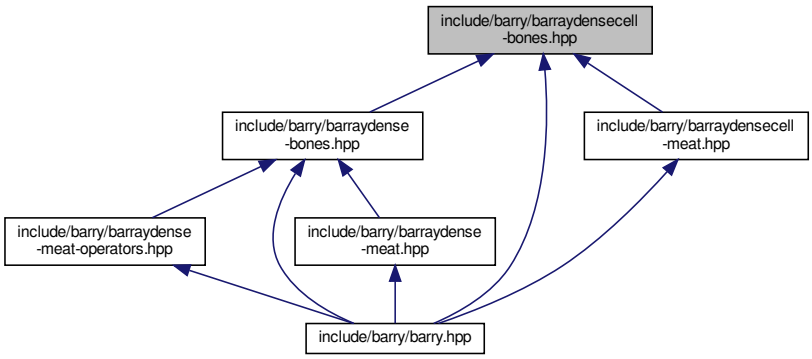
## 8.10 include/barry/barraydensecell-bones.hpp File Reference

```
#include "typedefs.hpp"
```

Include dependency graph for barraydensecell-bones.hpp:




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

### 8.10.1 Macro Definition Documentation

#### 8.10.1.1 POS

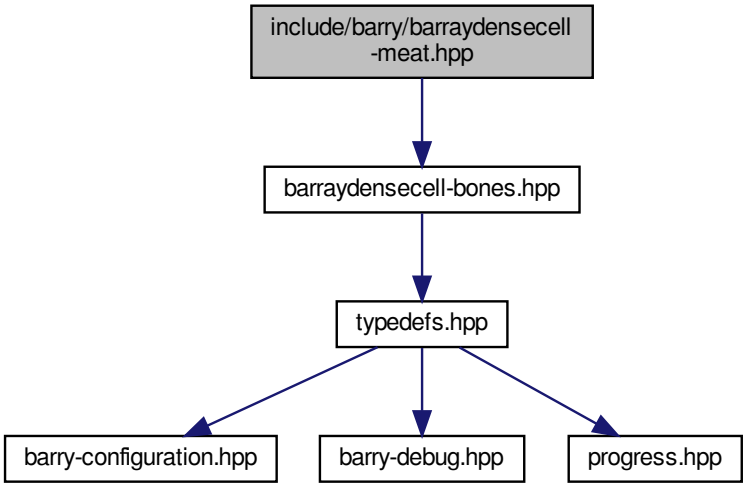
```
#define POS(
              a,
              b ) (a) + (b) * N
```

Definition at line 6 of file barraydensecell-bones.hpp.

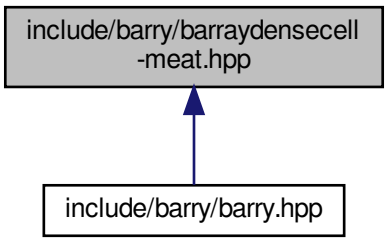
## 8.11 include/barry/barraydensecell-meat.hpp File Reference

```
#include "barraydensecell-bones.hpp"
```

Include dependency graph for barraydensecell-meat.hpp:




This graph shows which files directly or indirectly include this file:



### Macros

- #define POS(a, b) (a) + (b) ∗ dat->N

### 8.11.1 Macro Definition Documentation

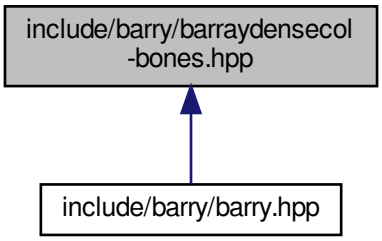
#### 8.11.1.1 POS

```
#define POS(
              a,
              b ) (a) + (b) * dat->N
```

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

## 8.12 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)

### 8.12.1 Macro Definition Documentation

#### 8.12.1.1 POS

```
#define POS(
              a,
              b ) (b)*N + (a)
```

Definition at line 4 of file barraydensecol-bones.hpp.

#### 8.12.1.2 POS_N

```
#define POS_N(
              a,
              b,
              c ) (b)*(c) + (a)
```

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

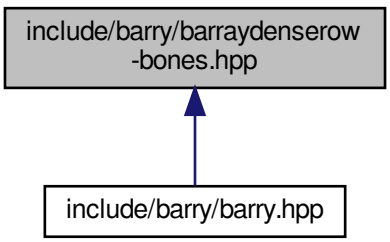
#### 8.12.1.3 ZERO_CELL

```
#define ZERO_CELL static_cast<Cell_Type>(0.0)
```

Definition at line 6 of file barraydensecol-bones.hpp.

## 8.13 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)

### 8.13.1 Macro Definition Documentation

#### 8.13.1.1 POS

```
#define POS(
              a,
              b ) (b) * N + (a)
```

Definition at line 4 of file barraydenserow-bones.hpp.

#### 8.13.1.2 POS_N

```
#define POS_N(
              a,
              b,
              c ) (b)*(c) + (a)
```

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

#### 8.13.1.3 ZERO_CELL

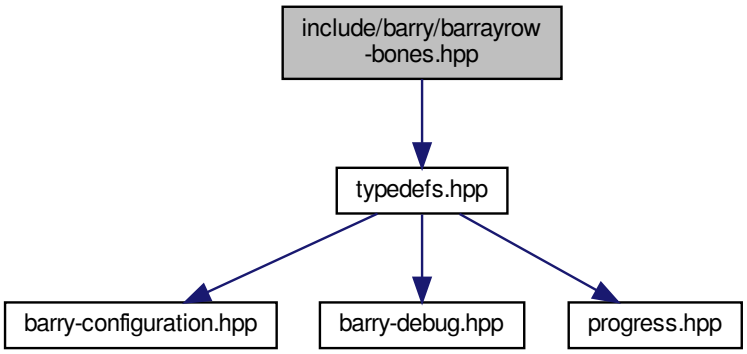
```
#define ZERO_CELL static_cast< Cell_Type >(0.0)
```

Definition at line 6 of file barraydenserow-bones.hpp.

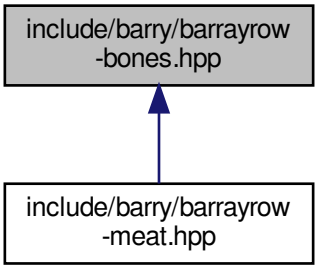
## 8.14 include/barry/barrayrow-bones.hpp File Reference

```
#include "typedefs.hpp"
```

Include dependency graph for barrayrow-bones.hpp:




This graph shows which files directly or indirectly include this file:



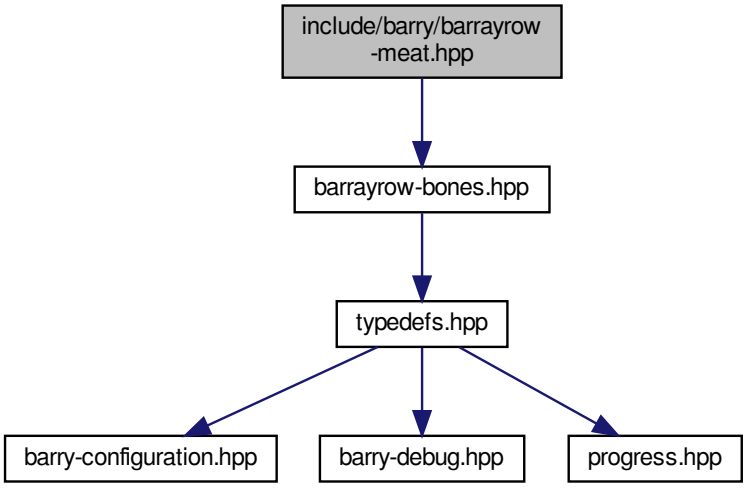

### Classes

- class BArrayRow< Cell_Type, Data_Type >
- class BArrayRow_const< Cell_Type, Data_Type >

## 8.15 include/barry/barrayrow-meat.hpp File Reference

```
#include "barrayrow-bones.hpp"
```

Include dependency graph for barrayrow-meat.hpp:




## Macros

- #define BARRY_BARRAYROW_MEAT_HPP 1
- #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)

### 8.15.1 Macro Definition Documentation

#### 8.15.1.1 BARRY_BARRAYROW_MEAT_HPP

```
#define BARRY_BARRAYROW_MEAT_HPP 1
```

Definition at line 4 of file barrayrow-meat.hpp.

#### 8.15.1.2 BROW_TEMPLATE

```
#define BROW_TEMPLATE(
              a,
              b )  template BROW_TEMPLATE_ARGS() inline a BROW_TYPE()::b
```

Definition at line 10 of file barrayrow-meat.hpp.

#### 8.15.1.3 BROW_TEMPLATE_ARGS

```
#define BROW_TEMPLATE_ARGS( ) <typename Cell_Type, typename Data_Type>
```

Definition at line 8 of file barrayrow-meat.hpp.

#### 8.15.1.4 BROW_TYPE

```
#define BROW_TYPE( ) BArrayRow<Cell_Type, Data_Type>
```

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

### 8.15.2 Function Documentation

#### 8.15.2.1 BROW_TEMPLATE() [1/5]

```
BROW_TEMPLATE (
             void ,
             operator*  ) const &
```

Definition at line 47 of file barrayrow-meat.hpp.

#### 8.15.2.2 BROW_TEMPLATE() [2/5]

```
BROW_TEMPLATE (
             void ,
             operator+  ) const &
```

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

#### 8.15.2.3 BROW_TEMPLATE() [3/5]

```
BROW_TEMPLATE (
            void ,
            operator-  ) const &
```

Definition at line 36 of file barrayrow-meat.hpp.

#### 8.15.2.4 BROW_TEMPLATE() [4/5]

```
BROW_TEMPLATE (
            void ,
            operator/  ) const &
```

Definition at line 57 of file barrayrow-meat.hpp.

#### 8.15.2.5 BROW_TEMPLATE() [5/5]

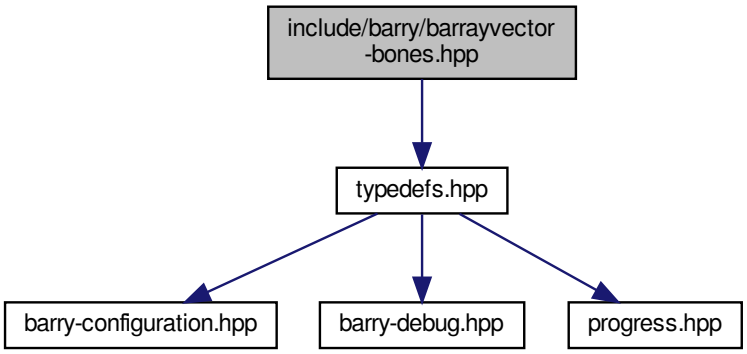
```
BROW_TEMPLATE (
            void ,
            operator  ) const &
```

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

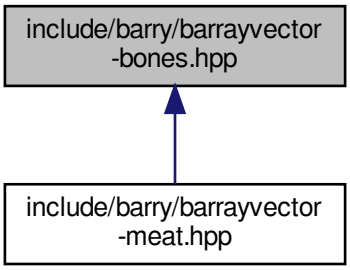
## 8.16 include/barry/barrayvector-bones.hpp File Reference

```
#include "typedefs.hpp"
```

Include dependency graph for barrayvector-bones.hpp:



This graph shows which files directly or indirectly include this file:




### Classes

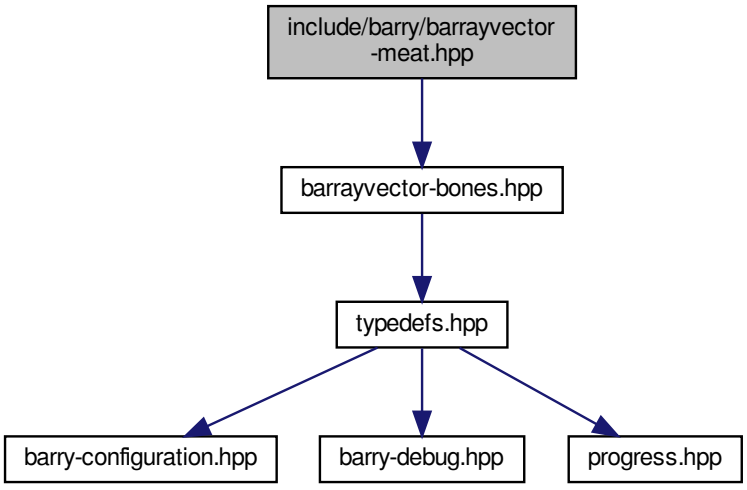
- class BArrayVector< Cell_Type, Data_Type >

  *Row or column of a `BArray`*
- class BArrayVector_const< Cell_Type, Data_Type >

## 8.17 include/barry/barrayvector-meat.hpp File Reference

```
#include "barrayvector-bones.hpp"
```

Include dependency graph for barrayvector-meat.hpp:



## Macros

- #define BARRY_BARRAYVECTOR_MEAT_HPP 1

### 8.17.1 Macro Definition Documentation

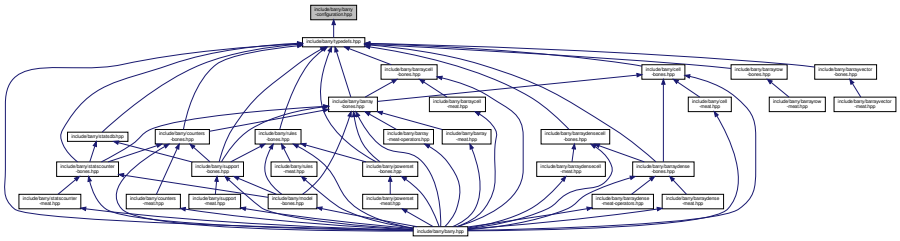
#### 8.17.1.1 BARRY_BARRAYVECTOR_MEAT_HPP

```
#define BARRY_BARRAYVECTOR_MEAT_HPP 1
```

Definition at line 4 of file barrayvector-meat.hpp.

## 8.18 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 >(UINT_MAX/2u)
- template<typename Ta , typename Tb >
  using Map = std::map< Ta, Tb >

## 8.18.1 Macro Definition Documentation

### 8.18.1.1 BARRY_CHECK_SUPPORT

```
#define BARRY_CHECK_SUPPORT(
              x,
              maxs )
```

Definition at line 47 of file barry-configuration.hpp.

### 8.18.1.2 BARRY_ISFINITE

```
#define BARRY_ISFINITE(
              a )
```

Definition at line 40 of file barry-configuration.hpp.

### 8.18.1.3 BARRY_MAX_NUM_ELEMENTS

```
#define BARRY_MAX_NUM_ELEMENTS static_cast< size_t >(UINT_MAX/2u)
```

Definition at line 55 of file barry-configuration.hpp.

### 8.18.1.4 BARRY_SAFE_EXP

```
#define BARRY_SAFE_EXP -100.0
```

Definition at line 33 of file barry-configuration.hpp.

### 8.18.1.5 printf_barry

```
#define printf_barry printf
```

Definition at line 51 of file barry-configuration.hpp.

## 8.18.2 Typedef Documentation

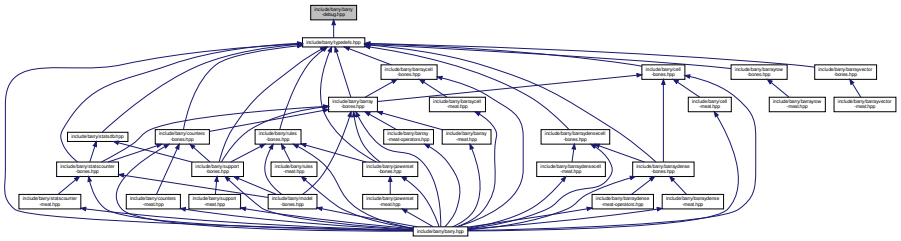
#### 8.18.2.1 Map

```
template<typename Ta , typename Tb >
using Map = std::map<Ta,Tb>
```

Definition at line 27 of file barry-configuration.hpp.

## 8.19 include/barry/barry-debug.hpp File Reference

This graph shows which files directly or indirectly include this file:



### Macros

- #define BARRY_DEBUG_LEVEL 0

### 8.19.1 Macro Definition Documentation

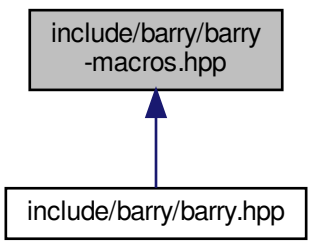
#### 8.19.1.1 BARRY_DEBUG_LEVEL

```
#define BARRY_DEBUG_LEVEL 0
```

Definition at line 5 of file barry-debug.hpp.

## 8.20 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);

## 8.20.1 Macro Definition Documentation

### 8.20.1.1 BARRY_ONE

```
#define BARRY_ONE Cell<Cell_Type>(1.0)
```

Definition at line 7 of file barry-macros.hpp.

### 8.20.1.2 BARRY_ONE_DENSE

```
#define BARRY_ONE_DENSE static_cast<Cell_Type>(1.0)
```

Definition at line 8 of file barry-macros.hpp.

### 8.20.1.3 BARRY_UNUSED

```
#define BARRY_UNUSED(
              expr ) do { (void)(expr); } while (0);
```

Definition at line 10 of file barry-macros.hpp.

### 8.20.1.4 BARRY_ZERO

```
#define BARRY_ZERO Cell<Cell_Type>(0.0)
```

Definition at line 4 of file barry-macros.hpp.

#### 8.20.1.5 BARRY_ZERO_DENSE

```
#define BARRY_ZERO_DENSE static_cast<Cell_Type>(0.0)
```

Definition at line 5 of file barry-macros.hpp.

## 8.21 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 "typedefs.hpp"
#include "barry-macros.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 "counters/phylo.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
- barry::counters::phylo

## Macros

- #define BARRY_HPP
- #define BARRY_VERSION 0.1
- #define COUNTER_FUNCTION(a)
- #define COUNTER_LAMBDA(a)
- #define RULE_FUNCTION(a)
- #define RULE_LAMBDA(a)

### 8.21.1 Macro Definition Documentation

#### 8.21.1.1 BARRY_HPP

```
#define BARRY_HPP
```

Definition at line 22 of file barry.hpp.

#### 8.21.1.2 BARRY_VERSION

```
#define BARRY_VERSION 0.1
```

Definition at line 24 of file barry.hpp.

#### 8.21.1.3 COUNTER_FUNCTION

```
#define COUNTER_FUNCTION(
                a )
```

**Value:**

```
    template <typename Array_Type = barry::BArray<>, typename Data_Type = bool> \
    inline double (a) (const Array_Type & Array, uint i, uint j, Data_Type * data)\
```

Definition at line 86 of file barry.hpp.

#### 8.21.1.4 COUNTER_LAMBDA

```
#define COUNTER_LAMBDA(
                a )
```

**Value:**

```
    template <typename Array_Type = barry::BArray<>, typename Data_Type = bool> \
    Counter_fun_type<Array_Type, Data_Type> a = \
    [](const Array_Type & Array, uint i, uint j, Data_Type * data)
```

Definition at line 89 of file barry.hpp.

#### 8.21.1.5 RULE_FUNCTION

```
#define RULE_FUNCTION(
                a )
```

**Value:**

```
    template <typename Array_Type = barry::BArray<>, typename Data_Type = bool> \
    inline bool (a) (const Array_Type & Array, uint i, uint j, Data_Type * data)\
```

Definition at line 93 of file barry.hpp.

#### 8.21.1.6 RULE_LAMBDA

```
#define RULE_LAMBDA(
                a )
```

**Value:**

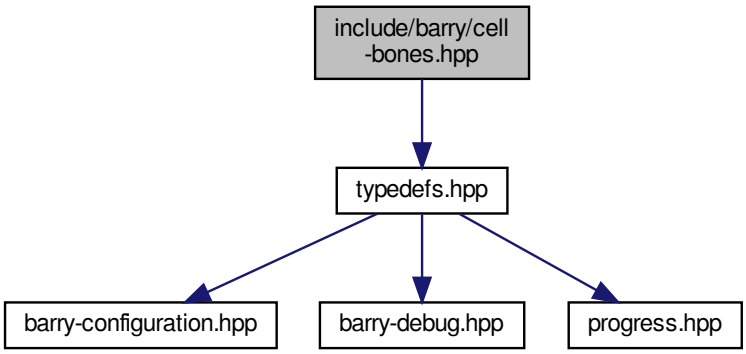
```
    template <typename Array_Type = barry::BArray<>, typename Data_Type = bool> \
    Rule_fun_type<Array_Type, Data_Type> a = \
    [](const Array_Type & Array, uint i, uint j, Data_Type * data)
```

Definition at line 96 of file barry.hpp.

