

# GDSiMS: Gene Drive Simulator of Mosquito Spread

Generated by Doxygen 1.10.0



<b>1 Namespace Index</b>	<b>1</b>
1.1 Namespace List	1
<b>2 Hierarchical Index</b>	<b>3</b>
2.1 Class Hierarchy	3
<b>3 Class Index</b>	<b>5</b>
3.1 Class List	5
<b>4 File Index</b>	<b>7</b>
4.1 File List	7
<b>5 Namespace Documentation</b>	<b>9</b>
5.1 constants Namespace Reference	9
5.1.1 Variable Documentation	9
5.1.1.1 max_dev	9
5.1.1.2 num_gen	9
<b>6 Class Documentation</b>	<b>11</b>
6.1 Aestivation Class Reference	11
6.1.1 Constructor & Destructor Documentation	12
6.1.1.1 Aestivation()	12
6.1.2 Member Function Documentation	12
6.1.2.1 hide()	12
6.1.2.2 is_hide_time()	13
6.1.2.3 is_wake_time()	13
6.1.2.4 wake()	13
6.1.3 Member Data Documentation	14
6.1.3.1 aes_F	14
6.1.3.2 mu_aes	14
6.1.3.3 psi	14
6.1.3.4 t_hide1	14
6.1.3.5 t_hide2	14
6.1.3.6 t_wake1	14
6.1.3.7 t_wake2	14
6.2 AestivationParams Struct Reference	15
6.2.1 Member Data Documentation	15
6.2.1.1 mu_aes	15
6.2.1.2 psi	15
6.2.1.3 t_hide1	15
6.2.1.4 t_hide2	16
6.2.1.5 t_wake1	16
6.2.1.6 t_wake2	16
6.3 AreaParams Struct Reference	16

6.3.1 Member Data Documentation	16
6.3.1.1 num_pat	16
6.3.1.2 side	17
6.4 BoundaryStrategy Class Reference	17
6.4.1 Constructor & Destructor Documentation	18
6.4.1.1 BoundaryStrategy()	18
6.4.2 Member Function Documentation	18
6.4.2.1 distance()	18
6.4.3 Member Data Documentation	18
6.4.3.1 side	18
6.5 Dispersal Class Reference	19
6.5.1 Constructor & Destructor Documentation	21
6.5.1.1 Dispersal()	21
6.5.1.2 ~Dispersal()	21
6.5.2 Member Function Documentation	21
6.5.2.1 adults_disperse()	21
6.5.2.2 F_dispersing_out()	21
6.5.2.3 M_dispersing_out()	22
6.5.2.4 set_connecs()	23
6.5.3 Member Data Documentation	23
6.5.3.1 boundary_strategy	23
6.5.3.2 connec_indices	23
6.5.3.3 connec_weights	23
6.5.3.4 disp_rate	23
6.5.3.5 max_disp	23
6.6 DispersalParams Struct Reference	24
6.6.1 Member Data Documentation	24
6.6.1.1 disp_rate	24
6.6.1.2 max_disp	24
6.7 DistanceKernelDispersal Class Reference	25
6.7.1 Constructor & Destructor Documentation	27
6.7.1.1 DistanceKernelDispersal()	27
6.7.2 Member Function Documentation	27
6.7.2.1 adults_disperse()	27
6.7.2.2 compute_connecs()	28
6.7.2.3 set_connecs()	28
6.8 EdgeBoundaryStrategy Class Reference	29
6.8.1 Constructor & Destructor Documentation	30
6.8.1.1 EdgeBoundaryStrategy()	30
6.8.2 Member Function Documentation	30
6.8.2.1 distance()	30
6.9 GDRelease Class Reference	30

6.9.1 Constructor & Destructor Documentation	31
6.9.1.1 GDRRelease()	31
6.9.2 Member Function Documentation	31
6.9.2.1 is_release_time()	31
6.9.2.2 put_driver_sites()	31
6.9.2.3 release_gene_drive()	32
6.9.2.4 select_driver_sites()	32
6.9.3 Member Data Documentation	33
6.9.3.1 driver_start	33
6.9.3.2 num_driver_M	33
6.9.3.3 num_driver_sites	33
6.10 InheritanceParams Struct Reference	33
6.10.1 Member Data Documentation	34
6.10.1.1 e	34
6.10.1.2 gamma	34
6.10.1.3 xi	34
6.11 InitialPopsParams Struct Reference	34
6.11.1 Member Data Documentation	35
6.11.1.1 initial_WF	35
6.11.1.2 initial_WJ	35
6.11.1.3 initial_WM	35
6.11.1.4 initial_WV	35
6.12 LifeParams Struct Reference	35
6.12.1 Member Data Documentation	36
6.12.1.1 alpha0	36
6.12.1.2 beta	36
6.12.1.3 mean_dev	36
6.12.1.4 min_dev	36
6.12.1.5 mu_a	36
6.12.1.6 mu_j	36
6.12.1.7 theta	36
6.13 Model Class Reference	37
6.13.1 Constructor & Destructor Documentation	38
6.13.1.1 Model()	38
6.13.1.2 ~Model()	39
6.13.2 Member Function Documentation	39
6.13.2.1 adults_die()	39
6.13.2.2 calculate_tot_F()	39
6.13.2.3 calculate_tot_J()	39
6.13.2.4 calculate_tot_M()	40
6.13.2.5 calculate_tot_M_gen()	40
6.13.2.6 calculate_tot_V()	40