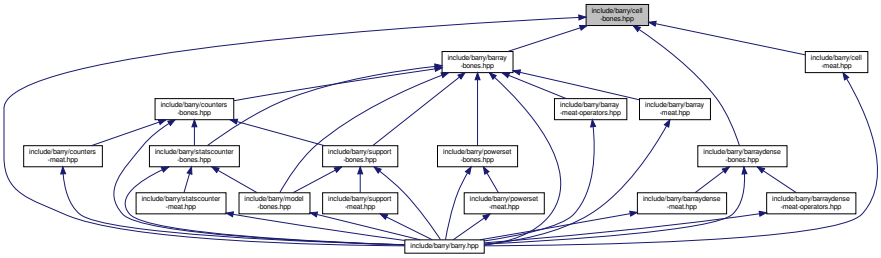
## 8.22 include/barry/cell-bones.hpp File Reference

```
#include "typedefs.hpp"
```

Include dependency graph for cell-bones.hpp:




This graph shows which files directly or indirectly include this file:



### Classes

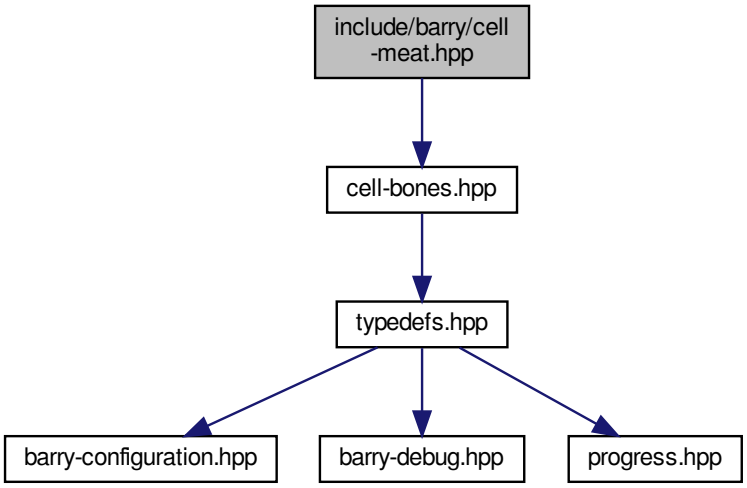
- class Cell< Cell_Type >

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

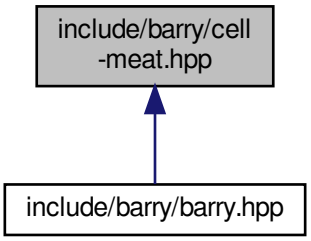
## 8.23 include/barry/cell-meat.hpp File Reference

```
#include "cell-bones.hpp"
```

Include dependency graph for cell-meat.hpp:




This graph shows which files directly or indirectly include this file:

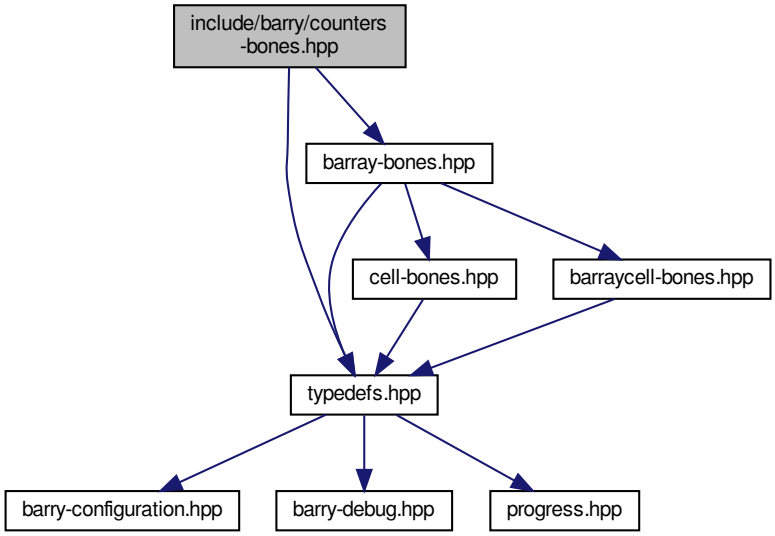



## 8.24 include/barry/col-bones.hpp File Reference

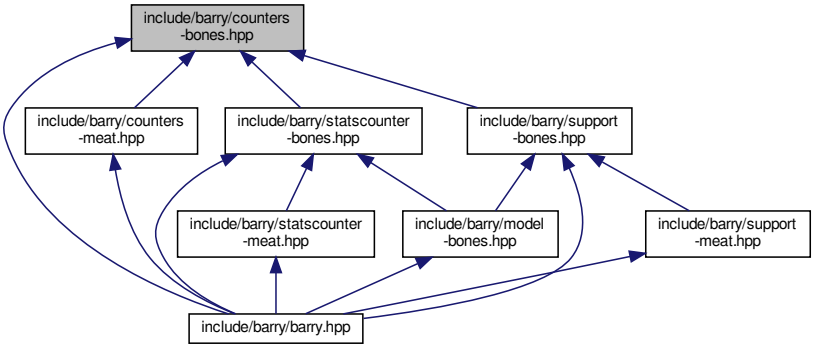
## 8.25 include/barry/counters-bones.hpp File Reference

```
#include "typedefs.hpp"
#include "barray-bones.hpp"
```

Include dependency graph for counters-bones.hpp:




This graph shows which files directly or indirectly include this file:




## Classes

- class Counter< Array_Type, Data_Type >

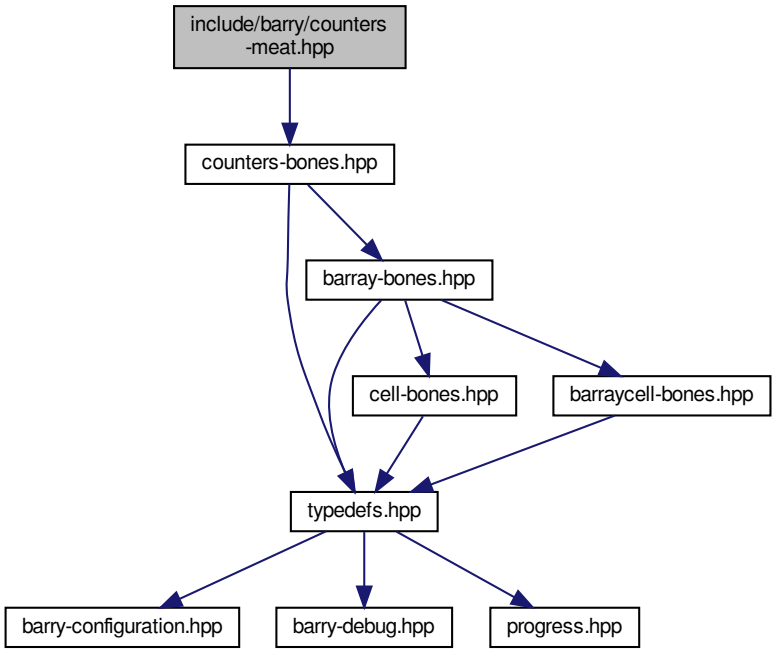
  *A counter function based on change statistics.*
- class Counters< Array_Type, Data_Type >

  *Vector of counters.*

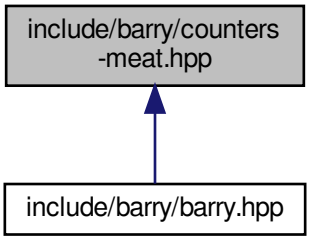
## 8.26 include/barry/counters-meat.hpp File Reference

```
#include "counters-bones.hpp"
```

Include dependency graph for counters-meat.hpp:




This graph shows which files directly or indirectly include this file:




### 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 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 &&counter_ init_fun (std::move(counter_.init_fun))
- Data_Type &&counter_ data (std::move(counter_.data))
- Data_Type &&counter_ delete_data (std::move(counter_.delete_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
- COUNTERS_TEMPLATE (, Counters)()
- COUNTERS_TEMPLATE (COUNTER_TYPE() &, operator[ ])(uint idx)
- Data_Type Data_Type to_be_deleted (new std::vector< uint >(0u))
- Data_Type Data_Type delete_data (true)
- Data_Type Data_Type delete_to_be_deleted (true)
- Data_Type &&counters_ to_be_deleted (std::move(counters_.to_be_deleted))
- Data_Type &&counters_ delete_data (std::move(counters_.delete_data))
- Data_Type &&counters_ delete_to_be_deleted (std::move(counters_.delete_to_be_deleted))
- 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
- data push_back (new Counter< Array_Type, Data_Type >(counter))
- data push_back (new Counter< Array_Type, Data_Type >(count_fun_, init_fun_, data_, delete_data_, name_, desc_))
- COUNTERS_TEMPLATE (void, clear)()
- COUNTERS_TEMPLATE (std::vector< std::string >, get_names)() const
- COUNTERS_TEMPLATE (std::vector< std::string >, get_descriptions)() const

## Variables

- Data_Type & counter_
- Data_Type &&counter_ noexcept
- uint i
- uint uint j
- 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 > Data_Type * data_
- Data_Type Counter_fun_type< Array_Type, Data_Type > Data_Type bool delete_data_
- Data_Type Counter_fun_type< Array_Type, Data_Type > Data_Type bool std::string name_
- Data_Type Counter_fun_type< Array_Type, Data_Type > Data_Type bool std::string std::string desc_

### 8.26.1 Macro Definition Documentation

#### 8.26.1.1 COUNTER_TEMPLATE

```
#define COUNTER_TEMPLATE(
              a,
              b )  template COUNTER_TEMPLATE_ARGS() inline a COUNTER_TYPE()::b
```

Definition at line 10 of file counters-meat.hpp.

#### 8.26.1.2 COUNTER_TEMPLATE_ARGS

```
#define COUNTER_TEMPLATE_ARGS( ) <typename Array_Type, typename Data_Type>
```

Definition at line 8 of file counters-meat.hpp.

#### 8.26.1.3 COUNTER_TYPE

```
#define COUNTER_TYPE( ) Counter<Array_Type,Data_Type>
```

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

#### 8.26.1.4 COUNTERS_TEMPLATE

```
#define COUNTERS_TEMPLATE(
              a,
              b )  template COUNTERS_TEMPLATE_ARGS() inline a COUNTERS_TYPE()::b
```

Definition at line 155 of file counters-meat.hpp.

#### 8.26.1.5 COUNTERS_TEMPLATE_ARGS

```
#define COUNTERS_TEMPLATE_ARGS( ) <typename Array_Type, typename Data_Type>
```

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

#### 8.26.1.6 COUNTERS_TYPE

```
#define COUNTERS_TYPE( ) Counters<Array_Type,Data_Type>
```

Definition at line 151 of file counters-meat.hpp.

### 8.26.2 Function Documentation

#### 8.26.2.1 count_fun()

```
return count_fun (
            Array ,
            i ,
            j ,
            data  )
```

#### 8.26.2.2 COUNTER_TEMPLATE() [1/7]

```
COUNTER_TEMPLATE (
            Counter  ) const
```

#### 8.26.2.3 COUNTER_TEMPLATE() [2/7]

```
COUNTER_TEMPLATE (
            COUNTER_TYPE() & ,
            operator  )
```

#### 8.26.2.4 COUNTER_TEMPLATE() [3/7]

```
COUNTER_TEMPLATE (
            COUNTER_TYPE() ,
            operator  ) const
```

#### 8.26.2.5 COUNTER_TEMPLATE() [4/7]

```
COUNTER_TEMPLATE (
            double ,
            count  ) &
```

< Move assignment

#### 8.26.2.6 COUNTER_TEMPLATE() [5/7]

```
COUNTER_TEMPLATE (
            double ,
            init  ) &
```

#### 8.26.2.7 COUNTER_TEMPLATE() [6/7]

```
COUNTER_TEMPLATE (
            std::string ,
            get_description  ) const
```

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

#### 8.26.2.8 COUNTER_TEMPLATE() [7/7]

```
COUNTER_TEMPLATE (
            std::string ,
            get_name  ) const
```

Definition at line 139 of file counters-meat.hpp.

#### 8.26.2.9 COUNTERS_TEMPLATE() [1/8]

```
COUNTERS_TEMPLATE (
            Counters  )
```

Definition at line 158 of file counters-meat.hpp.

#### 8.26.2.10 COUNTERS_TEMPLATE() [2/8]

```
COUNTERS_TEMPLATE (
            COUNTER_TYPE() & ,
            operator [] )
```

Definition at line 165 of file counters-meat.hpp.

#### 8.26.2.11 COUNTERS_TEMPLATE() [3/8]

```
COUNTERS_TEMPLATE (
            COUNTERS_TYPE() & ,
            operator  )
```

#### 8.26.2.12 COUNTERS_TEMPLATE() [4/8]

```
COUNTERS_TEMPLATE (
            COUNTERS_TYPE() ,
            operator  ) const
```

#### 8.26.2.13 COUNTERS_TEMPLATE() [5/8]

```
COUNTERS_TEMPLATE (
            std::vector< std::string > ,
            get_descriptions  ) const
```

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

#### 8.26.2.14 COUNTERS_TEMPLATE() [6/8]

```
COUNTERS_TEMPLATE (
            std::vector< std::string > ,
            get_names  ) const
```

Definition at line 344 of file counters-meat.hpp.

#### 8.26.2.15 COUNTERS_TEMPLATE() [7/8]

```
COUNTERS_TEMPLATE (
            void ,
            add_counter  )
```

#### 8.26.2.16 COUNTERS_TEMPLATE() [8/8]

```
COUNTERS_TEMPLATE (
            void ,
            clear  )
```

Definition at line 320 of file counters-meat.hpp.

#### 8.26.2.17 data()

```
Data_Type&& counter_ data (
            std::move(counter_.data)  )
```

#### 8.26.2.18 delete_data() [1/3]

```
Data_Type&& counter_ delete_data (
            std::move(counter_.delete_data)  )
```

#### 8.26.2.19 delete_data() [2/3]

```
Data_Type&& counters_ delete_data (
            std::move(counters_.delete_data)  )
```

#### 8.26.2.20 delete_data() [3/3]

```
Data_Type Data_Type delete_data (
            true  )
```

#### 8.26.2.21 delete_to_be_deleted() [1/2]

```
Data_Type&& counters_ delete_to_be_deleted (
             std::move(counters_.delete_to_be_deleted)  )
```

Definition at line 203 of file counters-meat.hpp.

#### 8.26.2.22 delete_to_be_deleted() [2/2]

```
Data_Type Data_Type delete_to_be_deleted (
             true  )
```

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

#### 8.26.2.23 desc()

```
Data_Type&& counter_ desc (
             std::move(counter_.desc)  )
```

Move constructor.

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

#### 8.26.2.24 init_fun() [1/3]

```
return init_fun (
             Array ,
             i ,
             j ,
             data  )
```

#### 8.26.2.25 init_fun() [2/3]

```
Data_Type init_fun (
             counter_.  init_fun )
```

Definition at line 15 of file counters-meat.hpp.

#### 8.26.2.26 init_fun() [3/3]

```
Data_Type&& counter_ init_fun (
            std::move(counter_.init_fun)  )
```

#### 8.26.2.27 name()

```
Data_Type&& counter_ name (
            std::move(counter_.name)  )
```

#### 8.26.2.28 push_back() [1/2]

```
data push_back (
            new Counter< Array_Type, Data_Type > count_fun_, init_fun_, data_, delete_data_,
name_, desc_ )
```

#### 8.26.2.29 push_back() [2/2]

```
data push_back (
            new Counter< Array_Type, Data_Type > counter )
```

#### 8.26.2.30 to_be_deleted() [1/2]

```
Data_Type Data_Type to_be_deleted (
            new std::vector< uint > 0u )
```

#### 8.26.2.31 to_be_deleted() [2/2]

```
Data_Type&& counters_ to_be_deleted (
            std::move(counters_.to_be_deleted)  )
```

### 8.26.3 Variable Documentation

#### 8.26.3.1 count_fun_

```
Data_Type count_fun_
```

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

#### 8.26.3.2 counter

```
Data_Type * counter
```

**Initial value:**

```
{
    to_be_deleted->push_back(data->size())
```

Definition at line 275 of file counters-meat.hpp.

#### 8.26.3.3 counter_

```
Data_Type & counter_
```

**Initial value:**

```
{
    if (this != &counter_) {
        this->count_fun = counter_.count_fun;
        this->init_fun = counter_.init_fun;
        if (counter_.delete_data)
        {
            this->data = new Data_Type(*counter_.data);
            this->delete_data = true;

        } else {
            this->data = counter_.data;
            this->delete_data = false;
        }
        this->name = counter_.name;
        this->desc = counter_.desc;
    }
    return *this
```

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

#### 8.26.3.4 data_

```
Data_Type Counter_fun_type<Array_Type,Data_Type> Data_Type* data_
```

Definition at line 295 of file counters-meat.hpp.

#### 8.26.3.5 delete_data_

```
Data_Type Counter_fun_type<Array_Type,Data_Type> Data_Type bool delete_data_
```

Definition at line 296 of file counters-meat.hpp.

#### 8.26.3.6 desc_

```
Data_Type Counter_fun_type<Array_Type,Data_Type> Data_Type bool std::string std::string desc↩
_
```

**Initial value:**

```
{

    to_be_deleted->push_back(data->size())
```

Definition at line 298 of file counters-meat.hpp.

#### 8.26.3.7 i

```
uint i
```

Definition at line 119 of file counters-meat.hpp.

#### 8.26.3.8 init_fun_

```
Data_Type Counter_fun_type<Array_Type,Data_Type> init_fun_
```

Definition at line 294 of file counters-meat.hpp.

#### 8.26.3.9 j

```
uint uint j
```

**Initial value:**

```
{
    if (count_fun == nullptr)
        return 0.0
```

Definition at line 119 of file counters-meat.hpp.

#### 8.26.3.10 name_

```
Data_Type Counter_fun_type<Array_Type,Data_Type> Data_Type bool std::string name_
```

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

#### 8.26.3.11 noexcept

```
Data_Type &&counters_ noexcept
```

**Initial value:**

```
{
    if (this != &counter_)
    {

        if (delete_data)
            delete data;

        this->data        = std::move(counter_.data);
        this->delete_data = std::move(counter_.delete_data);
        counter_.data        = nullptr;
        counter_.delete_data = false;

        this->count_fun = std::move(counter_.count_fun);
        this->init_fun = std::move(counter_.init_fun);

        this->name = std::move(counter_.name);
        this->desc = std::move(counter_.desc);
    }
    return *this
```

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

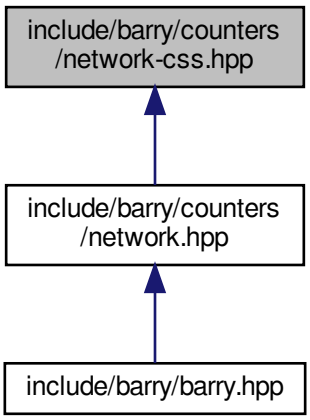
#### 8.26.3.12 return

```
return
```

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

## 8.27 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_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, uint netsize, const std::vector< uint > &end_)

  *Counts errors of commission.*
- template<typename Tnet = Network>
  void counter_css_partially_false_recip_omiss (NetCounters< Tnet > *counters, uint netsize, const std::vector< uint > &end_)

  *Counts errors of omission.*
- template<typename Tnet = Network>
  void counter_css_completely_false_recip_comiss (NetCounters< Tnet > *counters, uint netsize, const std::vector< uint > &end_)

  *Counts completely false reciprocity (comission)*

- template<typename Tnet = Network>
  void counter_css_completely_false_recip_omiss (NetCounters< Tnet > *counters, uint netsize, const std::vector< uint > &end_)

  *Counts completely false reciprocity (omission)*
- template<typename Tnet = Network>
  void counter_css_mixed_recip (NetCounters< Tnet > *counters, uint netsize, const std::vector< uint > &end_)

  *Counts mixed reciprocity errors.*
- template<typename Tnet = Network>
  void counter_css_census01 (NetCounters< Tnet > *counters, uint netsize, const std::vector< uint > &end_)
- template<typename Tnet = Network>
  void counter_css_census02 (NetCounters< Tnet > *counters, uint netsize, const std::vector< uint > &end_)
- template<typename Tnet = Network>
  void counter_css_census03 (NetCounters< Tnet > *counters, uint netsize, const std::vector< uint > &end_)
- template<typename Tnet = Network>
  void counter_css_census04 (NetCounters< Tnet > *counters, uint netsize, const std::vector< uint > &end_)
- template<typename Tnet = Network>
  void counter_css_census05 (NetCounters< Tnet > *counters, uint netsize, const std::vector< uint > &end_)
- template<typename Tnet = Network>
  void counter_css_census06 (NetCounters< Tnet > *counters, uint netsize, const std::vector< uint > &end_)
- template<typename Tnet = Network>
  void counter_css_census07 (NetCounters< Tnet > *counters, uint netsize, const std::vector< uint > &end_)
- template<typename Tnet = Network>
  void counter_css_census08 (NetCounters< Tnet > *counters, uint netsize, const std::vector< uint > &end_)
- template<typename Tnet = Network>
  void counter_css_census09 (NetCounters< Tnet > *counters, uint netsize, const std::vector< uint > &end_)
- template<typename Tnet = Network>
  void counter_css_census10 (NetCounters< Tnet > *counters, uint netsize, const std::vector< uint > &end_)

## 8.27.1 Macro Definition Documentation

### 8.27.1.1 CSS_APPEND

```
#define CSS_APPEND(
                name )
```

**Value:**

```
    std::string name_ = (name);\
    for (uint i = 0u; i < end_.size(); ++i) { \
    std::string tmpname = name_ + " (" + std::to_string(i) + ")";\
    counters->add_counter(tmp_count, tmp_init,\
            new NetCounterData({netsize, i == 0u ? netsize : end_[i-1], end_[i]}, {}),\
            true, tmpname);}
```

Definition at line 42 of file network-css.hpp.

#### 8.27.1.2 CSS_CASE_ELSE

```
#define CSS_CASE_ELSE( )
```

Definition at line 27 of file network-css.hpp.

#### 8.27.1.3 CSS_CASE_PERCEIVED

```
#define CSS_CASE_PERCEIVED( ) else if (((i >= s) && (i < e)) & ((j >= s) && (j < e)))
```

Definition at line 20 of file network-css.hpp.

#### 8.27.1.4 CSS_CASE_TRUTH

```
#define CSS_CASE_TRUTH( ) if ((i < n) && (j < n))
```

Definition at line 13 of file network-css.hpp.

#### 8.27.1.5 CSS_CHECK_SIZE

```
#define CSS_CHECK_SIZE( )
```

**Value:**

```
    for (uint 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 37 of file network-css.hpp.

#### 8.27.1.6 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 31 of file network-css.hpp.

#### 8.27.1.7 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 49 of file network-css.hpp.

#### 8.27.1.8 CSS_PERCEIVED_CELLS

```
#define CSS_PERCEIVED_CELLS( )
```

**Value:**

```
    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 21 of file network-css.hpp.

#### 8.27.1.9 CSS_SIZE

```
#define CSS_SIZE( )
```

**Value:**

```
    uint n = data->indices[0u]; \
    uint s = data->indices[1u]; \
    uint e = data->indices[2u];
```

Definition at line 7 of file network-css.hpp.

#### 8.27.1.10 CSS_TRUE_CELLS

```
#define CSS_TRUE_CELLS( )
```

**Value:**

```
    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 14 of file network-css.hpp.

### 8.27.2 Function Documentation

#### 8.27.2.1 counter_css_census01()

```
template<typename Tnet = Network>
void counter_css_census01 (
             NetCounters< Tnet > * counters,
             uint netsize,
             const std::vector< uint > & end_ )  [inline]
```

Definition at line 275 of file network-css.hpp.

#### 8.27.2.2 counter_css_census02()

```
template<typename Tnet = Network>
void counter_css_census02 (
             NetCounters< Tnet > * counters,
             uint netsize,
             const std::vector< uint > & end_ )  [inline]
```

Definition at line 325 of file network-css.hpp.

#### 8.27.2.3 counter_css_census03()

```
template<typename Tnet = Network>
void counter_css_census03 (
             NetCounters< Tnet > * counters,
             uint netsize,
             const std::vector< uint > & end_ )  [inline]
```

Definition at line 364 of file network-css.hpp.

#### 8.27.2.4 counter_css_census04()

```
template<typename Tnet = Network>
void counter_css_census04 (
             NetCounters< Tnet > * counters,
             uint netsize,
             const std::vector< uint > & end_ )  [inline]
```

Definition at line 403 of file network-css.hpp.

#### 8.27.2.5 counter_css_census05()

```
template<typename Tnet = Network>
void counter_css_census05 (
            NetCounters< Tnet > * counters,
            uint netsize,
            const std::vector< uint > & end_ )  [inline]
```

Definition at line 442 of file network-css.hpp.

#### 8.27.2.6 counter_css_census06()

```
template<typename Tnet = Network>
void counter_css_census06 (
            NetCounters< Tnet > * counters,
            uint netsize,
            const std::vector< uint > & end_ )  [inline]
```

Definition at line 481 of file network-css.hpp.

#### 8.27.2.7 counter_css_census07()

```
template<typename Tnet = Network>
void counter_css_census07 (
            NetCounters< Tnet > * counters,
            uint netsize,
            const std::vector< uint > & end_ )  [inline]
```

Definition at line 520 of file network-css.hpp.

#### 8.27.2.8 counter_css_census08()

```
template<typename Tnet = Network>
void counter_css_census08 (
            NetCounters< Tnet > * counters,
            uint netsize,
            const std::vector< uint > & end_ )  [inline]
```

Definition at line 559 of file network-css.hpp.

#### 8.27.2.9 counter_css_census09()

```
template<typename Tnet = Network>
void counter_css_census09 (
             NetCounters< Tnet > * counters,
             uint netsize,
             const std::vector< uint > & end_ )  [inline]
```

Definition at line 598 of file network-css.hpp.

#### 8.27.2.10 counter_css_census10()

```
template<typename Tnet = Network>
void counter_css_census10 (
             NetCounters< Tnet > * counters,
             uint netsize,
             const std::vector< uint > & end_ )  [inline]
```

Definition at line 637 of file network-css.hpp.

#### 8.27.2.11 counter_css_completely_false_recip_comiss()

```
template<typename Tnet = Network>
void counter_css_completely_false_recip_comiss (
             NetCounters< Tnet > * counters,
             uint netsize,
             const std::vector< uint > & end_ )  [inline]
```

Counts completely false reciprocity (comission)

Definition at line 154 of file network-css.hpp.

#### 8.27.2.12 counter_css_completely_false_recip_omiss()

```
template<typename Tnet = Network>
void counter_css_completely_false_recip_omiss (
             NetCounters< Tnet > * counters,
             uint netsize,
             const std::vector< uint > & end_ )  [inline]
```

Counts completely false reciprocity (omission)

Definition at line 194 of file network-css.hpp.

#### 8.27.2.13 counter_css_mixed_recip()

```
template<typename Tnet = Network>
void counter_css_mixed_recip (
            NetCounters< Tnet > * counters,
            uint netsize,
            const std::vector< uint > & end_ )  [inline]
```

Counts mixed reciprocity errors.

Definition at line 234 of file network-css.hpp.

#### 8.27.2.14 counter_css_partially_false_recip_commi()

```
template<typename Tnet = Network>
void counter_css_partially_false_recip_commi (
            NetCounters< Tnet > * counters,
            uint netsize,
            const std::vector< uint > & end_ )  [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) |

The `end_` parameter should be of length `N of networks` - 1. It is assumed that the first network ends at `netsize`.

Definition at line 63 of file network-css.hpp.

#### 8.27.2.15 counter_css_partially_false_recip_omiss()

```
template<typename Tnet = Network>
void counter_css_partially_false_recip_omiss (
            NetCounters< Tnet > * counters,
            uint netsize,
            const std::vector< uint > & end_ )  [inline]
```

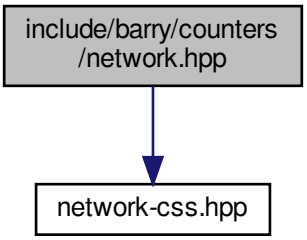
Counts errors of omission.

Definition at line 110 of file network-css.hpp.

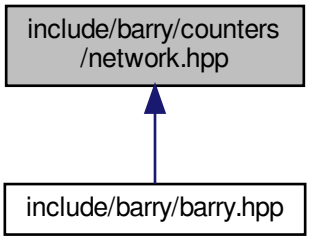
## 8.28 include/barry/counters/network.hpp File Reference

```
#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 uint or double vectors.*

### Macros

- #define NET_C_DATA_IDX(i) (data->indices[i])
- #define NET_C_DATA_NUM(i) (data->numbers[i])

### Macros for defining counters

- #define NETWORK_COUNTER(a)
- #define NETWORK_COUNTER_LAMBDA(a)
- #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, uint 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, uint attr_id, double alpha=1.0, double tail_head=true)

  *Sum of attribute difference between ego and alter to pow(alpha)*
- NETWORK_COUNTER (init_single_attr)
- template<typename Tnet = Network>
  void counter_nodeicov (NetCounters< Tnet > *counters, uint attr_id)
- template<typename Tnet = Network>
  void counter_nodeocov (NetCounters< Tnet > *counters, uint attr_id)

- template<typename Tnet = Network>
  void counter_nodecov (NetCounters< Tnet > *counters, uint attr_id)
- template<typename Tnet = Network>
  void counter_nodematch (NetCounters< Tnet > *counters, uint attr_id)
- template<typename Tnet = Network>
  void counter_idegree (NetCounters< Tnet > *counters, std::vector< uint > d)

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

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

  *Counts number of vertices with a given out-degree.*

**Rules for network models**

***Parameters***

| | |
|---|---|
| rules | *A pointer to a* `NetRules` *object (*`Rules<Network, bool>`*).* |

- 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 >

### 8.28.1 Macro Definition Documentation

#### 8.28.1.1 BARRY_ZERO_NETWORK

```
#define BARRY_ZERO_NETWORK 0.0
```

Definition at line 85 of file network.hpp.

#### 8.28.1.2 BARRY_ZERO_NETWORK_DENSE

```
#define BARRY_ZERO_NETWORK_DENSE 0
```

Definition at line 86 of file network.hpp.

#### 8.28.1.3 NET_C_DATA_IDX

```
#define NET_C_DATA_IDX(
              i ) (data->indices[i])
```

Definition at line 74 of file network.hpp.

#### 8.28.1.4 NET_C_DATA_NUM

```
#define NET_C_DATA_NUM(
              i ) (data->numbers[i])
```

Definition at line 75 of file network.hpp.

#### 8.28.1.5 NETWORK_COUNTER

```
#define NETWORK_COUNTER(
              a )
```

**Value:**

```
template<typename Tnet = Network>\
inline double (a) (const Tnet & Array, uint i, uint j, NetCounterData * data)
```

Function for definition of a network counter function

Definition at line 114 of file network.hpp.

#### 8.28.1.6 NETWORK_COUNTER_LAMBDA

```
#define NETWORK_COUNTER_LAMBDA(
                a )
```

**Value:**

```
Counter_fun_type<Tnet, NetCounterData> a = \
    [](const Tnet & Array, uint i, uint j, NetCounterData * data)
```

Lambda function for definition of a network counter function

Definition at line 119 of file network.hpp.

#### 8.28.1.7 NETWORK_RULE

```
#define NETWORK_RULE(
                a )
```

**Value:**

```
template<typename Tnet = Network>\
inline bool (a) (const Tnet & Array, uint i, uint j, bool * data)
```

Function for definition of a network counter function

Definition at line 133 of file network.hpp.

#### 8.28.1.8 NETWORK_RULE_LAMBDA

```
#define NETWORK_RULE_LAMBDA(
                a )
```

**Value:**

```
Rule_fun_type<Tnet, bool> a = \
[](const Tnet & Array, uint i, uint j, bool * data)
```

Lambda function for definition of a network counter function

Definition at line 138 of file network.hpp.

#### 8.28.1.9 NETWORKDENSE_COUNTER_LAMBDA

```
#define NETWORKDENSE_COUNTER_LAMBDA(
                a )
```

**Value:**

```
Counter_fun_type<NetworkDense, NetCounterData> a = \
    [](const NetworkDense & Array, uint i, uint j, NetCounterData * data)
```

Definition at line 123 of file network.hpp.

## 8.28.2 Typedef Documentation

### 8.28.2.1 NetCounter

```
template<typename Tnet = Network>
using NetCounter = Counter<Tnet, NetCounterData >
```

Definition at line 89 of file network.hpp.

### 8.28.2.2 NetCounters

```
template<typename Tnet = Network>
using NetCounters = Counters<Tnet, NetCounterData>
```

Definition at line 92 of file network.hpp.

### 8.28.2.3 NetModel

```
template<typename Tnet >
using NetModel = Model<Tnet, NetCounterData>
```

Definition at line 101 of file network.hpp.

### 8.28.2.4 NetRule

```
template<typename Tnet = Network>
using NetRule = Rule<Tnet, bool>
```

Definition at line 104 of file network.hpp.

### 8.28.2.5 NetRules

```
template<typename Tnet = Network>
using NetRules = Rules<Tnet, bool>
```

Definition at line 107 of file network.hpp.

#### 8.28.2.6 NetStatsCounter

```
template<typename Tnet = Network>
using NetStatsCounter = StatsCounter<Tnet, NetCounterData>
```

Definition at line 98 of file network.hpp.

#### 8.28.2.7 NetSupport

```
template<typename Tnet = Network>
using NetSupport = Support<Tnet, NetCounterData >
```

Definition at line 95 of file network.hpp.

#### 8.28.2.8 Network

```
typedef BArray<double, NetworkData> Network
```

Definition at line 82 of file network.hpp.

#### 8.28.2.9 NetworkDense

```
typedef BArrayDense<int, NetworkData> NetworkDense
```

Definition at line 83 of file network.hpp.

### 8.28.3 Function Documentation

#### 8.28.3.1 rules_zerodiag()

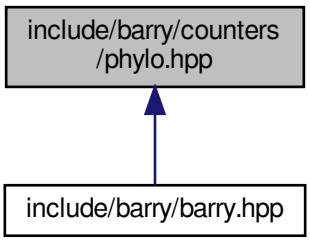
```
template<typename Tnet = Network>
void rules_zerodiag (
            NetRules< Tnet > * rules )  [inline]
```

Number of edges.

Definition at line 1325 of file network.hpp.

# 8.29 include/barry/counters/phylo.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 DEFAULT_DUPLICATION 1u
- #define DUPL_SPEC 0u
- #define DUPL_DUPL 1u
- #define DUPL_EITH 2u
- #define MAKE_DUPL_VARS()
- #define IS_EITHER() (DATA_AT == DUPL_EITH)
- #define IS_DUPLICATION() ((DATA_AT == DUPL_DUPL) & (DPL))
- #define IS_SPECIATION() ((DATA_AT == DUPL_SPEC) & (!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()

## Typedefs

- typedef std::vector< std::pair< uint, uint > > PhyloRuleData

**Convenient typedefs for Node objects.**

- typedef BArrayDense< uint, NodeData > PhyloArray
- typedef Counter< PhyloArray, PhyloCounterData > PhyloCounter
- typedef Counters< PhyloArray, PhyloCounterData > PhyloCounters
- typedef Rule< PhyloArray, PhyloRuleData > PhyloRule
- typedef Rules< PhyloArray, PhyloRuleData > PhyloRules
- typedef Rule< PhyloArray, PhyloRuleDynData > PhyloRuleDyn
- typedef Rules< PhyloArray, PhyloRuleDynData > PhyloRulesDyn
- typedef Support< PhyloArray, PhyloCounterData, PhyloRuleData, PhyloRuleDynData > PhyloSupport
- typedef StatsCounter< PhyloArray, PhyloCounterData > PhyloStatsCounter
- typedef Model< PhyloArray, PhyloCounterData, PhyloRuleData, PhyloRuleDynData > PhyloModel
- typedef PowerSet< PhyloArray, PhyloRuleData > PhyloPowerSet

## Functions

- std::string get_last_name (unsigned int d)
- void counter_overall_gains (PhyloCounters *counters, unsigned int duplication=DEFAULT_DUPLICATION)

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

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

  *k genes gain function nfun*
- void counter_genes_changing (PhyloCounters *counters, unsigned int duplication=DEFAULT_DUPLICATION)

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

  *Keeps track of how many pairs of genes preserve pseudostate.*
- void counter_prop_genes_changing (PhyloCounters *counters, unsigned int duplication=DEFAULT_DUPLICATION)

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

  *Overall functional loss.*
- void counter_maxfuns (PhyloCounters *counters, uint lb, uint ub, unsigned int duplication=DEFAULT_DUPLICATION)

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

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

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

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

  *Co-evolution (joint gain or loss)*
- void counter_longest (PhyloCounters *counters, unsigned int duplication=DEFAULT_DUPLICATION)

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

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

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

  *Function co-opting.*
- void counter_k_genes_changing (PhyloCounters *counters, unsigned int k, unsigned int duplication=DEFAULT_DUPLICATION)

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

- void counter_less_than_p_prop_genes_changing (PhyloCounters *counters, double p, unsigned int duplication=DEFAULT_DUPLICATION)

  *Indicator function. Equals to one if $k$ genes changed and zero otherwise.*
- void counter_gains_from_0 (PhyloCounters *counters, std::vector< uint > nfun, unsigned int duplication=DEFAULT_DUPLICAT

  *Used when all the functions are in 0 (like the root node prob.)*
- void counter_overall_gains_from_0 (PhyloCounters *counters, unsigned int duplication=DEFAULT_DUPLICATION)

  *Used when all the functions are in 0 (like the root node prob.)*
- void counter_pairwise_overall_change (PhyloCounters *counters, unsigned int duplication=DEFAULT_DUPLICATION)

  *Used when all the functions are in 0 (like the root node prob.)*
- void counter_pairwise_preserving (PhyloCounters *counters, uint nfunA, uint nfunB, unsigned int duplication=DEFAULT_DUPLICATION)

  *Used when all the functions are in 0 (like the root node prob.)*
- void counter_pairwise_first_gain (PhyloCounters *counters, uint nfunA, uint nfunB, unsigned int duplication=DEFAULT_DUPLICATION)

  *Used when all the functions are in 0 (like the root node prob.)*
- void rule_dyn_limit_changes (PhyloSupport *support, uint pos, uint lb, uint ub, unsigned int duplication=DEFAULT_DUPLICATIO

  *Overall functional gains.*

## 8.29.1 Macro Definition Documentation

### 8.29.1.1 DEFAULT_DUPLICATION

```
#define DEFAULT_DUPLICATION 1u
```

Definition at line 5 of file phylo.hpp.

### 8.29.1.2 DUPL_DUPL

```
#define DUPL_DUPL 1u
```

Definition at line 7 of file phylo.hpp.

### 8.29.1.3 DUPL_EITH

```
#define DUPL_EITH 2u
```

Definition at line 8 of file phylo.hpp.

#### 8.29.1.4 DUPL_SPEC

```
#define DUPL_SPEC 0u
```

Definition at line 6 of file phylo.hpp.

#### 8.29.1.5 IF_MATCHES

```
#define IF_MATCHES( )
```

**Value:**

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

Definition at line 19 of file phylo.hpp.

#### 8.29.1.6 IF_NOTMATCHES

```
#define IF_NOTMATCHES( )
```

**Value:**

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

Definition at line 21 of file phylo.hpp.

#### 8.29.1.7 IS_DUPLICATION

```
#define IS_DUPLICATION( ) ((DATA_AT == DUPL_DUPL) & (DPL))
```

Definition at line 16 of file phylo.hpp.

#### 8.29.1.8 IS_EITHER

```
#define IS_EITHER( ) (DATA_AT == DUPL_EITH)
```

Definition at line 15 of file phylo.hpp.

#### 8.29.1.9 IS_SPECIATION

```
#define IS_SPECIATION( ) ((DATA_AT == DUPL_SPEC) & (!DPL))
```

Definition at line 17 of file phylo.hpp.

#### 8.29.1.10 MAKE_DUPL_VARS

```
#define MAKE_DUPL_VARS( )
```

**Value:**

```
    bool DPL = Array.D()->duplication; \
    unsigned int DATA_AT = data->operator[](0u);
```

Definition at line 11 of file phylo.hpp.

#### 8.29.1.11 PHYLO_CHECK_MISSING

```
#define PHYLO_CHECK_MISSING( )
```

**Value:**

```
    if (Array.D() == nullptr) \
    throw std::logic_error("The array data is nullptr."); \
    if (data == nullptr) \
    throw std::logic_error("The counter/rule data is nullptr.")
```

Definition at line 137 of file phylo.hpp.

#### 8.29.1.12 PHYLO_COUNTER_LAMBDA

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

**Value:**

```
    Counter_fun_type<PhyloArray, PhyloCounterData> a = \
    [](const PhyloArray & Array, uint i, uint j, PhyloCounterData * data)
```

Extension of a simple counter.

It allows specifying extra arguments, in particular, the corresponding sets of rows to which this statistic may be relevant. This could be important in the case of, for example, counting correlation type statistics between function 1 and 2, and between function 1 and 3.

Definition at line 131 of file phylo.hpp.

#### 8.29.1.13 PHYLO_RULE_DYN_LAMBDA

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

**Value:**

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

Definition at line 134 of file phylo.hpp.

### 8.29.2 Typedef Documentation

#### 8.29.2.1 PhyloArray

```
typedef BArrayDense<uint, NodeData> PhyloArray
```

Definition at line 104 of file phylo.hpp.

#### 8.29.2.2 PhyloCounter

```
typedef Counter<PhyloArray, PhyloCounterData > PhyloCounter
```

Definition at line 105 of file phylo.hpp.

#### 8.29.2.3 PhyloCounters

```
typedef Counters< PhyloArray, PhyloCounterData> PhyloCounters
```

Definition at line 106 of file phylo.hpp.

#### 8.29.2.4 PhyloModel

```
typedef Model<PhyloArray, PhyloCounterData, PhyloRuleData, PhyloRuleDynData > PhyloModel
```

Definition at line 116 of file phylo.hpp.

#### 8.29.2.5 PhyloPowerSet

```
typedef PowerSet<PhyloArray, PhyloRuleData> PhyloPowerSet
```

Definition at line 117 of file phylo.hpp.

#### 8.29.2.6 PhyloRule

```
typedef Rule<PhyloArray,PhyloRuleData> PhyloRule
```

Definition at line 108 of file phylo.hpp.

#### 8.29.2.7 PhyloRuleData

```
typedef std::vector< std::pair< uint, uint > > PhyloRuleData
```

Definition at line 97 of file phylo.hpp.

#### 8.29.2.8 PhyloRuleDyn

```
typedef Rule<PhyloArray,PhyloRuleDynData> PhyloRuleDyn
```

Definition at line 111 of file phylo.hpp.

#### 8.29.2.9 PhyloRules

```
typedef Rules<PhyloArray,PhyloRuleData> PhyloRules
```

Definition at line 109 of file phylo.hpp.

#### 8.29.2.10 PhyloRulesDyn

```
typedef Rules<PhyloArray,PhyloRuleDynData> PhyloRulesDyn
```

Definition at line 112 of file phylo.hpp.

#### 8.29.2.11 PhyloStatsCounter

```
typedef StatsCounter<PhyloArray, PhyloCounterData> PhyloStatsCounter
```

Definition at line 115 of file phylo.hpp.

#### 8.29.2.12 PhyloSupport

```
typedef Support<PhyloArray, PhyloCounterData, PhyloRuleData, PhyloRuleDynData > PhyloSupport
```

Definition at line 114 of file phylo.hpp.

### 8.29.3 Function Documentation

#### 8.29.3.1 get_last_name()

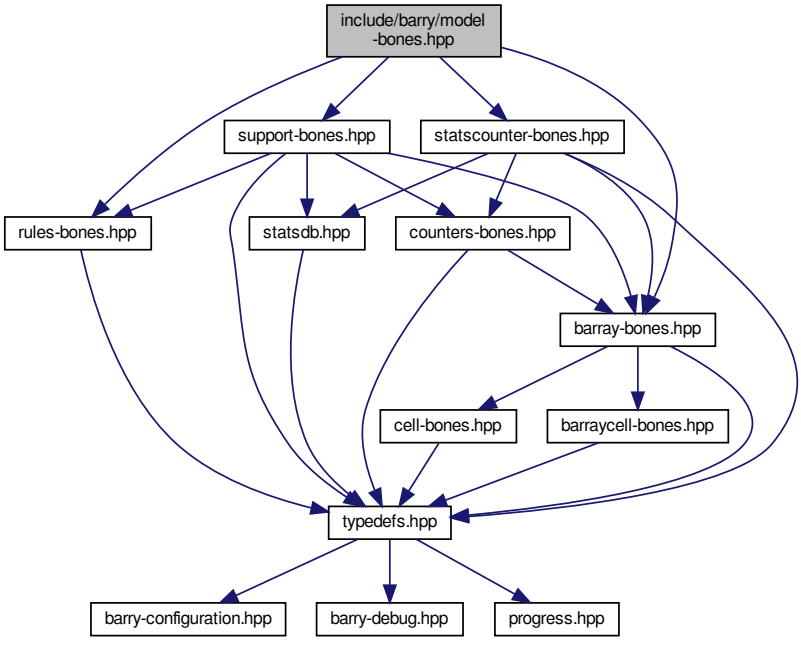
```
std::string get_last_name (
            unsigned int d )  [inline]
```

Definition at line 142 of file phylo.hpp.

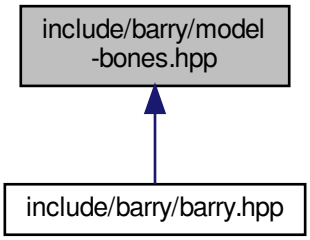
## 8.30 include/barry/model-bones.hpp File Reference

```
#include "barray-bones.hpp"
#include "support-bones.hpp"
#include "statscounter-bones.hpp"
#include "rules-bones.hpp"
```

Include dependency graph for model-bones.hpp:



This graph shows which files directly or indirectly include this file:




## Classes

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

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

## Functions

- template<typename Array_Type >
  std::vector< double > keygen_default (const Array_Type &Array_)

  *Array Hasher class (used for computing support)*

## 8.30.1 Function Documentation

### 8.30.1.1 keygen_default()

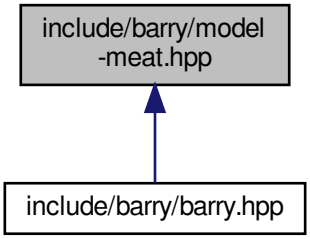
```
template<typename Array_Type >
std::vector< double > keygen_default (
            const Array_Type & Array_ )  [inline]
```

Array Hasher class (used for computing support)

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

## 8.31 include/barry/model-meat.hpp File Reference

This graph shows which files directly or indirectly include this file:




### Macros

- #define MODEL_TYPE()
- #define MODEL_TEMPLATE_ARGS()
- #define MODEL_TEMPLATE(a, b) template MODEL_TEMPLATE_ARGS() inline a MODEL_TYPE()::b

### Functions

- double update_normalizing_constant (const std::vector< double > &params, const std::vector< double > &support)
- double likelihood_ (const std::vector< double > &stats_target, const std::vector< double > &params, const double normalizing_constant, bool log_=false)
- MODEL_TEMPLATE (, Model)()
- MODEL_TEMPLATE (, Model)(const MODEL_TYPE() &Model_)

### 8.31.1 Macro Definition Documentation

#### 8.31.1.1 MODEL_TEMPLATE

```
#define MODEL_TEMPLATE(
              a,
              b )  template MODEL_TEMPLATE_ARGS() inline a MODEL_TYPE()::b
```

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

#### 8.31.1.2 MODEL_TEMPLATE_ARGS

```
#define MODEL_TEMPLATE_ARGS( )
```

**Value:**

```
    <typename Array_Type, typename Data_Counter_Type,\
    typename Data_Rule_Type, typename Data_Rule_Dyn_Type>
```

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

#### 8.31.1.3 MODEL_TYPE

```
#define MODEL_TYPE( )
```

**Value:**

```
    Model<Array_Type,Data_Counter_Type,Data_Rule_Type,\
    Data_Rule_Dyn_Type>
```

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

### 8.31.2 Function Documentation

#### 8.31.2.1 likelihood_()

```
double likelihood_ (
            const std::vector< double > & stats_target,
            const std::vector< double > & params,
            const double normalizing_constant,
            bool log_ = false )  [inline]
```

Definition at line 56 of file model-meat.hpp.

#### 8.31.2.2 MODEL_TEMPLATE() [1/2]

```
MODEL_TEMPLATE (
            Model  )
```

Definition at line 101 of file model-meat.hpp.

#### 8.31.2.3 MODEL_TEMPLATE() [2/2]

```
MODEL_TEMPLATE (
            Model  ) const &
```

Definition at line 159 of file model-meat.hpp.

#### 8.31.2.4 update_normalizing_constant()

```
double update_normalizing_constant (
            const std::vector< double > & params,
            const std::vector< double > & support )  [inline]
```

Definition at line 11 of file model-meat.hpp.

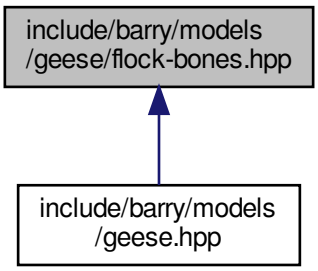
## 8.32 include/barry/models/geese.hpp File Reference