6.13.2.7 <a href="#">get_sites()</a>	41
6.13.2.8 <a href="#">initiate()</a>	41
6.13.2.9 <a href="#">juv_eclose()</a>	42
6.13.2.10 <a href="#">juv_get_older()</a>	42
6.13.2.11 <a href="#">lay_eggs()</a>	42
6.13.2.12 <a href="#">populate_sites()</a>	42
6.13.2.13 <a href="#">run()</a>	43
6.13.2.14 <a href="#">run_step()</a>	43
6.13.2.15 <a href="#">set_dev_duration_probs()</a>	44
6.13.2.16 <a href="#">virgins_mate()</a>	45
6.13.3 Member Data Documentation	45
6.13.3.1 <a href="#">aestivation</a>	45
6.13.3.2 <a href="#">dev_duration_probs</a>	45
6.13.3.3 <a href="#">dispersal</a>	45
6.13.3.4 <a href="#">gd_release</a>	45
6.13.3.5 <a href="#">initial_pops</a>	45
6.13.3.6 <a href="#">min_dev</a>	45
6.13.3.7 <a href="#">num_pat</a>	45
6.13.3.8 <a href="#">side</a>	46
6.13.3.9 <a href="#">sites</a>	46
6.14 Patch Class Reference	46
6.14.1 Constructor & Destructor Documentation	48
6.14.1.1 <a href="#">Patch()</a> [1/2]	48
6.14.1.2 <a href="#">Patch()</a> [2/2]	49
6.14.2 Member Function Documentation	49
6.14.2.1 <a href="#">add_driver_M()</a>	49
6.14.2.2 <a href="#">adults_die()</a>	49
6.14.2.3 <a href="#">calculate_tot_F()</a>	50
6.14.2.4 <a href="#">calculate_tot_J()</a>	50
6.14.2.5 <a href="#">calculate_tot_M()</a>	50
6.14.2.6 <a href="#">calculate_tot_V()</a>	50
6.14.2.7 <a href="#">F_disperse_in()</a>	51
6.14.2.8 <a href="#">F_disperse_out()</a>	51
6.14.2.9 <a href="#">F_hide()</a>	51
6.14.2.10 <a href="#">F_wake()</a>	51
6.14.2.11 <a href="#">get_coords()</a>	51
6.14.2.12 <a href="#">get_F()</a>	51
6.14.2.13 <a href="#">get_M()</a>	51
6.14.2.14 <a href="#">juv_eclose()</a>	51
6.14.2.15 <a href="#">juv_get_older()</a>	52
6.14.2.16 <a href="#">lay_eggs()</a>	52
6.14.2.17 <a href="#">M_disperse_in()</a>	52

6.14.2.18 M_disperse_out()	53
6.14.2.19 populate()	53
6.14.2.20 update_comp()	53
6.14.2.21 update_mate()	54
6.14.2.22 virgins_mate()	55
6.14.3 Member Data Documentation	55
6.14.3.1 comp	55
6.14.3.2 coords	55
6.14.3.3 F	56
6.14.3.4 J	56
6.14.3.5 M	56
6.14.3.6 mate_rate	56
6.14.3.7 params	56
6.14.3.8 V	56
6.15 Point Struct Reference	56
6.15.1 Member Data Documentation	57
6.15.1.1 x	57
6.15.1.2 y	57
6.16 ProgressionParams Struct Reference	57
6.16.1 Member Data Documentation	57
6.16.1.1 max_t	57
6.16.1.2 num_runs	58
6.17 RadialDispersal Class Reference	58
6.17.1 Constructor & Destructor Documentation	60
6.17.1.1 RadialDispersal()	60
6.17.2 Member Function Documentation	61
6.17.2.1 adults_disperse()	61
6.17.2.2 compute_connecs()	61
6.17.2.3 compute_distances()	62
6.17.2.4 compute_interval_union()	62
6.17.2.5 get_sorted_positions()	63
6.17.2.6 set_connecs()	63
6.17.2.7 wrap_around()	63
6.17.3 Member Data Documentation	64
6.17.3.1 connec_weights_sum	64
6.18 Record Class Reference	64
6.18.1 Constructor & Destructor Documentation	65
6.18.1.1 Record()	65
6.18.1.2 ~Record()	65
6.18.2 Member Function Documentation	65
6.18.2.1 is_rec_global_time()	65
6.18.2.2 is_rec_local_time()	66