```
#include "geese/geese-node-bones.hpp"
#include "geese/geese-bones.hpp"
#include "geese/geese-meat.hpp"
#include "geese/geese-meat-constructors.hpp"
#include "geese/geese-meat-likelihood.hpp"
#include "geese/geese-meat-likelihood_exhaust.hpp"
#include "geese/geese-meat-simulate.hpp"
#include "geese/geese-meat-predict.hpp"
#include "geese/geese-meat-predict_exhaust.hpp"
#include "geese/geese-meat-predict_sim.hpp"
#include "geese/flock-bones.hpp"
#include "geese/flock-meat.hpp"
```

Include dependency graph for geese.hpp:

## 8.33 include/barry/models/geese/flock-bones.hpp File Reference

This graph shows which files directly or indirectly include this file:



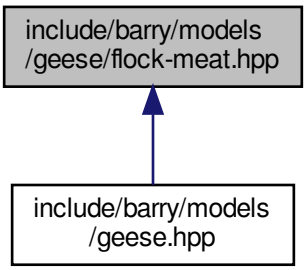

### Classes

- class Flock

  *A Flock is a group of Geese.*

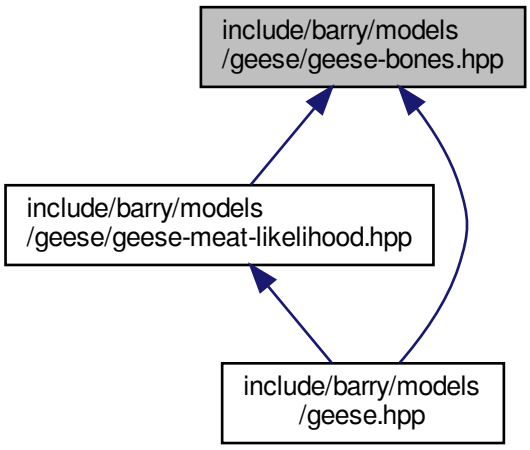
## 8.34 include/barry/models/geese/flock-meat.hpp File Reference

This graph shows which files directly or indirectly include this file:



## 8.35 include/barry/models/geese/geese-bones.hpp File Reference

This graph shows which files directly or indirectly include this file:




### Classes

- class Geese

  *Annotated Phylo Model.*

### Macros

- #define INITIALIZED()

### Functions

- template<typename Ta , typename Tb >
  std::vector< Ta > vector_caster (const std::vector< Tb > &x)
- RULE_FUNCTION (rule_empty_free)
- std::vector< double > keygen_full (const phylocounters::PhyloArray &array)
- bool vec_diff (const std::vector< unsigned int > &s, const std::vector< unsigned int > &a)

### 8.35.1 Macro Definition Documentation

#### 8.35.1.1 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.

### 8.35.2 Function Documentation

#### 8.35.2.1 keygen_full()

```
std::vector< double > keygen_full (
              const phylocounters::PhyloArray & array )  [inline]
```

Definition at line 35 of file geese-bones.hpp.

#### 8.35.2.2 RULE_FUNCTION()

```
RULE_FUNCTION (
              rule_empty_free  )
```

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

#### 8.35.2.3 vec_diff()

```
bool vec_diff (
              const std::vector< unsigned int > & s,
              const std::vector< unsigned int > & a )  [inline]
```

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

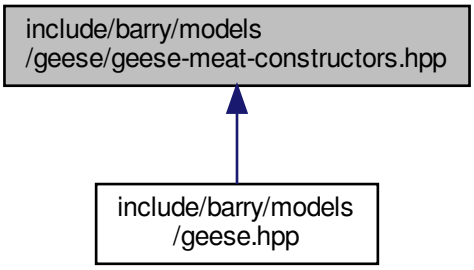
#### 8.35.2.4 vector_caster()

```
template<typename Ta , typename Tb >
std::vector< Ta > vector_caster (
              const std::vector< Tb > & x )  [inline]
```

Definition at line 10 of file geese-bones.hpp.

## 8.36 include/barry/models/geese/geese-meat-constructors.hpp File Reference

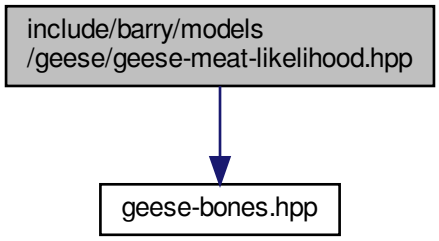
This graph shows which files directly or indirectly include this file:



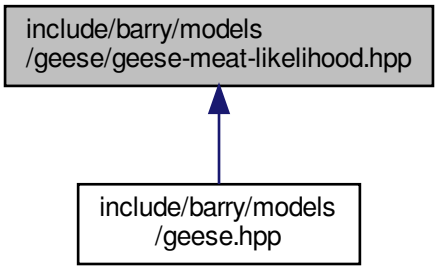

## 8.37 include/barry/models/geese/geese-meat-likelihood.hpp File Reference

```
#include "geese-bones.hpp"
```

Include dependency graph for geese-meat-likelihood.hpp:

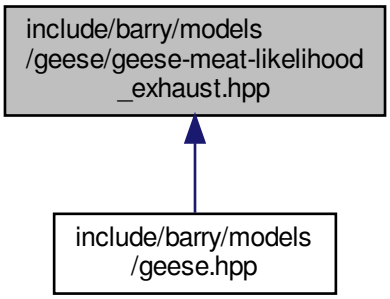


This graph shows which files directly or indirectly include this file:



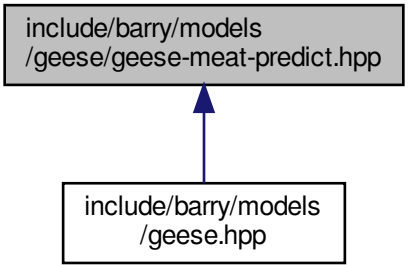

## 8.38 include/barry/models/geese/geese-meat-likelihood_exhaust.hpp File Reference

This graph shows which files directly or indirectly include this file:



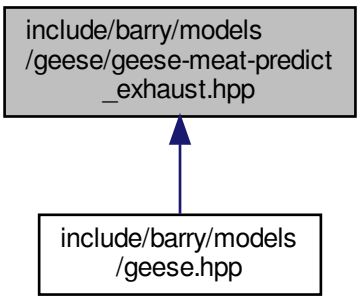
## 8.39 include/barry/models/geese/geese-meat-predict.hpp File Reference

This graph shows which files directly or indirectly include this file:



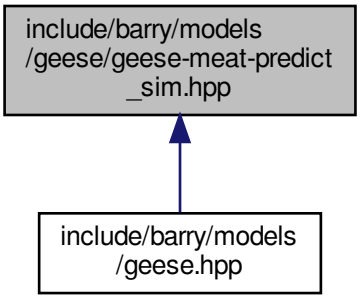

## 8.40 include/barry/models/geese/geese-meat-predict_exhaust.hpp File Reference

This graph shows which files directly or indirectly include this file:



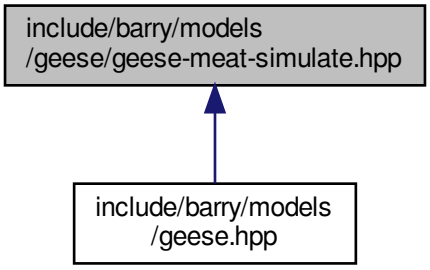
## 8.41 include/barry/models/geese/geese-meat-predict_sim.hpp File Reference

This graph shows which files directly or indirectly include this file:



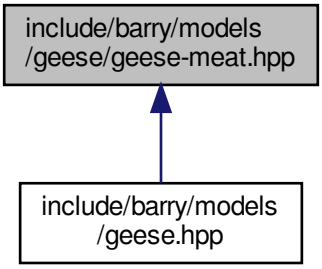

## 8.42 include/barry/models/geese/geese-meat-simulate.hpp File Reference

This graph shows which files directly or indirectly include this file:



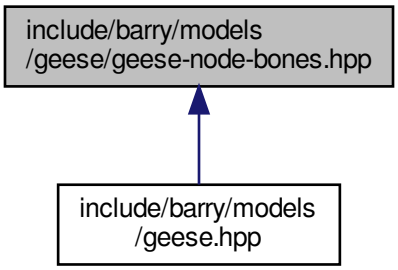
## 8.43 include/barry/models/geese/geese-meat.hpp File Reference

This graph shows which files directly or indirectly include this file:




## 8.44 include/barry/models/geese/geese-node-bones.hpp File Reference

This graph shows which files directly or indirectly include this file:




### Classes

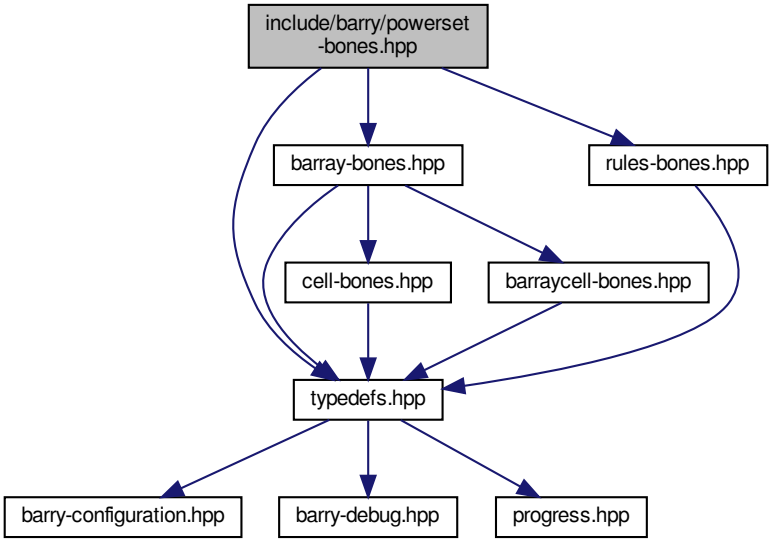
- class Node

  *A single node for the model.*

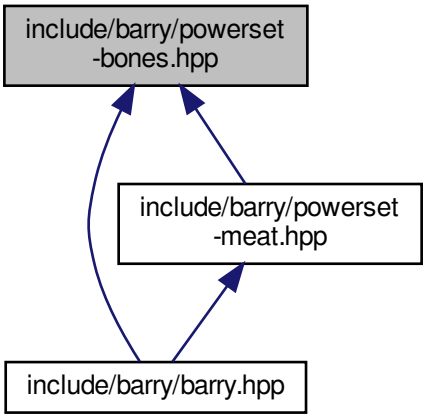
## 8.45 include/barry/powerset-bones.hpp File Reference

```
#include "typedefs.hpp"
#include "barray-bones.hpp"
#include "rules-bones.hpp"
```

Include dependency graph for powerset-bones.hpp:




This graph shows which files directly or indirectly include this file:




### Classes

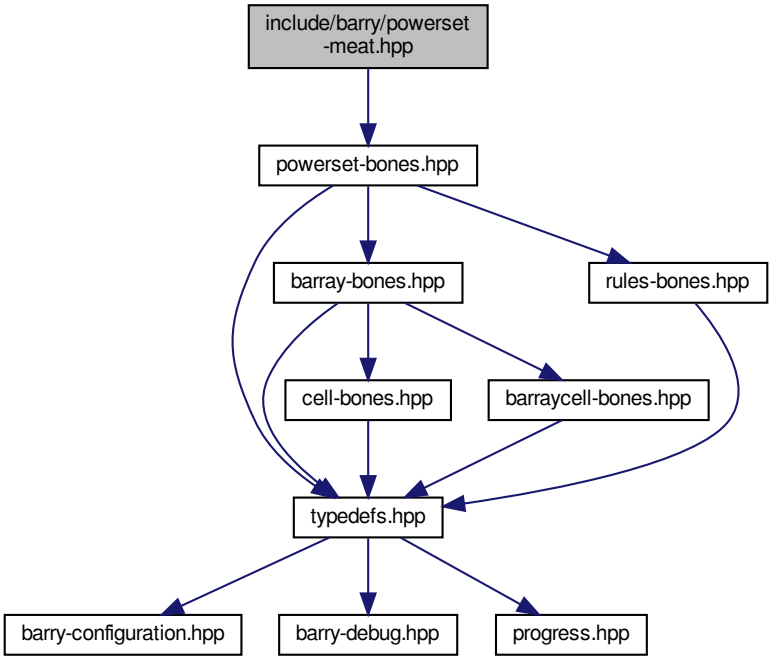
- class PowerSet< Array_Type, Data_Rule_Type >

*Powerset of a binary array.*

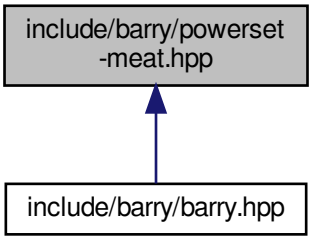
## 8.46 include/barry/powerset-meat.hpp File Reference

```
#include "powerset-bones.hpp"
```

Include dependency graph for powerset-meat.hpp:

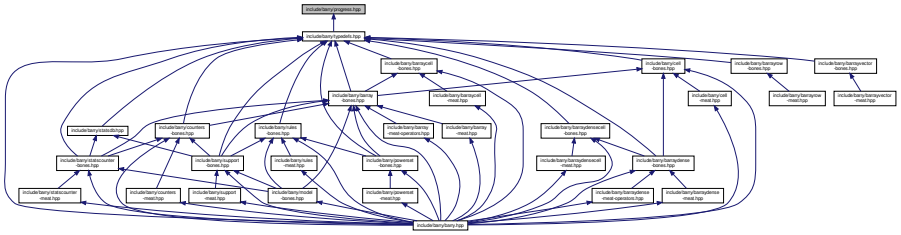



This graph shows which files directly or indirectly include this file:



# 8.47 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

### 8.47.1 Macro Definition Documentation

#### 8.47.1.1 BARRY_PROGRESS_BAR_WIDTH

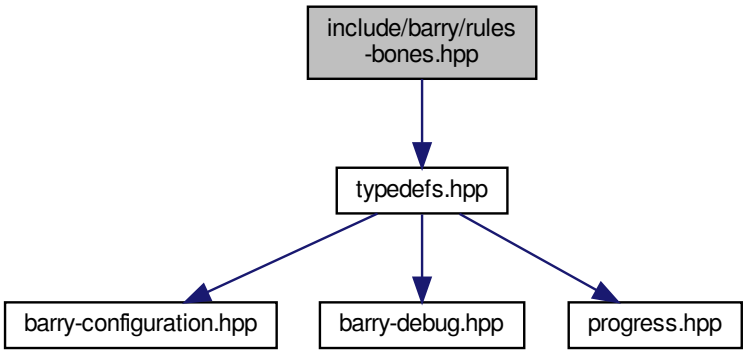
```
#define BARRY_PROGRESS_BAR_WIDTH 80
```

Definition at line 5 of file progress.hpp.

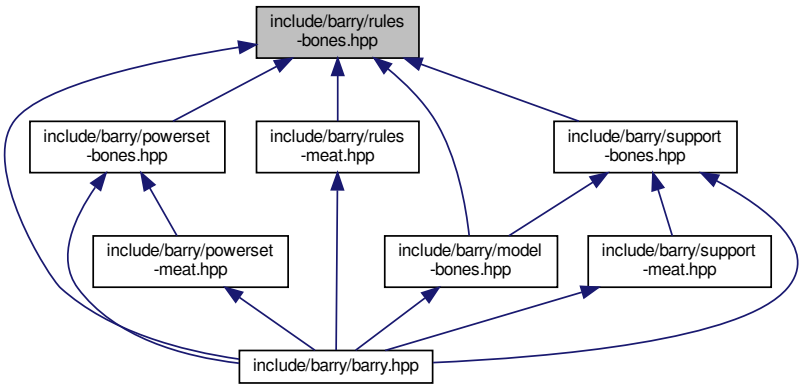
# 8.48 include/barry/rules-bones.hpp File Reference

```
#include "typedefs.hpp"
```

Include dependency graph for rules-bones.hpp:



This graph shows which files directly or indirectly include this file:




## Classes

- class Rule< Array_Type, Data_Type >

  *Rule for determining if a cell should be included in a sequence.*
- class Rules< Array_Type, Data_Type >

  *Vector of objects of class Rule.*

## Functions

- template<typename Array_Type , typename Data_Type >
  bool rule_fun_default (const Array_Type *array, uint i, uint j, Data_Type *dat)

## 8.48.1 Function Documentation

### 8.48.1.1 rule_fun_default()

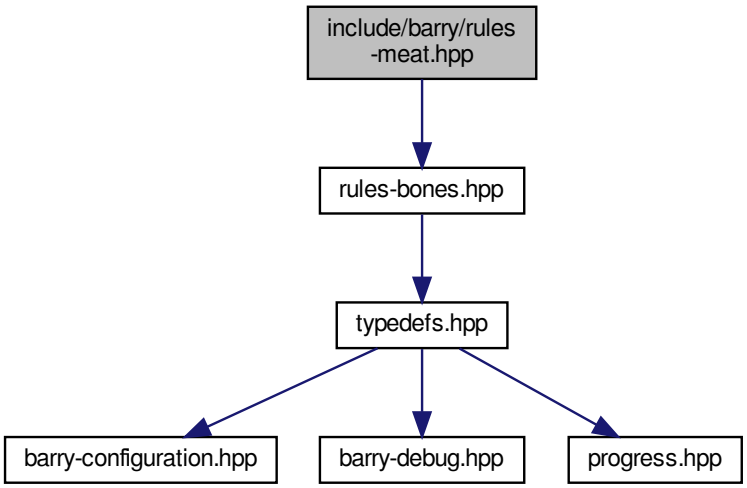
```
template<typename Array_Type , typename Data_Type >
bool rule_fun_default (
            const Array_Type * array,
            uint i,
            uint j,
            Data_Type * dat )
```

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

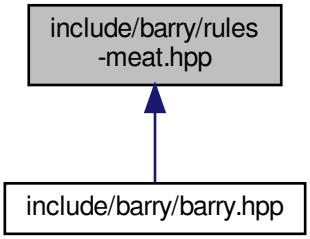
## 8.49 include/barry/rules-meat.hpp File Reference

```
#include "rules-bones.hpp"
```

Include dependency graph for rules-meat.hpp:




This graph shows which files directly or indirectly include this file:

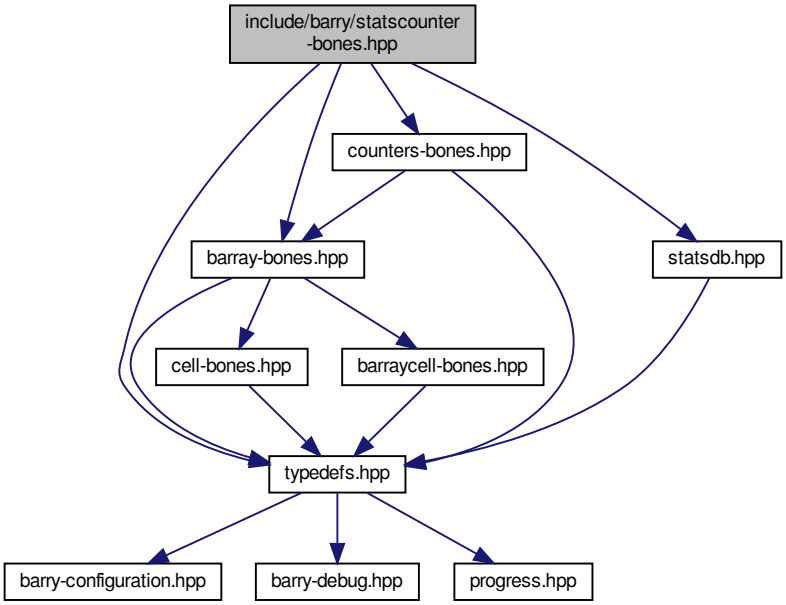



## 8.50 include/barry/statscounter-bones.hpp File Reference

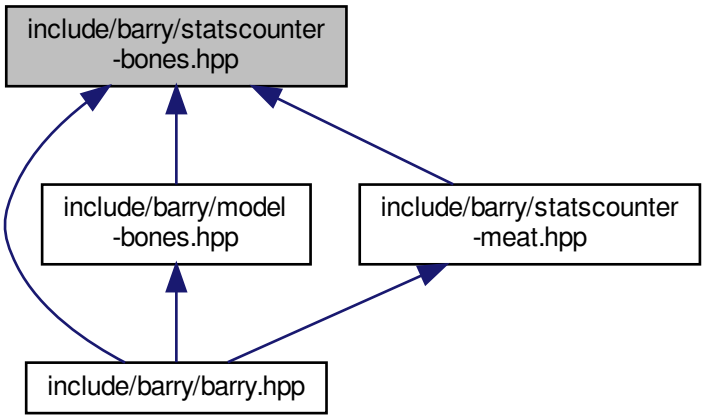
```
#include "typedefs.hpp"
#include "barray-bones.hpp"
#include "statsdb.hpp"
```

```
#include "counters-bones.hpp"
```

Include dependency graph for statscounter-bones.hpp:




This graph shows which files directly or indirectly include this file:



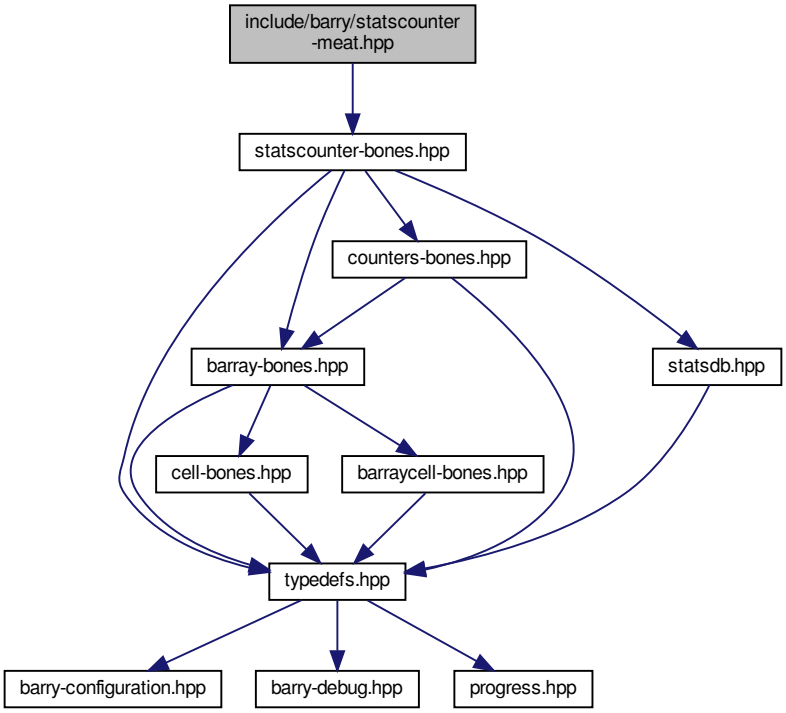

## Classes

- class StatsCounter< Array_Type, Data_Type >

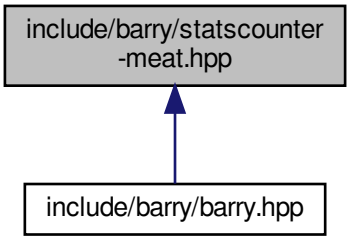
  *Count stats for a single Array.*

## 8.51 include/barry/statscounter-meat.hpp File Reference

`#include "statscounter-bones.hpp"`
Include dependency graph for statscounter-meat.hpp:




This graph shows which files directly or indirectly include this file:




### 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)(uint i
- current_stats resize (counters->size(), 0.0)
- for (uint n=0u;n< counters->size();++n) current_stats[n]
- STATSCOUNTER_TEMPLATE (void, count_current)(uint 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_
- uint j

## 8.51.1 Macro Definition Documentation

### 8.51.1.1 STATSCOUNTER_TEMPLATE