6.18.2.3 output_totals()	66
6.18.2.4 record_coords()	66
6.18.2.5 record_global()	67
6.18.2.6 record_local()	67
6.18.3 Member Data Documentation	67
6.18.3.1 coord_list	67
6.18.3.2 global_data	67
6.18.3.3 local_data	67
6.18.3.4 os1	68
6.18.3.5 os2	68
6.18.3.6 os3	68
6.18.3.7 rec_end	68
6.18.3.8 rec_interval_global	68
6.18.3.9 rec_interval_local	68
6.18.3.10 rec_sites_freq	68
6.18.3.11 rec_start	68
6.18.3.12 rep_label	68
6.18.3.13 set_label	68
6.19 RecordParams Struct Reference	69
6.19.1 Member Data Documentation	69
6.19.1.1 rec_end	69
6.19.1.2 rec_interval_global	69
6.19.1.3 rec_interval_local	69
6.19.1.4 rec_sites_freq	70
6.19.1.5 rec_start	70
6.19.1.6 set_label	70
6.20 ReleaseParams Struct Reference	70
6.20.1 Member Data Documentation	70
6.20.1.1 driver_start	70
6.20.1.2 num_driver_M	71
6.20.1.3 num_driver_sites	71
6.21 Simulation Class Reference	71
6.21.1 Constructor & Destructor Documentation	72
6.21.1.1 Simulation()	72
6.21.2 Member Function Documentation	72
6.21.2.1 run_reps()	72
6.21.2.2 set_boundary_type()	73
6.21.2.3 set_coords()	74
6.21.2.4 set_dispersal_type()	74
6.21.2.5 set_inheritance()	75
6.21.3 Member Data Documentation	75
6.21.3.1 aes_params	75



6.21.3.2 area_params . . . . .	75
6.21.3.3 boundary_type . . . . .	75
6.21.3.4 disp_params . . . . .	75
6.21.3.5 disp_type . . . . .	75
6.21.3.6 inher_fraction . . . . .	75
6.21.3.7 initial_params . . . . .	76
6.21.3.8 life_params . . . . .	76
6.21.3.9 max_t . . . . .	76
6.21.3.10 num_runs . . . . .	76
6.21.3.11 rec_params . . . . .	76
6.21.3.12 rel_params . . . . .	76
6.21.3.13 sites_coords . . . . .	76
6.22 ToroidalBoundaryStrategy Class Reference . . . . .	77
6.22.1 Constructor & Destructor Documentation . . . . .	78
6.22.1.1 ToroidalBoundaryStrategy() . . . . .	78
6.22.2 Member Function Documentation . . . . .	78
6.22.2.1 distance() . . . . .	78
<b>7 File Documentation</b> . . . . .	<b>79</b>
7.1 Aestivation.cpp File Reference . . . . .	79
7.2 Aestivation.h File Reference . . . . .	79
7.3 Aestivation.h . . . . .	80
7.4 BoundaryStrategy.cpp File Reference . . . . .	81
7.5 BoundaryStrategy.h File Reference . . . . .	82
7.6 BoundaryStrategy.h . . . . .	83
7.7 constants.h File Reference . . . . .	83
7.7.1 Enumeration Type Documentation . . . . .	83
7.7.1.1 BoundaryType . . . . .	83
7.7.1.2 DispersalType . . . . .	84
7.8 constants.h . . . . .	84
7.9 Dispersal.cpp File Reference . . . . .	84
7.9.1 Variable Documentation . . . . .	85
7.9.1.1 PI . . . . .	85
7.9.1.2 TWOPI . . . . .	85
7.10 Dispersal.h File Reference . . . . .	85
7.11 Dispersal.h . . . . .	86
7.12 GDRelease.cpp File Reference . . . . .	87
7.13 GDRelease.h File Reference . . . . .	88
7.14 GDRelease.h . . . . .	89
7.15 input.h File Reference . . . . .	89
7.15.1 Function Documentation . . . . .	90
7.15.1.1 check_bounds() . . . . .	90

7.15.1.2 read_and_validate_type() [1/2]	91
7.15.1.3 read_and_validate_type() [2/2]	91
7.16 input.h	92
7.17 main.cpp File Reference	92
7.17.1 Function Documentation	93
7.17.1.1 main()	93
7.18 Model.cpp File Reference	94
7.19 Model.h File Reference	94
7.20 Model.h	95
7.21 Params.h File Reference	96
7.22 Params.h	97
7.23 Patch.cpp File Reference	98
7.24 Patch.h File Reference	98
7.25 Patch.h	99
7.26 Point.h File Reference	100
7.27 Point.h	100
7.28 random.cpp File Reference	101
7.28.1 Function Documentation	101
7.28.1.1 random_binomial()	101
7.28.1.2 random_discrete()	102
7.28.1.3 random_multinomial() [1/3]	103
7.28.1.4 random_multinomial() [2/3]	103
7.28.1.5 random_multinomial() [3/3]	103
7.28.1.6 random_poisson()	104
7.28.1.7 random_real()	105
7.28.1.8 twister()	106
7.28.2 Variable Documentation	106
7.28.2.1 rd	106
7.29 random.h File Reference	106
7.29.1 Function Documentation	107
7.29.1.1 random_binomial()	107
7.29.1.2 random_discrete()	108
7.29.1.3 random_multinomial() [1/3]	109
7.29.1.4 random_multinomial() [2/3]	109
7.29.1.5 random_multinomial() [3/3]	109
7.29.1.6 random_poisson()	110
7.29.1.7 random_real()	111
7.30 random.h	112
7.31 Record.cpp File Reference	112
7.32 Record.h File Reference	112
7.33 Record.h	113
7.34 Simulation.cpp File Reference	114

---

7.34.1 Function Documentation . . . . .	114
7.34.1.1 invalid_interval_msg() . . . . .	114
7.34.1.2 out_of_bounds_msg() . . . . .	115
7.35 Simulation.h File Reference . . . . .	115
7.35.1 Function Documentation . . . . .	116
7.35.1.1 invalid_interval_msg() . . . . .	116
7.35.1.2 out_of_bounds_msg() . . . . .	117
7.36 Simulation.h . . . . .	117
<b>Index</b>	<b>119</b>



# Chapter 1

## Namespace Index

### 1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<a href="#">constants</a>	9
---------------------------	---



## Chapter 2

# Hierarchical Index

### 2.1 Class Hierarchy

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

Aestivation . . . . .	11
AestivationParams . . . . .	15
AreaParams . . . . .	16
BoundaryStrategy . . . . .	17
EdgeBoundaryStrategy . . . . .	29
ToroidalBoundaryStrategy . . . . .	77
Dispersal . . . . .	19
DistanceKernelDispersal . . . . .	25
RadialDispersal . . . . .	58
DispersalParams . . . . .	24
GDRelease . . . . .	30
InheritanceParams . . . . .	33
InitialPopsParams . . . . .	34
LifeParams . . . . .	35
Model . . . . .	37
Patch . . . . .	46
Point . . . . .	56
ProgressionParams . . . . .	57
Record . . . . .	64
RecordParams . . . . .	69
ReleaseParams . . . . .	70
Simulation . . . . .	71





## Chapter 3

# Class Index

### 3.1 Class List

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

<a href="#">Aestivation</a>	11
<a href="#">AestivationParams</a>	15
<a href="#">AreaParams</a>	16
<a href="#">BoundaryStrategy</a>	17
<a href="#">Dispersal</a>	19
<a href="#">DispersalParams</a>	24
<a href="#">DistanceKernelDispersal</a>	25
<a href="#">EdgeBoundaryStrategy</a>	29
<a href="#">GDRelease</a>	30
<a href="#">InheritanceParams</a>	33
<a href="#">InitialPopsParams</a>	34
<a href="#">LifeParams</a>	35
<a href="#">Model</a>	37
<a href="#">Patch</a>	46
<a href="#">Point</a>	56
<a href="#">ProgressionParams</a>	57
<a href="#">RadialDispersal</a>	58
<a href="#">Record</a>	64
<a href="#">RecordParams</a>	69
<a href="#">ReleaseParams</a>	70
<a href="#">Simulation</a>	71
<a href="#">ToroidalBoundaryStrategy</a>	77



## Chapter 4

# File Index

### 4.1 File List

Here is a list of all files with brief descriptions:

Aestivation.cpp	79
Aestivation.h	79
BoundaryStrategy.cpp	81
BoundaryStrategy.h	82
constants.h	83
Dispersal.cpp	84
Dispersal.h	85
GDRelease.cpp	87
GDRelease.h	88
input.h	89
main.cpp	92
Model.cpp	94
Model.h	94
Params.h	96
Patch.cpp	98
Patch.h	98
Point.h	100
random.cpp	101
random.h	106
Record.cpp	112
Record.h	112
Simulation.cpp	114
Simulation.h	115



## Chapter 5

# Namespace Documentation

### 5.1 constants Namespace Reference

#### Variables

- const int [max\\_dev](#) = 20
- const int [num\\_gen](#) = 6

#### 5.1.1 Variable Documentation

##### 5.1.1.1 max\_dev

```
const int constants::max_dev = 20
```

##### 5.1.1.2 num\_gen

```
const int constants::num_gen = 6
```



## Chapter 6

# Class Documentation

### 6.1 Aestivation Class Reference

```
#include <Aestivation.h>
```

Collaboration diagram for Aestivation:

Aestivation
<ul style="list-style-type: none"><li>- psi</li><li>- mu_aes</li><li>- t_hide1</li><li>- t_hide2</li><li>- t_wake1</li><li>- t_wake2</li><li>- aes_F</li></ul>
<ul style="list-style-type: none"><li>+ Aestivation()</li><li>+ hide()</li><li>+ wake()</li><li>+ is_hide_time()</li><li>+ is_wake_time()</li></ul>

#### Public Member Functions

- [Aestivation](#) ([AestivationParams](#) \*params, int sites\_size)
- void [hide](#) (std::vector< [Patch](#) \* > &sites)
- void [wake](#) (int day, std::vector< [Patch](#) \* > &sites)
- bool [is\\_hide\\_time](#) (int day)
- bool [is\\_wake\\_time](#) (int day)

## Private Attributes

- double [psi](#)
- double [mu\\_aes](#)
- int [t\\_hide1](#)
- int [t\\_hide2](#)
- int [t\\_wake1](#)
- int [t\\_wake2](#)
- `std::vector< std::array< std::array< long long int, num\_gen >, num\_gen > > aes\_F`

## 6.1.1 Constructor & Destructor Documentation

### 6.1.1.1 Aestivation()

```
Aestivation::Aestivation (
    AestivationParams * params,
    int sites_size )
```

## 6.1.2 Member Function Documentation

### 6.1.2.1 hide()

```
void Aestivation::hide (
    std::vector< Patch * > & sites )
```

Here is the call graph for this function:



Here is the caller graph for this function:





### 6.1.2.2 is\_hide\_time()

```
bool Aestivation::is_hide_time (
    int day )
```

Here is the caller graph for this function:



### 6.1.2.3 is\_wake\_time()

```
bool Aestivation::is_wake_time (
    int day )
```

Here is the caller graph for this function:



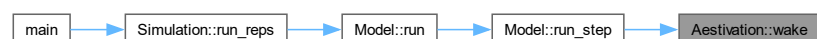
### 6.1.2.4 wake()

```
void Aestivation::wake (
    int day,
    std::vector< Patch * > & sites )
```

Here is the call graph for this function:



Here is the caller graph for this function:



### 6.1.3 Member Data Documentation

#### 6.1.3.1 aes\_F

```
std::vector<std::array<std::array<long long int, num_gen>, num_gen> > Aestivation::aes_F [private]
```

#### 6.1.3.2 mu\_aes

```
double Aestivation::mu_aes [private]
```

#### 6.1.3.3 psi

```
double Aestivation::psi [private]
```

#### 6.1.3.4 t\_hide1

```
int Aestivation::t_hide1 [private]
```

#### 6.1.3.5 t\_hide2

```
int Aestivation::t_hide2 [private]
```

#### 6.1.3.6 t\_wake1

```
int Aestivation::t_wake1 [private]
```

#### 6.1.3.7 t\_wake2

```
int Aestivation::t_wake2 [private]
```

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

- [Aestivation.h](#)
- [Aestivation.cpp](#)

## 6.2 AestivationParams Struct Reference

```
#include <Params.h>
```

Collaboration diagram for AestivationParams:

AestivationParams
+ psi
+ mu_aes
+ t_hide1
+ t_hide2
+ t_wake1
+ t_wake2

### Public Attributes

- double [psi](#) = 0.0
- double [mu\\_aes](#) = 0.0
- int [t\\_hide1](#) = 0
- int [t\\_hide2](#) = 0
- int [t\\_wake1](#) = 0
- int [t\\_wake2](#) = 0

### 6.2.1 Member Data Documentation

#### 6.2.1.1 mu\_aes

```
double AestivationParams::mu_aes = 0.0
```

#### 6.2.1.2 psi

```
double AestivationParams::psi = 0.0
```

#### 6.2.1.3 t\_hide1

```
int AestivationParams::t_hide1 = 0
```

#### 6.2.1.4 t\_hide2

```
int AestivationParams::t_hide2 = 0
```

#### 6.2.1.5 t\_wake1

```
int AestivationParams::t_wake1 = 0
```

#### 6.2.1.6 t\_wake2

```
int AestivationParams::t_wake2 = 0
```

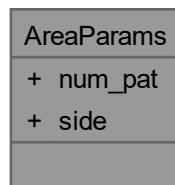
The documentation for this struct was generated from the following file:

- [Params.h](#)

## 6.3 AreaParams Struct Reference

```
#include <Params.h>
```

Collaboration diagram for AreaParams:



### Public Attributes

- int [num\\_pat](#) = 50
- double [side](#) = 1.0

### 6.3.1 Member Data Documentation

#### 6.3.1.1 num\_pat

```
int AreaParams::num_pat = 50
```

### 6.3.1.2 side

```
double AreaParams::side = 1.0
```

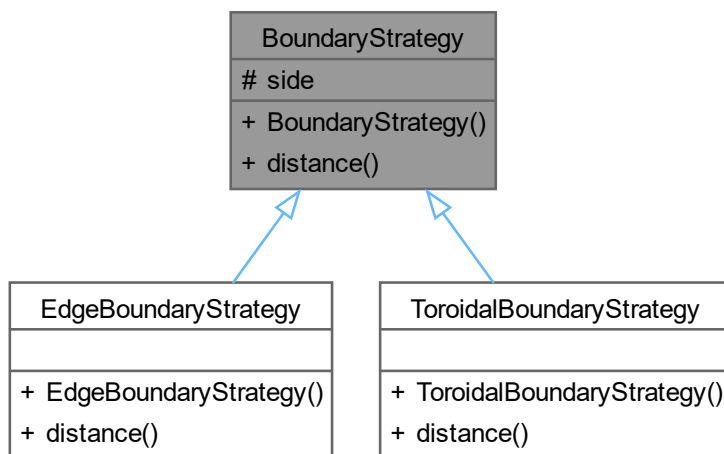
The documentation for this struct was generated from the following file:

- [Params.h](#)

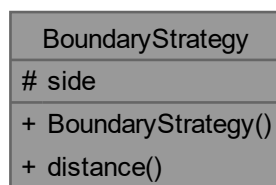
## 6.4 BoundaryStrategy Class Reference

```
#include <BoundaryStrategy.h>
```

Inheritance diagram for BoundaryStrategy:



Collaboration diagram for BoundaryStrategy:



## Public Member Functions

- [BoundaryStrategy](#) (double [side](#))
- virtual double [distance](#) (const [Point](#) &p1, const [Point](#) &p2)=0

## Protected Attributes

- double [side](#)

## 6.4.1 Constructor & Destructor Documentation

### 6.4.1.1 BoundaryStrategy()

```
BoundaryStrategy::BoundaryStrategy (
    double side ) [inline]
```

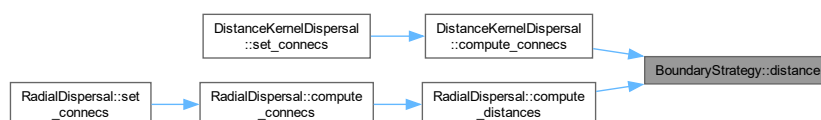
## 6.4.2 Member Function Documentation

### 6.4.2.1 distance()

```
virtual double BoundaryStrategy::distance (
    const Point & p1,
    const Point & p2 ) [pure virtual]
```

Implemented in [ToroidalBoundaryStrategy](#), and [EdgeBoundaryStrategy](#).