```
#define STATSCOUNTER_TEMPLATE(
              a,
              b )  template STATSCOUNTER_TEMPLATE_ARGS() inline a STATSCOUNTER_TYPE()::b
```

Definition at line 11 of file statscounter-meat.hpp.

### 8.51.1.2 STATSCOUNTER_TEMPLATE_ARGS

```
template STATSCOUNTER_TEMPLATE_ARGS( ) <typename Array_Type, typename Data_Type>
```

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

#### 8.51.1.3 STATSCOUNTER_TYPE

```
template Data_Type * STATSCOUNTER_TYPE( ) StatsCounter<Array_Type,Data_Type>
```

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

### 8.51.2 Function Documentation

#### 8.51.2.1 clear()

```
EmptyArray clear ( )
```

#### 8.51.2.2 for()

```
for (
            uint n = 0u;n< counters->size();++n )
```

#### 8.51.2.3 resize()

```
current_stats resize (
            counters-> size(),
            0.  0 )
```

#### 8.51.2.4 STATSCOUNTER_TEMPLATE() [1/9]

```
STATSCOUNTER_TEMPLATE (
            StatsCounter  ) const
```

#### 8.51.2.5 STATSCOUNTER_TEMPLATE() [2/9]

```
STATSCOUNTER_TEMPLATE (
            ~ StatsCounter )
```

Definition at line 30 of file statscounter-meat.hpp.

#### 8.51.2.6 STATSCOUNTER_TEMPLATE() [3/9]

```
STATSCOUNTER_TEMPLATE (
            std::vector< std::string > ,
            get_descriptions  ) const
```

Definition at line 267 of file statscounter-meat.hpp.

#### 8.51.2.7 STATSCOUNTER_TEMPLATE() [4/9]

```
STATSCOUNTER_TEMPLATE (
            std::vector< std::string > ,
            get_names  ) const
```

Definition at line 262 of file statscounter-meat.hpp.

#### 8.51.2.8 STATSCOUNTER_TEMPLATE() [5/9]

```
STATSCOUNTER_TEMPLATE (
            void ,
            add_counter  )
```

#### 8.51.2.9 STATSCOUNTER_TEMPLATE() [6/9]

```
STATSCOUNTER_TEMPLATE (
            void ,
            count_current  )
```

#### 8.51.2.10 STATSCOUNTER_TEMPLATE() [7/9]

```
STATSCOUNTER_TEMPLATE (
            void ,
            count_init  )
```

#### 8.51.2.11 STATSCOUNTER_TEMPLATE() [8/9]

```
STATSCOUNTER_TEMPLATE (
            void ,
            reset_array  ) const
```

Definition at line 37 of file statscounter-meat.hpp.

#### 8.51.2.12 STATSCOUNTER_TEMPLATE() [9/9]

```
STATSCOUNTER_TEMPLATE (
            void ,
            set_counters  )
```

### 8.51.3 Variable Documentation

#### 8.51.3.1 counter

```
Data_Type& counter
```

**Initial value:**

```
{
    Array      = counter.Array
```

Definition at line 15 of file statscounter-meat.hpp.

#### 8.51.3.2 counter_deleted

```
counter_deleted = false
```

Definition at line 26 of file statscounter-meat.hpp.

#### 8.51.3.3 counters

```
counters = new Counters<Array_Type,Data_Type>((*counter.counters))
```

Definition at line 25 of file statscounter-meat.hpp.

#### 8.51.3.4 counters_

```
Data_Type* counters_
```

**Initial value:**

```
{

    if (!counter_deleted)
        delete counters
```

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

#### 8.51.3.5 current_stats

```
current_stats = counter.current_stats
```

Definition at line 22 of file statscounter-meat.hpp.

#### 8.51.3.6 EmptyArray

```
EmptyArray = *Array
```

Definition at line 20 of file statscounter-meat.hpp.

#### 8.51.3.7 f_

```
Data_Rule_Dyn_Type f_
```

**Initial value:**

```
{
    counters->add_counter(f_)
```

Definition at line 47 of file statscounter-meat.hpp.

#### 8.51.3.8 j

```
uint j
```

**Initial value:**

```
{

    if (counters->size() == 0u)
        throw std::logic_error("No counters added: Cannot count without knowning what to count!")
```

Definition at line 77 of file statscounter-meat.hpp.

#### 8.51.3.9 return

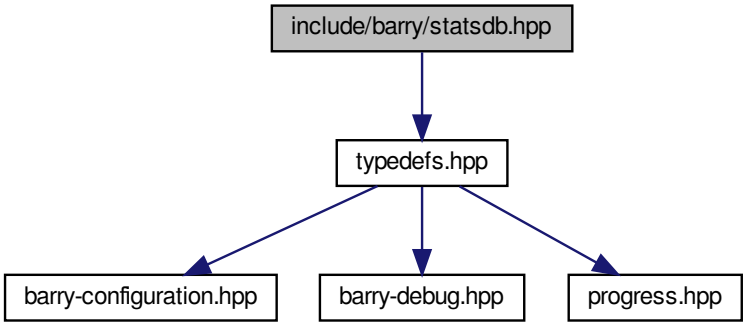
```
return
```

Definition at line 51 of file statscounter-meat.hpp.

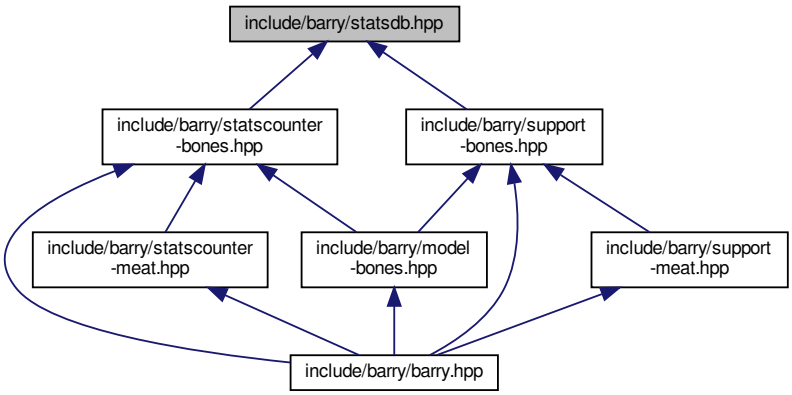
## 8.52 include/barry/statsdb.hpp File Reference

```
#include "typedefs.hpp"
```

Include dependency graph for statsdb.hpp:




This graph shows which files directly or indirectly include this file:




### Classes

- class FreqTable< T >

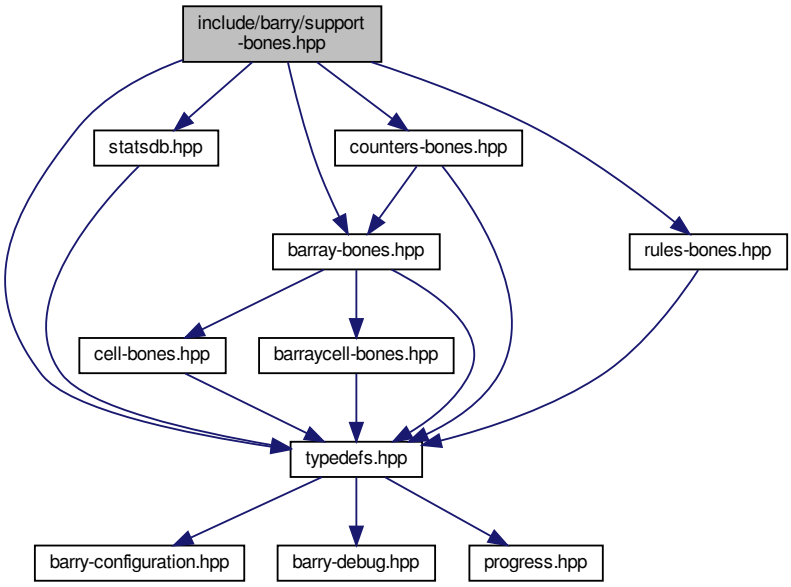
  *Database of statistics.*

## 8.53 include/barry/support-bones.hpp File Reference

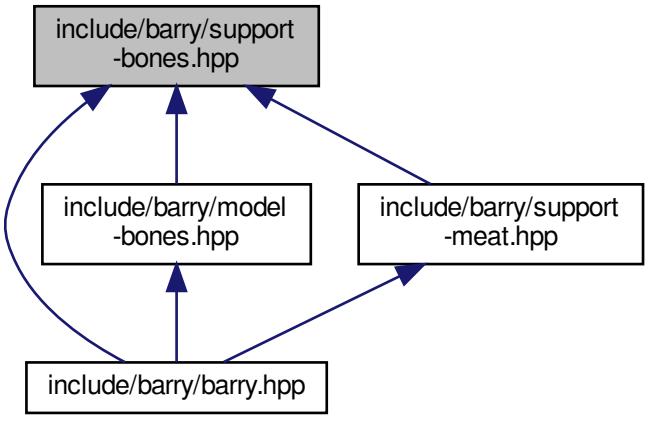
```
#include "typedefs.hpp"
#include "barray-bones.hpp"
```

```
#include "statsdb.hpp"
#include "counters-bones.hpp"
#include "rules-bones.hpp"
```

Include dependency graph for support-bones.hpp:




This graph shows which files directly or indirectly include this file:




## Classes

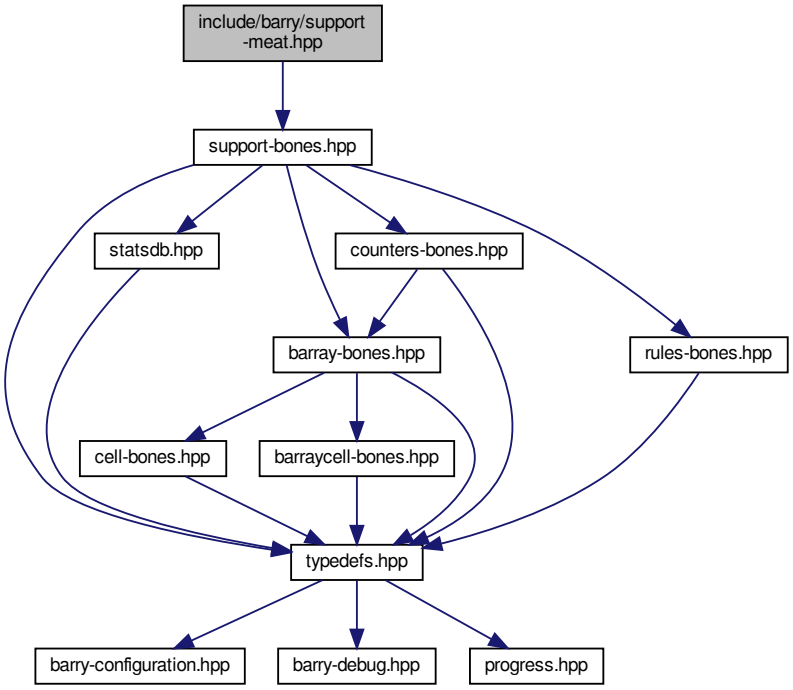
- class Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >

  *Compute the support of sufficient statistics.*

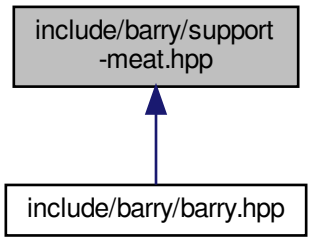
## 8.54 include/barry/support-meat.hpp File Reference

```
#include "support-bones.hpp"
```

Include dependency graph for support-meat.hpp:




This graph shows which files directly or indirectly include this file:




### Macros

- #define BARRY_SUPPORT_MEAT_HPP 1
- #define SUPPORT_TEMPLATE_ARGS()
- #define SUPPORT_TYPE()
- #define SUPPORT_TEMPLATE(a, b)

## Functions

- SUPPORT_TEMPLATE (void, init_support)(std
- SUPPORT_TEMPLATE (void, reset_array)()
- SUPPORT_TEMPLATE (void, reset_array)(const Array_Type &Array_)
- SUPPORT_TEMPLATE (void, calc_backend_sparse)(uint pos
- calc_backend_sparse (pos+1u, array_bank, stats_bank)
- EmptyArray insert_cell (coord_i, coord_j, EmptyArray.default_val().value, false, false)
- for (uint n=0u;n< n_counters;++n)
- if (rules_dyn->size() > 0u)
- if (array_bank !=nullptr) array_bank -> push_back(EmptyArray)
- if (stats_bank !=nullptr) stats_bank -> push_back(current_stats)
- EmptyArray rm_cell (coord_i, coord_j, false, false)
- if (change_stats_different > 0u)
- SUPPORT_TEMPLATE (void, calc_backend_dense)(uint pos
- calc_backend_dense (pos+1u, array_bank, stats_bank)
- EmptyArray insert_cell (coord_i, coord_j, 1, false, false)
- SUPPORT_TEMPLATE (void, calc)(std
- SUPPORT_TEMPLATE (void, add_counter)(Counter< Array_Type
- SUPPORT_TEMPLATE (void, set_counters)(Counters< Array_Type
- SUPPORT_TEMPLATE (void, add_rule)(Rule< Array_Type
- SUPPORT_TEMPLATE (void, set_rules)(Rules< Array_Type
- SUPPORT_TEMPLATE (void, add_rule_dyn)(Rule< Array_Type
- SUPPORT_TEMPLATE (void, set_rules_dyn)(Rules< Array_Type
- SUPPORT_TEMPLATE (bool, eval_rules_dyn)(const std
- SUPPORT_TEMPLATE (std::vector< double >, get_counts)() const
- SUPPORT_TEMPLATE (std::vector< double > *, get_current_stats)()
- SUPPORT_TEMPLATE (void, print)() const
- SUPPORT_TEMPLATE (const FreqTable<> &, get_data)() const

## Variables

- std::vector< Array_Type > * array_bank
- std::vector< Array_Type > std::vector< std::vector< double > > * stats_bank
- const size_t & coord_i = coordinates_free[pos * 2u]
- const size_t & coord_j = coordinates_free[pos * 2u + 1u]
- double tmp_chng
- unsigned int change_stats_different = hashes_initialized[pos] ? 0u : 1u
- else
- & hashes [pos]
- return
- Data_Counter_Type * f_
- Data_Counter_Type * counters_
- delete_counters = false
- counters = counters_
- Data_Rule_Type * rules_
- delete_rules = false
- rules = rules_
- delete_rules_dyn = false
- rules_dyn = rules_

### 8.54.1 Macro Definition Documentation

#### 8.54.1.1 BARRY_SUPPORT_MEAT_HPP

```
#define BARRY_SUPPORT_MEAT_HPP 1
```

Definition at line 4 of file support-meat.hpp.

#### 8.54.1.2 SUPPORT_TEMPLATE

```
#define SUPPORT_TEMPLATE(
              a,
              b )
```

**Value:**

```
template SUPPORT_TEMPLATE_ARGS() \
inline a SUPPORT_TYPE()::b
```

Definition at line 12 of file support-meat.hpp.

#### 8.54.1.3 SUPPORT_TEMPLATE_ARGS

```
template SUPPORT_TEMPLATE_ARGS( )
```

**Value:**

```
<typename Array_Type, typename \
Data_Counter_Type, typename Data_Rule_Type, typename Data_Rule_Dyn_Type>
```

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

#### 8.54.1.4 SUPPORT_TYPE

```
template Data_Rule_Dyn_Type * SUPPORT_TYPE( )
```

**Value:**

```
Support<Array_Type,Data_Counter_Type,Data_Rule_Type,\
Data_Rule_Dyn_Type>
```

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

### 8.54.2 Function Documentation

#### 8.54.2.1 calc_backend_dense()

```
calc_backend_dense (
            pos+ 1u,
            array_bank ,
            stats_bank  )
```

#### 8.54.2.2 calc_backend_sparse()

```
calc_backend_sparse (
            pos+ 1u,
            array_bank ,
            stats_bank  )
```

#### 8.54.2.3 for()

```
for ( )
```

Definition at line 162 of file support-meat.hpp.

#### 8.54.2.4 if() [1/4]

```
if (
            array_bank !    = nullptr ) -> push_back(EmptyArray)
```

#### 8.54.2.5 if() [2/4]

```
if (
            change_stats_different ,
            0u  )
```

Definition at line 243 of file support-meat.hpp.

#### 8.54.2.6 if() [3/4]

```
if (
            rules_dyn-> size(),
            0u  )
```

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

#### 8.54.2.7 if() [4/4]

```
if (
            stats_bank !     = nullptr ) -> push_back(current_stats)
```

#### 8.54.2.8 insert_cell() [1/2]

```
EmptyArray insert_cell (
            coord_i ,
            coord_j ,
            1 ,
            false ,
            false  )
```

#### 8.54.2.9 insert_cell() [2/2]

```
EmptyArray insert_cell (
            coord_i ,
            coord_j ,
            EmptyArray.default_val().  value,
            false ,
            false  )
```

#### 8.54.2.10 rm_cell()

```
EmptyArray rm_cell (
            coord_i ,
            coord_j ,
            false ,
            false  )
```

#### 8.54.2.11 SUPPORT_TEMPLATE() [1/17]

```
SUPPORT_TEMPLATE (
            bool ,
            eval_rules_dyn  ) const
```

Definition at line 499 of file support-meat.hpp.

#### 8.54.2.12 SUPPORT_TEMPLATE() [2/17]

```
SUPPORT_TEMPLATE (
            const FreqTable<> & ,
            get_data  ) const
```

Definition at line 548 of file support-meat.hpp.

#### 8.54.2.13 SUPPORT_TEMPLATE() [3/17]

```
SUPPORT_TEMPLATE (
            std::vector< double > * ,
            get_current_stats  )
```

Definition at line 533 of file support-meat.hpp.

#### 8.54.2.14 SUPPORT_TEMPLATE() [4/17]

```
SUPPORT_TEMPLATE (
            std::vector< double > ,
            get_counts  ) const
```

Definition at line 521 of file support-meat.hpp.

#### 8.54.2.15 SUPPORT_TEMPLATE() [5/17]

```
SUPPORT_TEMPLATE (
            void ,
            add_counter  )
```

#### 8.54.2.16 SUPPORT_TEMPLATE() [6/17]

```
SUPPORT_TEMPLATE (
            void ,
            add_rule  )
```

#### 8.54.2.17 SUPPORT_TEMPLATE() [7/17]

```
SUPPORT_TEMPLATE (
            void ,
            add_rule_dyn  )
```

#### 8.54.2.18 SUPPORT_TEMPLATE() [8/17]

```
SUPPORT_TEMPLATE (
            void ,
            calc  )
```

Definition at line 373 of file support-meat.hpp.

#### 8.54.2.19 SUPPORT_TEMPLATE() [9/17]

```
SUPPORT_TEMPLATE (
            void ,
            calc_backend_dense  )
```

#### 8.54.2.20 SUPPORT_TEMPLATE() [10/17]

```
SUPPORT_TEMPLATE (
            void ,
            calc_backend_sparse  )
```

#### 8.54.2.21 SUPPORT_TEMPLATE() [11/17]

```
SUPPORT_TEMPLATE (
            void ,
            init_support  )
```

Definition at line 16 of file support-meat.hpp.

#### 8.54.2.22 SUPPORT_TEMPLATE() [12/17]

```
SUPPORT_TEMPLATE (
            void ,
            print  ) const
```

Definition at line 537 of file support-meat.hpp.

#### 8.54.2.23 SUPPORT_TEMPLATE() [13/17]

```
SUPPORT_TEMPLATE (
            void ,
            reset_array  )
```

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

#### 8.54.2.24 SUPPORT_TEMPLATE() [14/17]

```
SUPPORT_TEMPLATE (
            void ,
            reset_array  ) const &
```

Definition at line 123 of file support-meat.hpp.

#### 8.54.2.25 SUPPORT_TEMPLATE() [15/17]

```
SUPPORT_TEMPLATE (
            void ,
            set_counters  )
```

#### 8.54.2.26 SUPPORT_TEMPLATE() [16/17]

```
SUPPORT_TEMPLATE (
            void ,
            set_rules  )
```

#### 8.54.2.27 SUPPORT_TEMPLATE() [17/17]

```
SUPPORT_TEMPLATE (
            void ,
            set_rules_dyn  )
```

### 8.54.3 Variable Documentation

#### 8.54.3.1 array_bank

```
std::vector< Array_Type > * array_bank
```

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

#### 8.54.3.2 change_stats_different

```
unsigned int change_stats_different = hashes_initialized[pos] ?  0u :  1u
```

Definition at line 161 of file support-meat.hpp.

#### 8.54.3.3 coord_i

```
const size_t & coord_i = coordinates_free[pos * 2u]
```

Definition at line 147 of file support-meat.hpp.

#### 8.54.3.4 coord_j

```
const size_t & coord_j = coordinates_free[pos * 2u + 1u]
```

Definition at line 148 of file support-meat.hpp.

#### 8.54.3.5 counters

```
counters = counters_
```

Definition at line 427 of file support-meat.hpp.

#### 8.54.3.6 counters_

```
Data_Counter_Type* counters_
```

**Initial value:**

```
{

    if (delete_counters)
        delete counters
```

Definition at line 420 of file support-meat.hpp.

#### 8.54.3.7 delete_counters

```
delete_counters = false
```

Definition at line 426 of file support-meat.hpp.

#### 8.54.3.8 delete_rules

```
delete_rules = false
```

Definition at line 460 of file support-meat.hpp.

#### 8.54.3.9 delete_rules_dyn

```
delete_rules_dyn = false
```

Definition at line 492 of file support-meat.hpp.

#### 8.54.3.10 else

```
else (
             void  )
```

**Initial value:**

```
{
        if (change_stats_different > 0u)
            hashes[pos] = data.add(current_stats, nullptr)
```

Definition at line 216 of file support-meat.hpp.

#### 8.54.3.11 f_

```
Data_Rule_Dyn_Type f_
```

**Initial value:**

```
{

    counters->add_counter(f_)
```

Definition at line 402 of file support-meat.hpp.

#### 8.54.3.12 hashes

```
& hashes
```

Definition at line 221 of file support-meat.hpp.

#### 8.54.3.13 return

```
return
```

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

#### 8.54.3.14 rules

```
rules = rules_
```

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

#### 8.54.3.15 rules_

```
Data_Rule_Dyn_Type * rules_
```

**Initial value:**

```
{

    if (delete_rules)
        delete rules
```

Definition at line 454 of file support-meat.hpp.

#### 8.54.3.16 rules_dyn

```
rules_dyn = rules_
```

Definition at line 493 of file support-meat.hpp.

#### 8.54.3.17 stats_bank

```
std::vector< Array_Type > std::vector< std::vector< double > > * stats_bank
```

**Initial value:**
```
{

    if (pos >= coordiantes_n_free)
        return
```

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

#### 8.54.3.18 tmp_chng

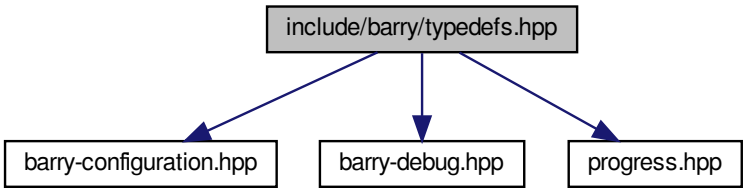
```
double tmp_chng
```

Definition at line 160 of file support-meat.hpp.

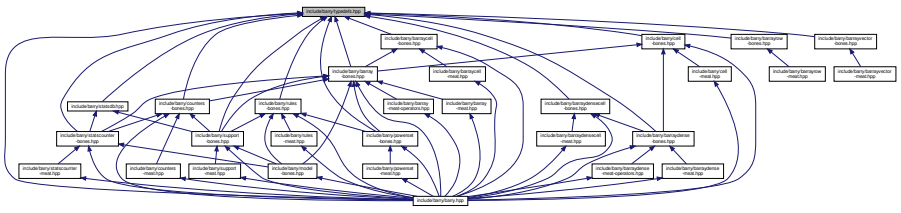
## 8.55 include/barry/typedefs.hpp File Reference

```
#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 unsigned int uint
- typedef std::vector< std::pair< std::vector< double >, uint > > Counts_type
- template<typename Cell_Type >
  using Row_type = Map< uint, Cell< Cell_Type > >
- template<typename Cell_Type >
  using Col_type = Map< uint, Cell< Cell_Type > * >
- template<typename Ta = double, typename Tb = uint>
  using MapVec_type = std::unordered_map< std::vector< Ta >, Tb, vecHasher< Ta > >
- template<typename Array_Type , typename Data_Type >
  using Counter_fun_type = std::function< double(const Array_Type &, uint, uint, Data_Type *)>

  *Counter and rule functions.*
- template<typename Array_Type , typename Data_Type >
  using Rule_fun_type = std::function< bool(const Array_Type &, uint, uint, Data_Type *)>

## Functions

- template<typename T >
  T vec_inner_prod (const std::vector< T > &a, const std::vector< T > &b)
- template<> double vec_inner_prod (const std::vector< double > &a, const std::vector< double > &b)
- template<typename T >
  bool vec_equal (const std::vector< T > &a, const std::vector< T > &b)

  *Compares if -a- and -b- are equal.*
- template<typename T >
  bool vec_equal_approx (const std::vector< T > &a, const std::vector< T > &b, double eps=1e-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

### 8.55.1 Typedef Documentation

#### 8.55.1.1 Col_type

```
template<typename Cell_Type >
using Col_type = Map< uint, Cell<Cell_Type>* >
```

Definition at line 71 of file typedefs.hpp.