Here is the caller graph for this function:



## 6.4.3 Member Data Documentation

### 6.4.3.1 side

```
double BoundaryStrategy::side [protected]
```

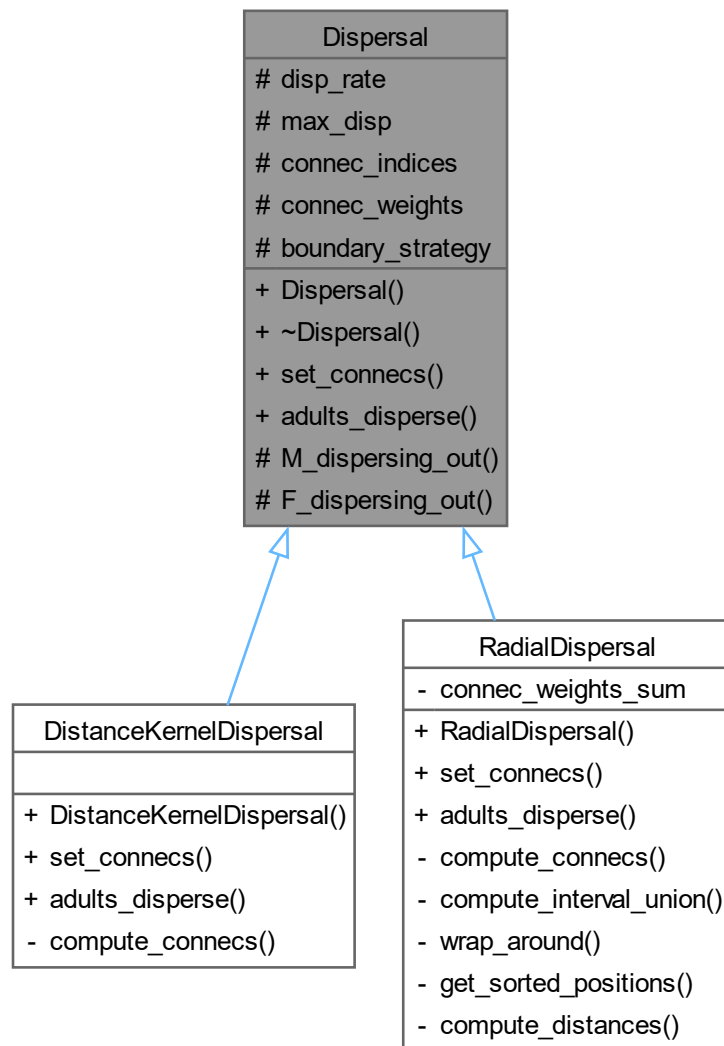
The documentation for this class was generated from the following file:

- [BoundaryStrategy.h](#)

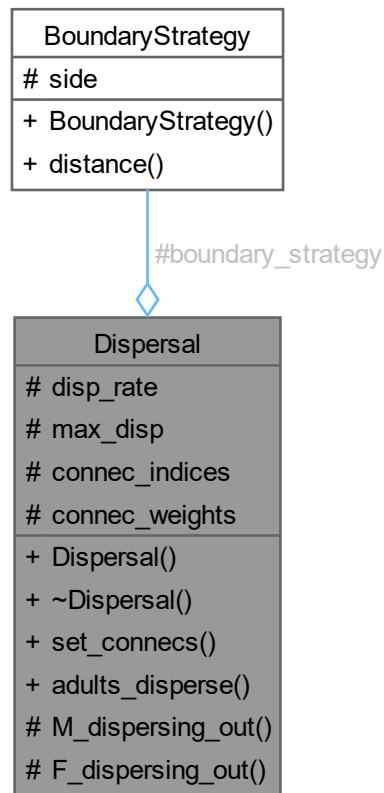
## 6.5 Dispersal Class Reference

```
#include <Dispersal.h>
```

Inheritance diagram for Dispersal:



Collaboration diagram for Dispersal:



### Public Member Functions

- [Dispersal](#) ([DispersalParams](#) \*params, [BoundaryType](#) boundary, double side)
- [~Dispersal](#) ()
- virtual void [set\\_convecs](#) (std::vector< [Patch](#) \* > &sites)=0
- virtual void [adults\\_disperse](#) (std::vector< [Patch](#) \* > &sites)=0

### Protected Member Functions

- std::vector< std::array< long long int, [num\\_gen](#) > > [M\\_dispersing\\_out](#) (const std::vector< [Patch](#) \* > &sites)
- std::vector< std::array< std::array< long long int, [num\\_gen](#) >, [num\\_gen](#) > > [F\\_dispersing\\_out](#) (const std::vector< [Patch](#) \* > &sites)

### Protected Attributes

- double [disp\\_rate](#)
- double [max\\_disp](#)
- std::vector< std::vector< int > > [convec\\_indices](#)
- std::vector< std::vector< double > > [convec\\_weights](#)
- [BoundaryStrategy](#) \* [boundary\\_strategy](#)



## 6.5.1 Constructor & Destructor Documentation

### 6.5.1.1 Dispersal()

```
Dispersal::Dispersal (
    DispersalParams * params,
    BoundaryType boundary,
    double side )
```

### 6.5.1.2 ~Dispersal()

```
Dispersal::~Dispersal ( )
```

## 6.5.2 Member Function Documentation

### 6.5.2.1 adults\_disperse()

```
virtual void Dispersal::adults_disperse (
    std::vector< Patch * > & sites ) [pure virtual]
```

Implemented in [DistanceKernelDispersal](#), and [RadialDispersal](#).