#### 8.55.1.2 Counter_fun_type

```
template<typename Array_Type , typename Data_Type >
using Counter_fun_type = std::function<double(const Array_Type &, uint, uint, Data_Type *)>
```

Counter and rule functions.

**Parameters**

| | |
|---|---|
| *Array_Type* | a BArray |
| *unit,uint* | Focal cell |
| *Data_Type* | Data associated with the function, for example, id of the attribute in the Array. |

**Returns**

`Counter_fun_type` a double (the change statistic)

`Rule_fun_type` a bool. True if the cell is blocked.

Definition at line 148 of file typedefs.hpp.

#### 8.55.1.3 Counts_type

```
typedef std::vector< std::pair< std::vector<double>, uint > > Counts_type
```

Definition at line 52 of file typedefs.hpp.

#### 8.55.1.4 MapVec_type

```
template<typename Ta = double, typename Tb = uint>
using MapVec_type = std::unordered_map< std::vector< Ta >, Tb, vecHasher<Ta> >
```

Definition at line 129 of file typedefs.hpp.

#### 8.55.1.5 Row_type

```
template<typename Cell_Type >
using Row_type = Map< uint, Cell<Cell_Type> >
```

Definition at line 68 of file typedefs.hpp.

#### 8.55.1.6 Rule_fun_type

```
template<typename Array_Type , typename Data_Type >
using Rule_fun_type = std::function<bool(const Array_Type &, uint, uint, Data_Type *)>
```

Definition at line 151 of file typedefs.hpp.

#### 8.55.1.7 uint

```
typedef unsigned int uint
```

Definition at line 18 of file typedefs.hpp.

### 8.55.2 Function Documentation

#### 8.55.2.1 vec_equal()

```
template<typename T >
bool vec_equal (
            const std::vector< T > & a,
            const std::vector< T > & b )  [inline]
```

Compares if -a- and -b- are equal.

**Parameters**

| | |
|---|---|
| *a,b* | Two vectors of the same length |

**Returns**

`true` if all elements are equal.

Definition at line 162 of file typedefs.hpp.

#### 8.55.2.2 vec_equal_approx()

```
template<typename T >
bool vec_equal_approx (
            const std::vector< T > & a,
            const std::vector< T > & b,
            double eps = 1e-100 )  [inline]
```

Definition at line 180 of file typedefs.hpp.

#### 8.55.2.3 vec_inner_prod() [1/2]

```
template<>
double vec_inner_prod (
            const std::vector< double > & a,
            const std::vector< double > & b )  [inline]
```

Definition at line 223 of file typedefs.hpp.

#### 8.55.2.4 vec_inner_prod() [2/2]

```
template<typename T >
T vec_inner_prod (
            const std::vector< T > & a,
            const std::vector< T > & b )  [inline]
```

Definition at line 200 of file typedefs.hpp.

## 8.56 README.md File Reference

# Index


~BArray
BArray< Cell_Type, Data_Type >, 37
~BArrayCell
BArrayCell< Cell_Type, Data_Type >, 48
~BArrayCell_const
BArrayCell_const< Cell_Type, Data_Type >, 50
~BArrayDense
BArrayDense< Cell_Type, Data_Type >, 56
~BArrayDenseCell
BArrayDenseCell< Cell_Type, Data_Type >, 68
~BArrayRow
BArrayRow< Cell_Type, Data_Type >, 81
~BArrayRow_const
BArrayRow_const< Cell_Type, Data_Type >, 83
~BArrayVector
BArrayVector< Cell_Type, Data_Type >, 86
~BArrayVector_const
BArrayVector_const< Cell_Type, Data_Type >, 89
~Cell
Cell< Cell_Type >, 93
~ConstBArrayRowIter
ConstBArrayRowIter< Cell_Type, Data_Type >, 98
~Counter
Counter< Array_Type, Data_Type >, 101
~Counters
Counters< Array_Type, Data_Type >, 105
~Entries
Entries< Cell_Type >, 110
~Flock
Flock, 112
~FreqTable
FreqTable< T >, 119
~Geese
Geese, 124
~Model
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 137
~NetCounterData
NetCounterData, 148
~NetworkData
NetworkData, 150
~Node
Node, 153
~PhyloRuleDynData
PhyloRuleDynData, 162
~PowerSet
PowerSet< Array_Type, Data_Rule_Type >, 165
~Progress
Progress, 170
~Rule
Rule< Array_Type, Data_Type >, 172
~Rules
Rules< Array_Type, Data_Type >, 174
~StatsCounter
StatsCounter< Array_Type, Data_Type >, 178
~Support
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 183

active
Cell< Cell_Type >, 96
add
barray-meat.hpp, 212
barraydense-meat.hpp, 239
Cell< Cell_Type >, 94, 95
FreqTable< T >, 119
add_array
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 137
add_counter
Counters< Array_Type, Data_Type >, 106
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 137, 138
StatsCounter< Array_Type, Data_Type >, 179
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 184
add_data
Flock, 112
add_rule
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 138
PowerSet< Array_Type, Data_Rule_Type >, 166
Rules< Array_Type, Data_Type >, 175
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 184
add_rule_dyn
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 138, 139
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 184

annotations
Node, 154
ans
barray-meat.hpp, 203, 212
barraydense-meat.hpp, 229, 240
Array
ConstBArrayRowIter< Cell_Type, Data_Type >, 98
array
Node, 154
Array_
barray-meat.hpp, 212
array_bank
support-meat.hpp, 334
arrays
Node, 154
AS_ONE
EXISTS, 31
as_vector
FreqTable< T >, 119
AS_ZERO
EXISTS, 31
at
PhyloCounterData, 159

BArray
BArray< Cell_Type, Data_Type >, 36, 37
BArray< Cell_Type, Data_Type >, 33
∼BArray, 37
BArray, 36, 37
BArrayCell< Cell_Type, Data_Type >, 46
BArrayCell_const< Cell_Type, Data_Type >, 46
clear, 37
col, 37
D, 38
default_val, 38
flush_data, 38
get_cell, 38
get_col_vec, 38, 39
get_entries, 39
get_row_vec, 39
insert_cell, 39, 40
is_dense, 40
is_empty, 40
ncol, 40
nnozero, 41
nrow, 41
operator∗=, 41
operator(), 41
operator+=, 41, 42
operator-=, 42
operator/=, 42
operator=, 43
operator==, 43
out_of_range, 43
print, 43
reserve, 43
resize, 44
rm_cell, 44
row, 44
set_data, 44
swap_cells, 45
swap_cols, 45
swap_rows, 45
toggle_cell, 45
toggle_lock, 45
transpose, 46
visited, 47
zero_col, 46
zero_row, 46
barray-bones.hpp
BARRAY_BONES_HPP, 194
barray-meat-operators.hpp
BARRAY_TEMPLATE, 196–198
BARRAY_TEMPLATE_ARGS, 196, 198
BARRAY_TYPE, 196, 198
BARRY_BARRAY_MEAT_OPERATORS_HPP, 196
COL, 197
for, 198
operator(), 199
rhs, 199
ROW, 197
this, 199
barray-meat.hpp
add, 212
ans, 203, 212
Array_, 212
BARRAY_TEMPLATE, 202–207
BARRAY_TEMPLATE_ARGS, 202
BARRAY_TYPE, 202
check_bounds, 212
check_exists, 212
COL, 203, 207
col0, 213
const, 213
copy_data, 213
data, 213
delete_data, 213
delete_data_, 214
else, 214
false, 214
first, 214
for, 207, 208
i1, 214
if, 208–210
j, 215
j0, 215
j1, 215
M, 211, 215
M_, 215
N, 216
NCells, 216
report, 216
resize, 211
return, 211, 216
ROW, 203, 211
row0, 216

search, 217
source, 217
target, 217
v, 217
value, 217
BARRAY_BONES_HPP
barray-bones.hpp, 194
BARRAY_TEMPLATE
barray-meat-operators.hpp, 196–198
barray-meat.hpp, 202–207
BARRAY_TEMPLATE_ARGS
barray-meat-operators.hpp, 196, 198
barray-meat.hpp, 202
BARRAY_TYPE
barray-meat-operators.hpp, 196, 198
barray-meat.hpp, 202
BArrayCell
BArrayCell< Cell_Type, Data_Type >, 48
BArrayCell< Cell_Type, Data_Type >, 47
∼BArrayCell, 48
BArray< Cell_Type, Data_Type >, 46
BArrayCell, 48
operator Cell_Type, 48
operator∗=, 48
operator+=, 48
operator-=, 49
operator/=, 49
operator=, 49
operator==, 49
BArrayCell_const
BArrayCell_const< Cell_Type, Data_Type >, 50
BArrayCell_const< Cell_Type, Data_Type >, 50
∼BArrayCell_const, 50
BArray< Cell_Type, Data_Type >, 46
BArrayCell_const, 50
operator Cell_Type, 51
operator!=, 51
operator<, 51
operator<=, 51
operator>, 51
operator>=, 52
operator==, 51
BArrayDense
BArrayDense< Cell_Type, Data_Type >, 55, 56
BArrayDense< Cell_Type, Data_Type >, 52
∼BArrayDense, 56
BArrayDense, 55, 56
BArrayDenseCell< Cell_Type, Data_Type >, 66, 70
BArrayDenseCol< Cell_Type, Data_Type >, 66, 73
BArrayDenseCol_const< Cell_Type, Data_Type >, 66
BArrayDenseRow< Cell_Type, Data_Type >, 66, 77
BArrayDenseRow_const< Cell_Type, Data_Type >, 66
clear, 56
col, 56, 57
colsum, 57
D, 57
default_val, 57
get_cell, 57
get_col_vec, 58
get_data, 58
get_entries, 58
get_row_vec, 58, 59
insert_cell, 59
is_dense, 59
is_empty, 59
ncol, 60
nnozero, 60
nrow, 60
operator∗=, 60
operator(), 60
operator+=, 61
operator-=, 61
operator/=, 62
operator=, 62
operator==, 62
out_of_range, 62
print, 62
reserve, 63
resize, 63
rm_cell, 63
row, 63
rowsum, 63
set_data, 64
swap_cells, 64
swap_cols, 64
swap_rows, 64
toggle_cell, 65
toggle_lock, 65
transpose, 65
visited, 67
zero_col, 65
zero_row, 65
barraydense-bones.hpp
BARRY_BARRAYDENSE_BONES_HPP, 221
barraydense-meat-operators.hpp
BARRY_BARRAYDENSE_MEAT_OPERATORS_HPP, 222
BDENSE_TEMPLATE, 222–224
BDENSE_TEMPLATE_ARGS, 222, 224
BDENSE_TYPE, 222, 224
COL, 223
POS, 223
POS_N, 223
ROW, 223
barraydense-meat.hpp
add, 239
ans, 229, 240
BDENSE_TEMPLATE, 227, 229–236
BDENSE_TEMPLATE_ARGS, 227
BDENSE_TYPE, 228
check_bounds, 240
check_exists, 240

COL, 228
col, 240
const, 241
copy_data, 241
data, 241
delete_data, 241
delete_data_, 241
el, 242
el_colsums, 242
el_rowsums, 242
else, 242
false, 242
for, 236
i1, 243
if, 236
insert_cell, 237
j, 243
j0, 243
j1, 243
M, 237, 243
M_, 243
N, 244
POS, 228
POS_N, 228
report, 244
resize, 237, 238
return, 244
rm_cell, 238, 239
ROW, 228
source, 244
target, 244
v, 245
va_end, 239
va_start, 239
val0, 245
val1, 245
value, 245
vprintf, 239
ZERO_CELL, 229
BArrayDenseCell
BArrayDenseCell< Cell_Type, Data_Type >, 68
BArrayDenseCell< Cell_Type, Data_Type >, 67
~BArrayDenseCell, 68
BArrayDense< Cell_Type, Data_Type >, 66, 70
BArrayDenseCell, 68
BArrayDenseCol< Cell_Type, Data_Type >, 70, 73
BArrayDenseCol_const< Cell_Type, Data_Type >, 70, 75
BArrayDenseRow< Cell_Type, Data_Type >, 77
BArrayDenseRow_const< Cell_Type, Data_Type >, 80
operator Cell_Type, 68
operator*=, 68
operator+=, 69
operator-=, 69
operator/=, 69
operator=, 69
operator==, 69
barraydensecell-bones.hpp
POS, 246
barraydensecell-meat.hpp
POS, 248
BArrayDenseCell_const< Cell_Type, Data_Type >, 71
BArrayDenseCol< Cell_Type, Data_Type >, 73
BArrayDenseCol_const< Cell_Type, Data_Type >, 75
BArrayDenseRow< Cell_Type, Data_Type >, 77
BArrayDenseRow_const< Cell_Type, Data_Type >, 80
BArrayDenseCol
BArrayDenseCol< Cell_Type, Data_Type >, 71
BArrayDenseCol< Cell_Type, Data_Type >, 71
BArrayDense< Cell_Type, Data_Type >, 66, 73
BArrayDenseCell< Cell_Type, Data_Type >, 70, 73
BArrayDenseCell_const< Cell_Type, Data_Type >, 73
BArrayDenseCol, 71
begin, 72
end, 72
operator(), 72
size, 72
barraydensecol-bones.hpp
POS, 249
POS_N, 249
ZERO_CELL, 249
BArrayDenseCol_const
BArrayDenseCol_const< Cell_Type, Data_Type >, 74
BArrayDenseCol_const< Cell_Type, Data_Type >, 73
BArrayDense< Cell_Type, Data_Type >, 66
BArrayDenseCell< Cell_Type, Data_Type >, 70, 75
BArrayDenseCell_const< Cell_Type, Data_Type >, 75
BArrayDenseCol_const, 74
begin, 74
end, 74
operator(), 74
size, 75
BArrayDenseRow
BArrayDenseRow< Cell_Type, Data_Type >, 76
BArrayDenseRow< Cell_Type, Data_Type >, 75
BArrayDense< Cell_Type, Data_Type >, 66, 77
BArrayDenseCell< Cell_Type, Data_Type >, 77
BArrayDenseCell_const< Cell_Type, Data_Type >, 77
BArrayDenseRow, 76
begin, 76
end, 76
operator(), 77
size, 77
barraydenserow-bones.hpp
POS, 250
POS_N, 250
ZERO_CELL, 250

BArrayDenseRow_const
BArrayDenseRow_const< Cell_Type, Data_Type >, 78
BArrayDenseRow_const< Cell_Type, Data_Type >, 78
BArrayDense< Cell_Type, Data_Type >, 66
BArrayDenseCell< Cell_Type, Data_Type >, 80
BArrayDenseCell_const< Cell_Type, Data_Type >, 80
BArrayDenseRow_const, 78
begin, 79
end, 79
operator(), 79
size, 79
BArrayRow
BArrayRow< Cell_Type, Data_Type >, 81
BArrayRow< Cell_Type, Data_Type >, 80
~BArrayRow, 81
BArrayRow, 81
operator BArrayRow< Cell_Type, Data_Type >, 81
operator*=, 81
operator+=, 81
operator-=, 81
operator/=, 82
operator=, 82
operator==, 82
barrayrow-meat.hpp
BARRY_BARRAYROW_MEAT_HPP, 252
BROW_TEMPLATE, 252–254
BROW_TEMPLATE_ARGS, 253
BROW_TYPE, 253
BArrayRow_const
BArrayRow_const< Cell_Type, Data_Type >, 83
BArrayRow_const< Cell_Type, Data_Type >, 82
~BArrayRow_const, 83
BArrayRow_const, 83
operator BArrayRow_const< Cell_Type, Data_Type >, 83
operator!=, 83
operator<, 83
operator<=, 84
operator>, 84
operator>=, 84
operator==, 84
BArrayVector
BArrayVector< Cell_Type, Data_Type >, 85
BArrayVector< Cell_Type, Data_Type >, 84
~BArrayVector, 86
BArrayVector, 85
begin, 86
end, 86
is_col, 86
is_row, 86
operator std::vector< Cell_Type >, 87
operator*=, 87
operator+=, 87
operator-=, 87
operator/=, 87
operator=, 88
operator==, 88
size, 88
barrayvector-meat.hpp
BARRY_BARRAYVECTOR_MEAT_HPP, 256
BArrayVector_const
BArrayVector_const< Cell_Type, Data_Type >, 89
BArrayVector_const< Cell_Type, Data_Type >, 88
~BArrayVector_const, 89
BArrayVector_const, 89
begin, 90
end, 90
is_col, 90
is_row, 90
operator std::vector< Cell_Type >, 90
operator!=, 90
operator<, 91
operator<=, 91
operator>, 91
operator>=, 91
operator==, 91
size, 92
barry, 29
barry-configuration.hpp
BARRY_CHECK_SUPPORT, 257
BARRY_ISFINITE, 257
BARRY_MAX_NUM_ELEMENTS, 257
BARRY_SAFE_EXP, 257
Map, 257
printf_barry, 257
barry-debug.hpp
BARRY_DEBUG_LEVEL, 258
barry-macros.hpp
BARRY_ONE, 259
BARRY_ONE_DENSE, 259
BARRY_UNUSED, 259
BARRY_ZERO, 259
BARRY_ZERO_DENSE, 259
barry.hpp
BARRY_HPP, 261
BARRY_VERSION, 261
COUNTER_FUNCTION, 261
COUNTER_LAMBDA, 262
RULE_FUNCTION, 262
RULE_LAMBDA, 262
barry::counters, 29
barry::counters::network, 30
barry::counters::phylo, 30
BARRY_BARRAY_MEAT_OPERATORS_HPP
barray-meat-operators.hpp, 196
BARRY_BARRAYDENSE_BONES_HPP
barraydense-bones.hpp, 221
BARRY_BARRAYDENSE_MEAT_OPERATORS_HPP
barraydense-meat-operators.hpp, 222
BARRY_BARRAYROW_MEAT_HPP
barrayrow-meat.hpp, 252
BARRY_BARRAYVECTOR_MEAT_HPP
barrayvector-meat.hpp, 256
BARRY_CHECK_SUPPORT

barry-configuration.hpp, 257
BARRY_DEBUG_LEVEL
barry-debug.hpp, 258
BARRY_HPP
barry.hpp, 261
BARRY_ISFINITE
barry-configuration.hpp, 257
BARRY_MAX_NUM_ELEMENTS
barry-configuration.hpp, 257
BARRY_ONE
barry-macros.hpp, 259
BARRY_ONE_DENSE
barry-macros.hpp, 259
BARRY_PROGRESS_BAR_WIDTH
progress.hpp, 315
BARRY_SAFE_EXP
barry-configuration.hpp, 257
BARRY_SUPPORT_MEAT_HPP
support-meat.hpp, 328
BARRY_UNUSED
barry-macros.hpp, 259
BARRY_VERSION
barry.hpp, 261
BARRY_ZERO
barry-macros.hpp, 259
BARRY_ZERO_DENSE
barry-macros.hpp, 259
BARRY_ZERO_NETWORK
network.hpp, 288
BARRY_ZERO_NETWORK_DENSE
network.hpp, 289
BDENSE_TEMPLATE
barraydense-meat-operators.hpp, 222–224
barraydense-meat.hpp, 227, 229–236
BDENSE_TEMPLATE_ARGS
barraydense-meat-operators.hpp, 222, 224
barraydense-meat.hpp, 227
BDENSE_TYPE
barraydense-meat-operators.hpp, 222, 224
barraydense-meat.hpp, 228
begin
BArrayDenseCol< Cell_Type, Data_Type >, 72
BArrayDenseCol_const< Cell_Type, Data_Type >, 74
BArrayDenseRow< Cell_Type, Data_Type >, 76
BArrayDenseRow_const< Cell_Type, Data_Type >, 79
BArrayVector< Cell_Type, Data_Type >, 86
BArrayVector_const< Cell_Type, Data_Type >, 90
PhyloCounterData, 159
PowerSet< Array_Type, Data_Rule_Type >, 166
blengths
NodeData, 157
BOTH
CHECK, 30
EXISTS, 31
BROW_TEMPLATE
barrayrow-meat.hpp, 252–254
BROW_TEMPLATE_ARGS
barrayrow-meat.hpp, 253
BROW_TYPE
barrayrow-meat.hpp, 253

calc
PowerSet< Array_Type, Data_Rule_Type >, 166
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 185
calc_backend_dense
support-meat.hpp, 329
calc_backend_sparse
support-meat.hpp, 330
calc_reduced_sequence
Geese, 125
calc_sequence
Geese, 125
Cell
Cell< Cell_Type >, 93, 94
Cell< Cell_Type >, 92
~Cell, 93
active, 96
add, 94, 95
Cell, 93, 94
operator Cell_Type, 95
operator!=, 95
operator=, 95, 96
operator==, 96
value, 96
visited, 96
Cell_const< Cell_Type >, 97
change_stats
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 188
change_stats_different
support-meat.hpp, 335
CHECK, 30
BOTH, 30
NONE, 30
ONE, 30
TWO, 30
check_bounds
barray-meat.hpp, 212
barraydense-meat.hpp, 240
check_exists
barray-meat.hpp, 212
barraydense-meat.hpp, 240
clear
BArray< Cell_Type, Data_Type >, 37
BArrayDense< Cell_Type, Data_Type >, 56
Counters< Array_Type, Data_Type >, 106
FreqTable< T >, 119
Rules< Array_Type, Data_Type >, 175
statscounter-meat.hpp, 321
COL
barray-meat-operators.hpp, 197
barray-meat.hpp, 203, 207

barraydense-meat-operators.hpp, 223
barraydense-meat.hpp, 228
col
BArray< Cell_Type, Data_Type >, 37
BArrayDense< Cell_Type, Data_Type >, 56, 57
barraydense-meat.hpp, 240
col0
barray-meat.hpp, 213
Col_type
typedefs.hpp, 340
colnames
Flock, 113
Geese, 125
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 139
colsum
BArrayDense< Cell_Type, Data_Type >, 57
conditional_prob
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 139
const
barray-meat.hpp, 213
barraydense-meat.hpp, 241
ConstBArrayRowIter
ConstBArrayRowIter< Cell_Type, Data_Type >, 98
ConstBArrayRowIter< Cell_Type, Data_Type >, 97
~ConstBArrayRowIter, 98
Array, 98
ConstBArrayRowIter, 98
current_col, 99
current_row, 99
iter, 99
coord_i
support-meat.hpp, 335
coord_j
support-meat.hpp, 335
coordiantes_n_free
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 188
coordiantes_n_locked
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 188
coordinates_free
PowerSet< Array_Type, Data_Rule_Type >, 168
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 188
coordinates_locked
PowerSet< Array_Type, Data_Rule_Type >, 168
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 189
copy_data
barray-meat.hpp, 213
barraydense-meat.hpp, 241
count
Counter< Array_Type, Data_Type >, 102
count_all
StatsCounter< Array_Type, Data_Type >, 179
count_current
StatsCounter< Array_Type, Data_Type >, 179
count_fun
Counter< Array_Type, Data_Type >, 103
counters-meat.hpp, 269
count_fun_
counters-meat.hpp, 274
count_init
StatsCounter< Array_Type, Data_Type >, 179
Counter
Counter< Array_Type, Data_Type >, 100, 101
counter
counters-meat.hpp, 275
statscounter-meat.hpp, 323
Counter< Array_Type, Data_Type >, 99
~Counter, 101
count, 102
count_fun, 103
Counter, 100, 101
data, 103
delete_data, 103
desc, 103
get_description, 102
get_name, 102
init, 102
init_fun, 103
name, 104
operator=, 102
counter_
counters-meat.hpp, 275
counter_absdiff
Network counters, 13
counter_co_opt
Phylo counters, 20
counter_cogain
Phylo counters, 21
counter_css_census01
network-css.hpp, 281
counter_css_census02
network-css.hpp, 282
counter_css_census03
network-css.hpp, 282
counter_css_census04
network-css.hpp, 282
counter_css_census05
network-css.hpp, 282
counter_css_census06
network-css.hpp, 283
counter_css_census07
network-css.hpp, 283
counter_css_census08
network-css.hpp, 283
counter_css_census09

network-css.hpp, 283
counter_css_census10
network-css.hpp, 284
counter_css_completely_false_recip_comiss
network-css.hpp, 284
counter_css_completely_false_recip_omiss
network-css.hpp, 284
counter_css_mixed_recip
network-css.hpp, 284
counter_css_partially_false_recip_commi
network-css.hpp, 285
counter_css_partially_false_recip_omiss
network-css.hpp, 285
counter_ctriads
Network counters, 13, 14
counter_degree
Network counters, 14
counter_deleted
statscounter-meat.hpp, 323
counter_density
Network counters, 14
counter_diff
Network counters, 14
counter_edges
Network counters, 14
Counter_fun_type
typedefs.hpp, 340
COUNTER_FUNCTION
barry.hpp, 261
counter_gains
Phylo counters, 21
counter_gains_from_0
Phylo counters, 21
counter_gains_k_offspring
Phylo counters, 22
counter_genes_changing
Phylo counters, 22
counter_idegree
Network counters, 15
counter_idegree15
Network counters, 15
counter_isolates
Network counters, 15, 16
counter_istar2
Network counters, 16
counter_k_genes_changing
Phylo counters, 22
COUNTER_LAMBDA
barry.hpp, 262
counter_less_than_p_prop_genes_changing
Phylo counters, 22
counter_longest
Phylo counters, 23
counter_loss
Phylo counters, 23
counter_maxfuns
Phylo counters, 23
counter_mutual
Network counters, 16
counter_neofun
Phylo counters, 23
counter_neofun_a2b
Phylo counters, 24
counter_nodecov
Network counters, 16
counter_nodeicov
Network counters, 17
counter_nodematch
Network counters, 17
counter_nodeocov
Network counters, 17
counter_odegree
Network counters, 17
counter_odegree15
Network counters, 18
counter_ostar2
Network counters, 18
counter_overall_changes
Phylo counters, 24
counter_overall_gains
Phylo counters, 24
counter_overall_gains_from_0
Phylo counters, 24
counter_overall_loss
Phylo counters, 25
counter_pairwise_first_gain
Phylo counters, 25
counter_pairwise_overall_change
Phylo counters, 25
counter_pairwise_preserving
Phylo counters, 25
counter_preserve_pseudogene
Phylo counters, 26
counter_prop_genes_changing
Phylo counters, 26
counter_subfun
Phylo counters, 26
COUNTER_TEMPLATE
counters-meat.hpp, 268–270
COUNTER_TEMPLATE_ARGS
counters-meat.hpp, 268
counter_ttriads
Network counters, 18, 19
COUNTER_TYPE
counters-meat.hpp, 268
Counters
Counters< Array_Type, Data_Type >, 105
counters
statscounter-meat.hpp, 323
support-meat.hpp, 335
Counters< Array_Type, Data_Type >, 104
∼Counters, 105
add_counter, 106
clear, 106
Counters, 105
get_descriptions, 107

get_names, 107
operator=, 107
operator[], 108
size, 108
counters-meat.hpp
count_fun, 269
count_fun_, 274
counter, 275
counter_, 275
COUNTER_TEMPLATE, 268–270
COUNTER_TEMPLATE_ARGS, 268
COUNTER_TYPE, 268
COUNTERS_TEMPLATE, 268, 270–272
COUNTERS_TEMPLATE_ARGS, 268
COUNTERS_TYPE, 268
data, 272
data_, 275
delete_data, 272
delete_data_, 275
delete_to_be_deleted, 272, 273
desc, 273
desc_, 276
i, 276
init_fun, 273
init_fun_, 276
j, 276
name, 274
name_, 276
noexcept, 277
push_back, 274
return, 277
to_be_deleted, 274
counters_
statscounter-meat.hpp, 323
support-meat.hpp, 335
COUNTERS_TEMPLATE
counters-meat.hpp, 268, 270–272
COUNTERS_TEMPLATE_ARGS
counters-meat.hpp, 268
COUNTERS_TYPE
counters-meat.hpp, 268
Counting, 11
counts
PhyloRuleDynData, 162
Counts_type
typedefs.hpp, 340
CSS_APPEND
network-css.hpp, 279
CSS_CASE_ELSE
network-css.hpp, 279
CSS_CASE_PERCEIVED
network-css.hpp, 280
CSS_CASE_TRUTH
network-css.hpp, 280
CSS_CHECK_SIZE
network-css.hpp, 280
CSS_CHECK_SIZE_INIT
network-css.hpp, 280
CSS_NET_COUNTER_LAMBDA_INIT
network-css.hpp, 280
CSS_PERCEIVED_CELLS
network-css.hpp, 281
CSS_SIZE
network-css.hpp, 281
CSS_TRUE_CELLS
network-css.hpp, 281
current_col
ConstBArrayRowIter< Cell_Type, Data_Type >, 99
current_row
ConstBArrayRowIter< Cell_Type, Data_Type >, 99
current_stats
statscounter-meat.hpp, 323
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 189

D
BArray< Cell_Type, Data_Type >, 38
BArrayDense< Cell_Type, Data_Type >, 57
Rule< Array_Type, Data_Type >, 173
dat
Flock, 117
data
barray-meat.hpp, 213
barraydense-meat.hpp, 241
Counter< Array_Type, Data_Type >, 103
counters-meat.hpp, 272
PowerSet< Array_Type, Data_Rule_Type >, 168
data_
counters-meat.hpp, 275
DEFAULT_DUPLICATION
phylo.hpp, 295
default_val
BArray< Cell_Type, Data_Type >, 38
BArrayDense< Cell_Type, Data_Type >, 57
delete_counters
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 189
support-meat.hpp, 335
delete_data
barray-meat.hpp, 213
barraydense-meat.hpp, 241
Counter< Array_Type, Data_Type >, 103
counters-meat.hpp, 272
delete_data_
barray-meat.hpp, 214
barraydense-meat.hpp, 241
counters-meat.hpp, 275
delete_rengine
Geese, 131
delete_rules
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 189
support-meat.hpp, 336
delete_rules_dyn

Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 189
support-meat.hpp, 336
delete_support
Geese, 132
delete_to_be_deleted
counters-meat.hpp, 272, 273
desc
Counter< Array_Type, Data_Type >, 103
counters-meat.hpp, 273
desc_
counters-meat.hpp, 276
directed
NetworkData, 150
DUPL_DUPL
phylo.hpp, 295
DUPL_EITH
phylo.hpp, 295
DUPL_SPEC
phylo.hpp, 295
duplication
Node, 155
NodeData, 158
PhyloRuleDynData, 162

el
barraydense-meat.hpp, 242
el_colsums
barraydense-meat.hpp, 242
el_rowsums
barraydense-meat.hpp, 242
else
barray-meat.hpp, 214
barraydense-meat.hpp, 242
support-meat.hpp, 336
empty
PhyloCounterData, 159
EmptyArray
PowerSet< Array_Type, Data_Rule_Type >, 168
statscounter-meat.hpp, 324
end
BArrayDenseCol< Cell_Type, Data_Type >, 72
BArrayDenseCol_const< Cell_Type, Data_Type >, 74
BArrayDenseRow< Cell_Type, Data_Type >, 76
BArrayDenseRow_const< Cell_Type, Data_Type >, 79
BArrayVector< Cell_Type, Data_Type >, 86
BArrayVector_const< Cell_Type, Data_Type >, 90
PhyloCounterData, 159
PowerSet< Array_Type, Data_Rule_Type >, 166
Progress, 171
Entries
Entries< Cell_Type >, 109
Entries< Cell_Type >, 109
~Entries, 110
Entries, 109
resize, 110
source, 110
target, 110
val, 110
eval_rules_dyn
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 185
EXISTS, 31
AS_ONE, 31
AS_ZERO, 31
BOTH, 31
NONE, 32
ONE, 32
TWO, 32
UKNOWN, 32

f_
statscounter-meat.hpp, 324
support-meat.hpp, 336
false
barray-meat.hpp, 214
barraydense-meat.hpp, 242
first
barray-meat.hpp, 214
Flock, 111
~Flock, 112
add_data, 112
colnames, 113
dat, 117
Flock, 112
get_counters, 113
get_model, 113
get_stats_support, 113
get_stats_target, 113
get_support_fun, 114
init, 114
initialized, 117
likelihood_joint, 114
model, 117
nfunctions, 117
nfuns, 114
nleafs, 115
nnodes, 115
nterms, 115
ntrees, 115
operator(), 115
parse_polytomies, 116
print, 116
rengine, 117
set_seed, 116
support_size, 116
flush_data
BArray< Cell_Type, Data_Type >, 38
for
barray-meat-operators.hpp, 198
barray-meat.hpp, 207, 208
barraydense-meat.hpp, 236
statscounter-meat.hpp, 321
support-meat.hpp, 330

FreqTable
FreqTable< T >, 118
FreqTable< T >, 118
~FreqTable, 119
add, 119
as_vector, 119
clear, 119
FreqTable, 118
get_data, 119
get_index, 120
make_hash, 120
print, 120
reserve, 120
size, 120

Geese, 121
~Geese, 124
calc_reduced_sequence, 125
calc_sequence, 125
colnames, 125
delete_rengine, 131
delete_support, 132
Geese, 124
get_annotated_nodes, 125
get_counters, 125
get_model, 125
get_probabilities, 126
get_rengine, 126
get_states, 126
get_support_fun, 126
inherit_support, 126
init, 127
init_node, 127
initialized, 132
likelihood, 127
likelihood_exhaust, 127
map_to_nodes, 132
nannotations, 127
nfunctions, 132
nfuns, 128
nleafs, 128
nnodes, 128
nodes, 132
nterms, 128
observed_counts, 128
operator=, 129
parse_polytomies, 129
predict, 129
predict_backend, 129
predict_exhaust, 130
predict_exhaust_backend, 130
predict_sim, 130
print, 130
print_observed_counts, 130
pset_loc, 132
reduced_sequence, 133
sequence, 133
set_seed, 131
simulate, 131
support_size, 131
update_annotations, 131
geese-bones.hpp
INITIALIZED, 306
keygen_full, 307
RULE_FUNCTION, 307
vec_diff, 307
vector_caster, 307
gen_key
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 140
get_annotated_nodes
Geese, 125
get_arrays2support
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 140
get_cell
BArray< Cell_Type, Data_Type >, 38
BArrayDense< Cell_Type, Data_Type >, 57
get_col_vec
BArray< Cell_Type, Data_Type >, 38, 39
BArrayDense< Cell_Type, Data_Type >, 58
get_counters
Flock, 113
Geese, 125
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 140
PhyloCounterData, 160
StatsCounter< Array_Type, Data_Type >, 179
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 185
get_counts
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 185
get_current_stats
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 186
get_data
BArrayDense< Cell_Type, Data_Type >, 58
FreqTable< T >, 119
PowerSet< Array_Type, Data_Rule_Type >, 167
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 186
get_data_ptr
PowerSet< Array_Type, Data_Rule_Type >, 167
get_description
Counter< Array_Type, Data_Type >, 102
get_descriptions
Counters< Array_Type, Data_Type >, 107
StatsCounter< Array_Type, Data_Type >, 180
get_entries

BArray< Cell_Type, Data_Type >, 39
BArrayDense< Cell_Type, Data_Type >, 58
get_index
FreqTable< T >, 120
get_last_name
phylo.hpp, 300
get_model
Flock, 113
Geese, 125
get_name
Counter< Array_Type, Data_Type >, 102
get_names
Counters< Array_Type, Data_Type >, 107
StatsCounter< Array_Type, Data_Type >, 180
get_norm_const
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 140
get_parent
Node, 153
get_probabilities
Geese, 126
get_pset
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 140
get_pset_arrays
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 141
get_pset_probs
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 141
get_pset_stats
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 141
get_rengine
Geese, 126
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 141
get_row_vec
BArray< Cell_Type, Data_Type >, 39
BArrayDense< Cell_Type, Data_Type >, 58, 59
get_rules
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 142
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 186
get_rules_dyn
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 142
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 186
get_seq
Rules< Array_Type, Data_Type >, 175
get_states
Geese, 126
get_stats_support
Flock, 113
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 142
get_stats_target
Flock, 113
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 142
get_support_fun
Flock, 114
Geese, 126
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 142

hashes
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 190
support-meat.hpp, 336
hashes_initialized
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 190

i
counters-meat.hpp, 276
i1
barray-meat.hpp, 214
barraydense-meat.hpp, 243
id
Node, 155
if
barray-meat.hpp, 208–210
barraydense-meat.hpp, 236
support-meat.hpp, 330
IF_MATCHES
phylo.hpp, 296
IF_NOTMATCHES
phylo.hpp, 296
include/barry/barray-bones.hpp, 193
include/barry/barray-iterator.hpp, 194
include/barry/barray-meat-operators.hpp, 195
include/barry/barray-meat.hpp, 200
include/barry/barraycell-bones.hpp, 218
include/barry/barraycell-meat.hpp, 218
include/barry/barraydense-bones.hpp, 219
include/barry/barraydense-meat-operators.hpp, 221
include/barry/barraydense-meat.hpp, 225
include/barry/barraydensecell-bones.hpp, 246
include/barry/barraydensecell-meat.hpp, 247

include/barry/barraydensecol-bones.hpp, 248
include/barry/barraydenserow-bones.hpp, 249
include/barry/barrayrow-bones.hpp, 251
include/barry/barrayrow-meat.hpp, 251
include/barry/barrayvector-bones.hpp, 254
include/barry/barrayvector-meat.hpp, 255
include/barry/barry-configuration.hpp, 256
include/barry/barry-debug.hpp, 258
include/barry/barry-macros.hpp, 258
include/barry/barry.hpp, 260
include/barry/cell-bones.hpp, 263
include/barry/cell-meat.hpp, 263
include/barry/col-bones.hpp, 264
include/barry/counters-bones.hpp, 264
include/barry/counters-meat.hpp, 266
include/barry/counters/network-css.hpp, 278
include/barry/counters/network.hpp, 286
include/barry/counters/phylo.hpp, 293
include/barry/model-bones.hpp, 300
include/barry/model-meat.hpp, 302
include/barry/models/geese.hpp, 304
include/barry/models/geese/flock-bones.hpp, 305
include/barry/models/geese/flock-meat.hpp, 305
include/barry/models/geese/geese-bones.hpp, 306
include/barry/models/geese/geese-meat-constructors.hpp, 308
include/barry/models/geese/geese-meat-likelihood.hpp, 308
include/barry/models/geese/geese-meat-likelihood_exhaust.hpp, 309
include/barry/models/geese/geese-meat-predict.hpp, 310
include/barry/models/geese/geese-meat-predict_exhaust.hpp, 310
include/barry/models/geese/geese-meat-predict_sim.hpp, 311
include/barry/models/geese/geese-meat-simulate.hpp, 311
include/barry/models/geese/geese-meat.hpp, 312
include/barry/models/geese/geese-node-bones.hpp, 312
include/barry/powerset-bones.hpp, 313
include/barry/powerset-meat.hpp, 314
include/barry/progress.hpp, 315
include/barry/rules-bones.hpp, 315
include/barry/rules-meat.hpp, 317
include/barry/statscounter-bones.hpp, 317
include/barry/statscounter-meat.hpp, 319
include/barry/statsdb.hpp, 325
include/barry/support-bones.hpp, 325
include/barry/support-meat.hpp, 327
include/barry/typedefs.hpp, 338
indices
    NetCounterData, 148
inherit_support
    Geese, 126
init
    Counter< Array_Type, Data_Type >, 102
    Flock, 114
    Geese, 127
init_fun
    Counter< Array_Type, Data_Type >, 103
    counters-meat.hpp, 273
init_fun_
    counters-meat.hpp, 276
init_node
    Geese, 127
init_support
    PowerSet< Array_Type, Data_Rule_Type >, 167
    Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 186
INITIALIZED
    geese-bones.hpp, 306
initialized
    Flock, 117
    Geese, 132
insert_cell
    BArray< Cell_Type, Data_Type >, 39, 40
    BArrayDense< Cell_Type, Data_Type >, 59
    barraydense-meat.hpp, 237
    support-meat.hpp, 331
is_col
    BArrayVector< Cell_Type, Data_Type >, 86
    BArrayVector_const< Cell_Type, Data_Type >, 90
is_dense
    BArray< Cell_Type, Data_Type >, 40
    BArrayDense< Cell_Type, Data_Type >, 59
IS_DUPLICATION
    phylo.hpp, 296
IS_EITHER
    phylo.hpp, 296
is_empty
    BArray< Cell_Type, Data_Type >, 40
    BArrayDense< Cell_Type, Data_Type >, 59
is_leaf
    Node, 154
is_row
    BArrayVector< Cell_Type, Data_Type >, 86
    BArrayVector_const< Cell_Type, Data_Type >, 90
IS_SPECIATION
    phylo.hpp, 296
iter
    ConstBArrayRowIter< Cell_Type, Data_Type >, 99

j
    barray-meat.hpp, 215
    barraydense-meat.hpp, 243
    counters-meat.hpp, 276
    statscounter-meat.hpp, 324
j0
    barray-meat.hpp, 215
    barraydense-meat.hpp, 243
j1
    barray-meat.hpp, 215
    barraydense-meat.hpp, 243

keygen_default
model-bones.hpp, 301
keygen_full
geese-bones.hpp, 307

lb
PhyloRuleDynData, 162
likelihood
Geese, 127
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 143
likelihood_
model-meat.hpp, 303
likelihood_exhaust
Geese, 127
likelihood_joint
Flock, 114
likelihood_total
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 143

M
barray-meat.hpp, 211, 215
barraydense-meat.hpp, 237, 243
PowerSet< Array_Type, Data_Rule_Type >, 169
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 190
M_
barray-meat.hpp, 215
barraydense-meat.hpp, 243
MAKE_DUPL_VARS
phylo.hpp, 297
make_hash
FreqTable< T >, 120
Map
barry-configuration.hpp, 257
map_to_nodes
Geese, 132
MapVec_type
typedefs.hpp, 341
max_num_elements
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 190
Model
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 136
model
Flock, 117
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 133
~Model, 137
add_array, 137
add_counter, 137, 138
add_rule, 138
add_rule_dyn, 138, 139
colnames, 139
conditional_prob, 139
gen_key, 140
get_arrays2support, 140
get_counters, 140
get_norm_const, 140
get_pset, 140
get_pset_arrays, 141
get_pset_probs, 141
get_pset_stats, 141
get_rengine, 141
get_rules, 142
get_rules_dyn, 142
get_stats_support, 142
get_stats_target, 142
get_support_fun, 142
likelihood, 143
likelihood_total, 143
Model, 136
nterms, 144
operator=, 144
print, 144
print_stats, 144
sample, 144, 145
set_counters, 145
set_keygen, 145
set_rengine, 145
set_rules, 145
set_rules_dyn, 146
set_seed, 146
size, 146
size_unique, 146
store_psets, 146
support_size, 147
model-bones.hpp
keygen_default, 301
model-meat.hpp
likelihood_, 303
MODEL_TEMPLATE, 302, 303
MODEL_TEMPLATE_ARGS, 302
MODEL_TYPE, 303
update_normalizing_constant, 304
MODEL_TEMPLATE
model-meat.hpp, 302, 303
MODEL_TEMPLATE_ARGS
model-meat.hpp, 302
MODEL_TYPE
model-meat.hpp, 303

N
barray-meat.hpp, 216
barraydense-meat.hpp, 244
PowerSet< Array_Type, Data_Rule_Type >, 169
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 190
n_counters

Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 191
n_free
PowerSet< Array_Type, Data_Rule_Type >, 169
n_locked
PowerSet< Array_Type, Data_Rule_Type >, 169
name
Counter< Array_Type, Data_Type >, 104
counters-meat.hpp, 274
name_
counters-meat.hpp, 276
nannotations
Geese, 127
narray
Node, 155
NCells
barray-meat.hpp, 216
ncol
BArray< Cell_Type, Data_Type >, 40
BArrayDense< Cell_Type, Data_Type >, 60
NET_C_DATA_IDX
network.hpp, 289
NET_C_DATA_NUM
network.hpp, 289
NetCounter
network.hpp, 291
NetCounterData, 147
∼NetCounterData, 148
indices, 148
NetCounterData, 147, 148
numbers, 148
NetCounters
network.hpp, 291
NetModel
network.hpp, 291
NetRule
network.hpp, 291
NetRules
network.hpp, 291
NetStatsCounter
network.hpp, 291
NetSupport
network.hpp, 292
Network
network.hpp, 292
Network counters, 12
counter_absdiff, 13
counter_ctriads, 13, 14
counter_degree, 14
counter_density, 14
counter_diff, 14
counter_edges, 14
counter_idegree, 15
counter_idegree15, 15
counter_isolates, 15, 16
counter_istar2, 16
counter_mutual, 16
counter_nodecov, 16
counter_nodeicov, 17
counter_nodematch, 17
counter_nodeocov, 17
counter_odegree, 17
counter_odegree15, 18
counter_ostar2, 18
counter_ttriads, 18, 19
NETWORK_COUNTER, 19
network-css.hpp
counter_css_census01, 281
counter_css_census02, 282
counter_css_census03, 282
counter_css_census04, 282
counter_css_census05, 282
counter_css_census06, 283
counter_css_census07, 283
counter_css_census08, 283
counter_css_census09, 283
counter_css_census10, 284
counter_css_completely_false_recip_comiss, 284
counter_css_completely_false_recip_omiss, 284
counter_css_mixed_recip, 284
counter_css_partially_false_recip_commi, 285
counter_css_partially_false_recip_omiss, 285
CSS_APPEND, 279
CSS_CASE_ELSE, 279
CSS_CASE_PERCEIVED, 280
CSS_CASE_TRUTH, 280
CSS_CHECK_SIZE, 280
CSS_CHECK_SIZE_INIT, 280
CSS_NET_COUNTER_LAMBDA_INIT, 280
CSS_PERCEIVED_CELLS, 281
CSS_SIZE, 281
CSS_TRUE_CELLS, 281
network.hpp
BARRY_ZERO_NETWORK, 288
BARRY_ZERO_NETWORK_DENSE, 289
NET_C_DATA_IDX, 289
NET_C_DATA_NUM, 289
NetCounter, 291
NetCounters, 291
NetModel, 291
NetRule, 291
NetRules, 291
NetStatsCounter, 291
NetSupport, 292
Network, 292
NETWORK_COUNTER, 289
NETWORK_COUNTER_LAMBDA, 289
NETWORK_RULE, 290
NETWORK_RULE_LAMBDA, 290
NetworkDense, 292
NETWORKDENSE_COUNTER_LAMBDA, 290
rules_zerodiag, 292
NETWORK_COUNTER
Network counters, 19
network.hpp, 289

NETWORK_COUNTER_LAMBDA
network.hpp, 289
NETWORK_RULE
network.hpp, 290
NETWORK_RULE_LAMBDA
network.hpp, 290
NetworkData, 149
~NetworkData, 150
directed, 150
NetworkData, 149, 150
vertex_attr, 150
NetworkDense
network.hpp, 292
NETWORKDENSE_COUNTER_LAMBDA
network.hpp, 290
next
Progress, 171
nfunctions
Flock, 117
Geese, 132
nfuns
Flock, 114
Geese, 128
nleafs
Flock, 115
Geese, 128
nnodes
Flock, 115
Geese, 128
nnozero
BArray< Cell_Type, Data_Type >, 41
BArrayDense< Cell_Type, Data_Type >, 60
Node, 151
~Node, 153
annotations, 154
array, 154
arrays, 154
duplication, 155
get_parent, 153
id, 155
is_leaf, 154
narray, 155
Node, 152, 153
noffspring, 154
offspring, 155
ord, 155
parent, 156
probability, 156
subtree_prob, 156
visited, 156
NodeData, 157
blengths, 157
duplication, 158
NodeData, 157
states, 158
nodes
Geese, 132
noexcept
counters-meat.hpp, 277
noffspring
Node, 154
NONE
CHECK, 30
EXISTS, 32
nrow
BArray< Cell_Type, Data_Type >, 41
BArrayDense< Cell_Type, Data_Type >, 60
nterms
Flock, 115
Geese, 128
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 144
ntrees
Flock, 115
numbers
NetCounterData, 148

observed_counts
Geese, 128
offspring
Node, 155
ONE
CHECK, 30
EXISTS, 32
operator BArrayRow< Cell_Type, Data_Type >
BArrayRow< Cell_Type, Data_Type >, 81
operator BArrayRow_const< Cell_Type, Data_Type >
BArrayRow_const< Cell_Type, Data_Type >, 83
operator Cell_Type
BArrayCell< Cell_Type, Data_Type >, 48
BArrayCell_const< Cell_Type, Data_Type >, 51
BArrayDenseCell< Cell_Type, Data_Type >, 68
Cell< Cell_Type >, 95
operator std::vector< Cell_Type >
BArrayVector< Cell_Type, Data_Type >, 87
BArrayVector_const< Cell_Type, Data_Type >, 90
operator!=
BArrayCell_const< Cell_Type, Data_Type >, 51
BArrayRow_const< Cell_Type, Data_Type >, 83
BArrayVector_const< Cell_Type, Data_Type >, 90
Cell< Cell_Type >, 95
operator<
BArrayCell_const< Cell_Type, Data_Type >, 51
BArrayRow_const< Cell_Type, Data_Type >, 83
BArrayVector_const< Cell_Type, Data_Type >, 91
operator<=
BArrayCell_const< Cell_Type, Data_Type >, 51
BArrayRow_const< Cell_Type, Data_Type >, 84
BArrayVector_const< Cell_Type, Data_Type >, 91
operator>
BArrayCell_const< Cell_Type, Data_Type >, 51
BArrayRow_const< Cell_Type, Data_Type >, 84
BArrayVector_const< Cell_Type, Data_Type >, 91
operator>=
BArrayCell_const< Cell_Type, Data_Type >, 52
BArrayRow_const< Cell_Type, Data_Type >, 84

BArrayVector_const< Cell_Type, Data_Type >, 91
operator*=
BArray< Cell_Type, Data_Type >, 41
BArrayCell< Cell_Type, Data_Type >, 48
BArrayDense< Cell_Type, Data_Type >, 60
BArrayDenseCell< Cell_Type, Data_Type >, 68
BArrayRow< Cell_Type, Data_Type >, 81
BArrayVector< Cell_Type, Data_Type >, 87
operator()
BArray< Cell_Type, Data_Type >, 41
barray-meat-operators.hpp, 199
BArrayDense< Cell_Type, Data_Type >, 60
BArrayDenseCol< Cell_Type, Data_Type >, 72
BArrayDenseCol_const< Cell_Type, Data_Type >, 74
BArrayDenseRow< Cell_Type, Data_Type >, 77
BArrayDenseRow_const< Cell_Type, Data_Type >, 79
Flock, 115
PhyloCounterData, 160
Rule< Array_Type, Data_Type >, 173
Rules< Array_Type, Data_Type >, 176
vecHasher< T >, 191
operator+=
BArray< Cell_Type, Data_Type >, 41, 42
BArrayCell< Cell_Type, Data_Type >, 48
BArrayDense< Cell_Type, Data_Type >, 61
BArrayDenseCell< Cell_Type, Data_Type >, 69
BArrayRow< Cell_Type, Data_Type >, 81
BArrayVector< Cell_Type, Data_Type >, 87
operator-=
BArray< Cell_Type, Data_Type >, 42
BArrayCell< Cell_Type, Data_Type >, 49
BArrayDense< Cell_Type, Data_Type >, 61
BArrayDenseCell< Cell_Type, Data_Type >, 69
BArrayRow< Cell_Type, Data_Type >, 81
BArrayVector< Cell_Type, Data_Type >, 87
operator/=
BArray< Cell_Type, Data_Type >, 42
BArrayCell< Cell_Type, Data_Type >, 49
BArrayDense< Cell_Type, Data_Type >, 62
BArrayDenseCell< Cell_Type, Data_Type >, 69
BArrayRow< Cell_Type, Data_Type >, 82
BArrayVector< Cell_Type, Data_Type >, 87
operator=
BArray< Cell_Type, Data_Type >, 43
BArrayCell< Cell_Type, Data_Type >, 49
BArrayDense< Cell_Type, Data_Type >, 62
BArrayDenseCell< Cell_Type, Data_Type >, 69
BArrayRow< Cell_Type, Data_Type >, 82
BArrayVector< Cell_Type, Data_Type >, 88
Cell< Cell_Type >, 95, 96
Counter< Array_Type, Data_Type >, 102
Counters< Array_Type, Data_Type >, 107
Geese, 129
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 144
Rules< Array_Type, Data_Type >, 176
operator==
BArray< Cell_Type, Data_Type >, 43
BArrayCell< Cell_Type, Data_Type >, 49
BArrayCell_const< Cell_Type, Data_Type >, 51
BArrayDense< Cell_Type, Data_Type >, 62
BArrayDenseCell< Cell_Type, Data_Type >, 69
BArrayRow< Cell_Type, Data_Type >, 82
BArrayRow_const< Cell_Type, Data_Type >, 84
BArrayVector< Cell_Type, Data_Type >, 88
BArrayVector_const< Cell_Type, Data_Type >, 91
Cell< Cell_Type >, 96
operator[]
Counters< Array_Type, Data_Type >, 108
PhyloCounterData, 160
PowerSet< Array_Type, Data_Rule_Type >, 167
ord
Node, 155
out_of_range
BArray< Cell_Type, Data_Type >, 43
BArrayDense< Cell_Type, Data_Type >, 62

parent
Node, 156
parse_polytomies
Flock, 116
Geese, 129
Phylo counters, 19
counter_co_opt, 20
counter_cogain, 21
counter_gains, 21
counter_gains_from_0, 21
counter_gains_k_offspring, 22
counter_genes_changing, 22
counter_k_genes_changing, 22
counter_less_than_p_prop_genes_changing, 22
counter_longest, 23
counter_loss, 23
counter_maxfuns, 23
counter_neofun, 23
counter_neofun_a2b, 24
counter_overall_changes, 24
counter_overall_gains, 24
counter_overall_gains_from_0, 24
counter_overall_loss, 25
counter_pairwise_first_gain, 25
counter_pairwise_overall_change, 25
counter_pairwise_preserving, 25
counter_preserve_pseudogene, 26
counter_prop_genes_changing, 26
counter_subfun, 26
Phylo rules, 27
rule_dyn_limit_changes, 27
phylo.hpp
DEFAULT_DUPLICATION, 295
DUPL_DUPL, 295
DUPL_EITH, 295
DUPL_SPEC, 295
get_last_name, 300

IF_MATCHES, 296
IF_NOTMATCHES, 296
IS_DUPLICATION, 296
IS_EITHER, 296
IS_SPECIATION, 296
MAKE_DUPL_VARS, 297
PHYLO_CHECK_MISSING, 297
PHYLO_COUNTER_LAMBDA, 297
PHYLO_RULE_DYN_LAMBDA, 297
PhyloArray, 298
PhyloCounter, 298
PhyloCounters, 298
PhyloModel, 298
PhyloPowerSet, 298
PhyloRule, 299
PhyloRuleData, 299
PhyloRuleDyn, 299
PhyloRules, 299
PhyloRulesDyn, 299
PhyloStatsCounter, 299
PhyloSupport, 300
PHYLO_CHECK_MISSING
phylo.hpp, 297
PHYLO_COUNTER_LAMBDA
phylo.hpp, 297
PHYLO_RULE_DYN_LAMBDA
phylo.hpp, 297
PhyloArray
phylo.hpp, 298
PhyloCounter
phylo.hpp, 298
PhyloCounterData, 158
at, 159
begin, 159
empty, 159
end, 159
get_counters, 160
operator(), 160
operator[], 160
PhyloCounterData, 159
push_back, 160
reserve, 160
shrink_to_fit, 160
size, 161
PhyloCounters
phylo.hpp, 298
PhyloModel
phylo.hpp, 298
PhyloPowerSet
phylo.hpp, 298
PhyloRule
phylo.hpp, 299
PhyloRuleData
phylo.hpp, 299
PhyloRuleDyn
phylo.hpp, 299
PhyloRuleDynData, 161
∼PhyloRuleDynData, 162
counts, 162
duplication, 162
lb, 162
PhyloRuleDynData, 161
pos, 162
ub, 163
PhyloRules
phylo.hpp, 299
PhyloRulesDyn
phylo.hpp, 299
PhyloStatsCounter
phylo.hpp, 299
PhyloSupport
phylo.hpp, 300
POS
barraydense-meat-operators.hpp, 223
barraydense-meat.hpp, 228
barraydensecell-bones.hpp, 246
barraydensecell-meat.hpp, 248
barraydensecol-bones.hpp, 249
barraydenserow-bones.hpp, 250
pos
PhyloRuleDynData, 162
POS_N
barraydense-meat-operators.hpp, 223
barraydense-meat.hpp, 228
barraydensecol-bones.hpp, 249
barraydenserow-bones.hpp, 250
PowerSet
PowerSet< Array_Type, Data_Rule_Type >, 165
PowerSet< Array_Type, Data_Rule_Type >, 163
∼PowerSet, 165
add_rule, 166
begin, 166
calc, 166
coordinates_free, 168
coordinates_locked, 168
data, 168
EmptyArray, 168
end, 166
get_data, 167
get_data_ptr, 167
init_support, 167
M, 169
N, 169
n_free, 169
n_locked, 169
operator[], 167
PowerSet, 165
reset, 167
rules, 169
rules_deleted, 169
size, 168
predict
Geese, 129
predict_backend
Geese, 129
predict_exhaust

Geese, 130
predict_exhaust_backend
Geese, 130
predict_sim
Geese, 130
print
BArray< Cell_Type, Data_Type >, 43
BArrayDense< Cell_Type, Data_Type >, 62
Flock, 116
FreqTable< T >, 120
Geese, 130
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 144
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 187
print_observed_counts
Geese, 130
print_stats
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 144
printf_barry
barry-configuration.hpp, 257
probability
Node, 156
Progress, 170
~Progress, 170
end, 171
next, 171
Progress, 170
progress.hpp
BARRY_PROGRESS_BAR_WIDTH, 315
pset_loc
Geese, 132
push_back
counters-meat.hpp, 274
PhyloCounterData, 160

README.md, 342
reduced_sequence
Geese, 133
rengine
Flock, 117
report
barray-meat.hpp, 216
barraydense-meat.hpp, 244
reserve
BArray< Cell_Type, Data_Type >, 43
BArrayDense< Cell_Type, Data_Type >, 63
FreqTable< T >, 120
PhyloCounterData, 160
reset
PowerSet< Array_Type, Data_Rule_Type >, 167
reset_array
StatsCounter< Array_Type, Data_Type >, 180
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 187
resize
BArray< Cell_Type, Data_Type >, 44
barray-meat.hpp, 211
BArrayDense< Cell_Type, Data_Type >, 63
barraydense-meat.hpp, 237, 238
Entries< Cell_Type >, 110
statscounter-meat.hpp, 321
return
barray-meat.hpp, 211, 216
barraydense-meat.hpp, 244
counters-meat.hpp, 277
statscounter-meat.hpp, 324
support-meat.hpp, 337
rhs
barray-meat-operators.hpp, 199
rm_cell
BArray< Cell_Type, Data_Type >, 44
BArrayDense< Cell_Type, Data_Type >, 63
barraydense-meat.hpp, 238, 239
support-meat.hpp, 331
ROW
barray-meat-operators.hpp, 197
barray-meat.hpp, 203, 211
barraydense-meat-operators.hpp, 223
barraydense-meat.hpp, 228
row
BArray< Cell_Type, Data_Type >, 44
BArrayDense< Cell_Type, Data_Type >, 63
row0
barray-meat.hpp, 216
Row_type
typedefs.hpp, 341
rowsum
BArrayDense< Cell_Type, Data_Type >, 63
Rule
Rule< Array_Type, Data_Type >, 172
Rule< Array_Type, Data_Type >, 171
~Rule, 172
D, 173
operator(), 173
Rule, 172
rule_dyn_limit_changes
Phylo rules, 27
rule_fun_default
rules-bones.hpp, 316
Rule_fun_type
typedefs.hpp, 341
RULE_FUNCTION
barry.hpp, 262
geese-bones.hpp, 307
RULE_LAMBDA
barry.hpp, 262
Rules
Rules< Array_Type, Data_Type >, 174
rules

PowerSet< Array_Type, Data_Rule_Type >, 169
support-meat.hpp, 337
Rules< Array_Type, Data_Type >, 173
~Rules, 174
add_rule, 175
clear, 175
get_seq, 175
operator(), 176
operator=, 176
Rules, 174
size, 176
rules-bones.hpp
rule_fun_default, 316
rules_
support-meat.hpp, 337
rules_deleted
PowerSet< Array_Type, Data_Rule_Type >, 169
rules_dyn
support-meat.hpp, 337
rules_zerodiag
network.hpp, 292

sample
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 144, 145
search
barray-meat.hpp, 217
sequence
Geese, 133
set_counters
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 145
StatsCounter< Array_Type, Data_Type >, 180
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 187
set_data
BArray< Cell_Type, Data_Type >, 44
BArrayDense< Cell_Type, Data_Type >, 64
set_keygen
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 145
set_rengine
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 145
set_rules
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 145
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 187
set_rules_dyn
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 146
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 188
set_seed
Flock, 116
Geese, 131
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 146
shrink_to_fit
PhyloCounterData, 160
simulate
Geese, 131
size
BArrayDenseCol< Cell_Type, Data_Type >, 72
BArrayDenseCol_const< Cell_Type, Data_Type >, 75
BArrayDenseRow< Cell_Type, Data_Type >, 77
BArrayDenseRow_const< Cell_Type, Data_Type >, 79
BArrayVector< Cell_Type, Data_Type >, 88
BArrayVector_const< Cell_Type, Data_Type >, 92
Counters< Array_Type, Data_Type >, 108
FreqTable< T >, 120
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 146
PhyloCounterData, 161
PowerSet< Array_Type, Data_Rule_Type >, 168
Rules< Array_Type, Data_Type >, 176
size_unique
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 146
source
barray-meat.hpp, 217
barraydense-meat.hpp, 244
Entries< Cell_Type >, 110
states
NodeData, 158
Statistical Models, 11
stats_bank
support-meat.hpp, 337
StatsCounter
StatsCounter< Array_Type, Data_Type >, 178
StatsCounter< Array_Type, Data_Type >, 177
~StatsCounter, 178
add_counter, 179
count_all, 179
count_current, 179
count_init, 179
get_counters, 179
get_descriptions, 180
get_names, 180
reset_array, 180
set_counters, 180
StatsCounter, 178

statscounter-meat.hpp
clear, 321
counter, 323
counter_deleted, 323
counters, 323
counters_, 323
current_stats, 323
EmptyArray, 324
f_, 324
for, 321
j, 324
resize, 321
return, 324
STATSCOUNTER_TEMPLATE, 320–322
STATSCOUNTER_TEMPLATE_ARGS, 320
STATSCOUNTER_TYPE, 320
STATSCOUNTER_TEMPLATE
statscounter-meat.hpp, 320–322
STATSCOUNTER_TEMPLATE_ARGS
statscounter-meat.hpp, 320
STATSCOUNTER_TYPE
statscounter-meat.hpp, 320
store_psets
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 146
subtree_prob
Node, 156
Support
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 183
Support< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 181
∼Support, 183
add_counter, 184
add_rule, 184
add_rule_dyn, 184
calc, 185
change_stats, 188
coordiantes_n_free, 188
coordiantes_n_locked, 188
coordinates_free, 188
coordinates_locked, 189
current_stats, 189
delete_counters, 189
delete_rules, 189
delete_rules_dyn, 189
eval_rules_dyn, 185
get_counters, 185
get_counts, 185
get_current_stats, 186
get_data, 186
get_rules, 186
get_rules_dyn, 186
hashes, 190
hashes_initialized, 190
init_support, 186
M, 190
max_num_elements, 190
N, 190
n_counters, 191
print, 187
reset_array, 187
set_counters, 187
set_rules, 187
set_rules_dyn, 188
Support, 183
support-meat.hpp
array_bank, 334
BARRY_SUPPORT_MEAT_HPP, 328
calc_backend_dense, 329
calc_backend_sparse, 330
change_stats_different, 335
coord_i, 335
coord_j, 335
counters, 335
counters_, 335
delete_counters, 335
delete_rules, 336
delete_rules_dyn, 336
else, 336
f_, 336
for, 330
hashes, 336
if, 330
insert_cell, 331
return, 337
rm_cell, 331
rules, 337
rules_, 337
rules_dyn, 337
stats_bank, 337
SUPPORT_TEMPLATE, 329, 331–334
SUPPORT_TEMPLATE_ARGS, 329
SUPPORT_TYPE, 329
tmp_chng, 338
support_size
Flock, 116
Geese, 131
Model< Array_Type, Data_Counter_Type, Data_Rule_Type, Data_Rule_Dyn_Type >, 147
SUPPORT_TEMPLATE
support-meat.hpp, 329, 331–334
SUPPORT_TEMPLATE_ARGS
support-meat.hpp, 329
SUPPORT_TYPE
support-meat.hpp, 329
swap_cells
BArray< Cell_Type, Data_Type >, 45
BArrayDense< Cell_Type, Data_Type >, 64
swap_cols
BArray< Cell_Type, Data_Type >, 45
BArrayDense< Cell_Type, Data_Type >, 64
swap_rows

BArray< Cell_Type, Data_Type >, 45
BArrayDense< Cell_Type, Data_Type >, 64

target
barray-meat.hpp, 217
barraydense-meat.hpp, 244
Entries< Cell_Type >, 110
this
barray-meat-operators.hpp, 199
tmp_chng
support-meat.hpp, 338
to_be_deleted
counters-meat.hpp, 274
toggle_cell
BArray< Cell_Type, Data_Type >, 45
BArrayDense< Cell_Type, Data_Type >, 65
toggle_lock
BArray< Cell_Type, Data_Type >, 45
BArrayDense< Cell_Type, Data_Type >, 65
transpose
BArray< Cell_Type, Data_Type >, 46
BArrayDense< Cell_Type, Data_Type >, 65
TWO
CHECK, 30
EXISTS, 32
typedefs.hpp
Col_type, 340
Counter_fun_type, 340
Counts_type, 340
MapVec_type, 341
Row_type, 341
Rule_fun_type, 341
uint, 341
vec_equal, 341
vec_equal_approx, 342
vec_inner_prod, 342

ub
PhyloRuleDynData, 163
uint
typedefs.hpp, 341
UKNOWN
EXISTS, 32
update_annotations
Geese, 131
update_normalizing_constant
model-meat.hpp, 304

v
barray-meat.hpp, 217
barraydense-meat.hpp, 245
va_end
barraydense-meat.hpp, 239
va_start
barraydense-meat.hpp, 239
val
Entries< Cell_Type >, 110
val0
barraydense-meat.hpp, 245
val1
barraydense-meat.hpp, 245
value
barray-meat.hpp, 217
barraydense-meat.hpp, 245
Cell< Cell_Type >, 96
vec_diff
geese-bones.hpp, 307
vec_equal
typedefs.hpp, 341
vec_equal_approx
typedefs.hpp, 342
vec_inner_prod
typedefs.hpp, 342
vecHasher< T >, 191
operator(), 191
vector_caster
geese-bones.hpp, 307
vertex_attr
NetworkData, 150
visited
BArray< Cell_Type, Data_Type >, 47
BArrayDense< Cell_Type, Data_Type >, 67
Cell< Cell_Type >, 96
Node, 156
vprintf
barraydense-meat.hpp, 239

ZERO_CELL
barraydense-meat.hpp, 229
barraydensecol-bones.hpp, 249
barraydenserow-bones.hpp, 250
zero_col
BArray< Cell_Type, Data_Type >, 46
BArrayDense< Cell_Type, Data_Type >, 65
zero_row
BArray< Cell_Type, Data_Type >, 46
BArrayDense< Cell_Type, Data_Type >, 65