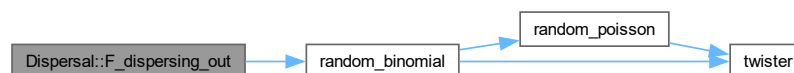
Here is the caller graph for this function:



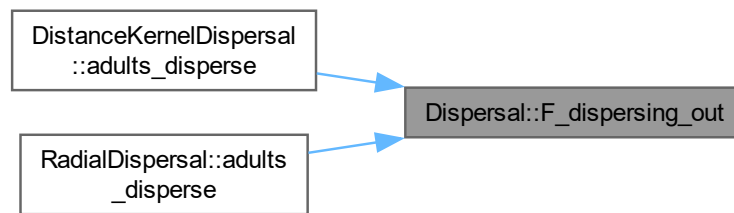
### 6.5.2.2 F\_dispersing\_out()

```
std::vector< std::array< std::array< long long int, num_gen >, num_gen > > Dispersal::F_↔
dispersing_out (
    const std::vector< Patch * > & sites ) [protected]
```

Here is the call graph for this function:



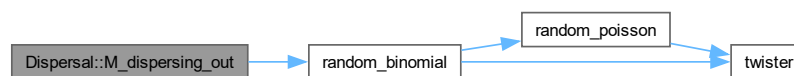
Here is the caller graph for this function:



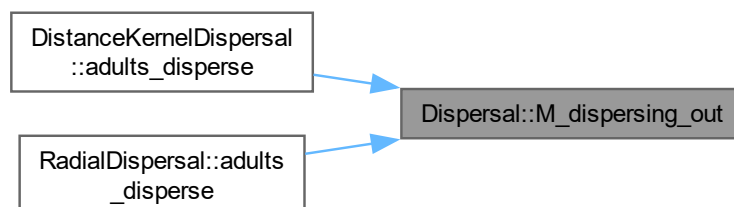
### 6.5.2.3 M\_dispersing\_out()

```
std::vector< std::array< long long int, num_gen > > Dispersal::M_dispersing_out (
    const std::vector< Patch * > & sites ) [protected]
```

Here is the call graph for this function:



Here is the caller graph for this function:



#### 6.5.2.4 set\_connecs()

```
virtual void Dispersal::set_connecs (
    std::vector< Patch * > & sites ) [pure virtual]
```

Implemented in [DistanceKernelDispersal](#), and [RadialDispersal](#).

Here is the caller graph for this function:



### 6.5.3 Member Data Documentation

#### 6.5.3.1 boundary\_strategy

```
BoundaryStrategy* Dispersal::boundary_strategy [protected]
```

#### 6.5.3.2 connec\_indices

```
std::vector<std::vector<int> > Dispersal::connec_indices [protected]
```

#### 6.5.3.3 connec\_weights

```
std::vector<std::vector<double> > Dispersal::connec_weights [protected]
```

#### 6.5.3.4 disp\_rate

```
double Dispersal::disp_rate [protected]
```

#### 6.5.3.5 max\_disp

```
double Dispersal::max_disp [protected]
```

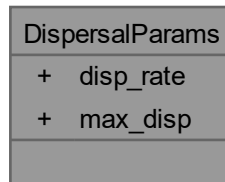
The documentation for this class was generated from the following files:

- [Dispersal.h](#)
- [Dispersal.cpp](#)

## 6.6 DispersalParams Struct Reference

```
#include <Params.h>
```

Collaboration diagram for DispersalParams:



### Public Attributes

- double [disp\\_rate](#) = 0.01
- double [max\\_disp](#) = 0.2

### 6.6.1 Member Data Documentation

#### 6.6.1.1 [disp\\_rate](#)

```
double DispersalParams::disp_rate = 0.01
```

#### 6.6.1.2 [max\\_disp](#)

```
double DispersalParams::max_disp = 0.2
```

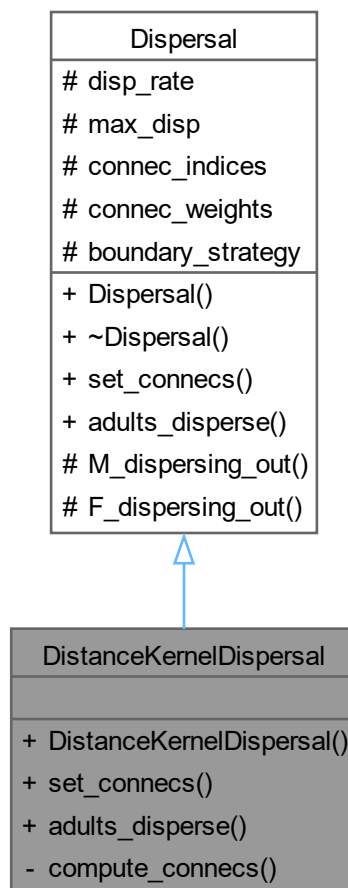
The documentation for this struct was generated from the following file:

- [Params.h](#)

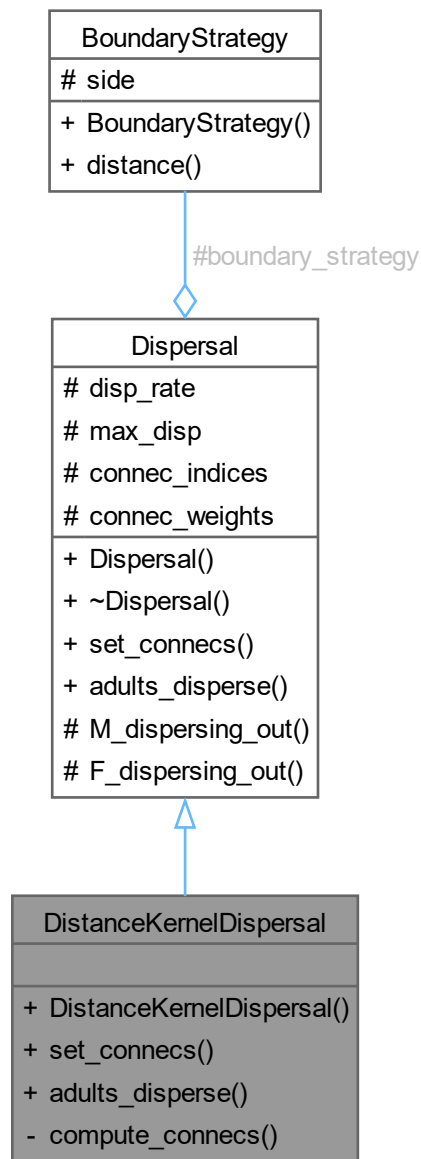
## 6.7 DistanceKernelDispersal Class Reference

```
#include <Dispersal.h>
```

Inheritance diagram for DistanceKernelDispersal:



Collaboration diagram for DistanceKernelDispersal:



### Public Member Functions

- [DistanceKernelDispersal](#) ([DispersalParams](#) \*params, [BoundaryType](#) boundary, double side)
- void [set\\_connecs](#) (std::vector< [Patch](#) \* > &sites) override
- void [adults\\_disperse](#) (std::vector< [Patch](#) \* > &sites) override

### Public Member Functions inherited from [Dispersal](#)

- [Dispersal](#) ([DispersalParams](#) \*params, [BoundaryType](#) boundary, double side)
- [~Dispersal](#) ()

## Private Member Functions

- `std::pair< std::vector< std::vector< int > >, std::vector< std::vector< double > > >` [compute\\_connecs](#) (`std::vector< Patch * >` &sites)

## Additional Inherited Members

## Protected Member Functions inherited from [Dispersal](#)

- `std::vector< std::array< long long int, num\_gen > >` [M\\_dispersing\\_out](#) (`const std::vector< Patch * >` &sites)
- `std::vector< std::array< std::array< long long int, num\_gen >, num\_gen > >` [F\\_dispersing\\_out](#) (`const std::vector< Patch * >` &sites)

## Protected Attributes inherited from [Dispersal](#)

- double [disp\\_rate](#)
- double [max\\_disp](#)
- `std::vector< std::vector< int > >` [connec\\_indices](#)
- `std::vector< std::vector< double > >` [connec\\_weights](#)
- [BoundaryStrategy](#) \* [boundary\\_strategy](#)

## 6.7.1 Constructor & Destructor Documentation

### 6.7.1.1 DistanceKernelDispersal()

```
DistanceKernelDispersal::DistanceKernelDispersal (
    DispersalParams * params,
    BoundaryType boundary,
    double side ) [inline]
```

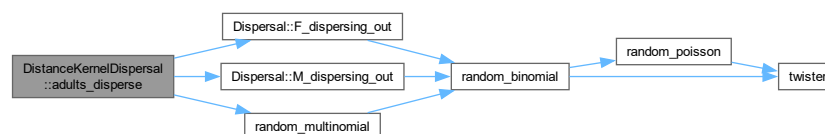
## 6.7.2 Member Function Documentation

### 6.7.2.1 adults\_disperse()

```
void DistanceKernelDispersal::adults_disperse (
    std::vector< Patch * > & sites ) [override], [virtual]
```

Implements [Dispersal](#).

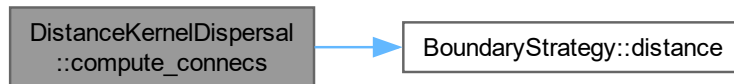
Here is the call graph for this function:



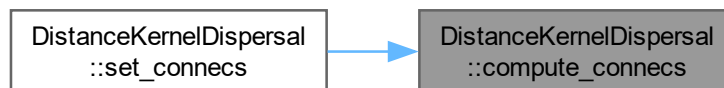
### 6.7.2.2 compute\_connects()

```
std::pair< std::vector< std::vector< int > >, std::vector< std::vector< double > > > DistanceKernelDispersal::compute_connects (
    std::vector< Patch * > & sites ) [private]
```

Here is the call graph for this function:



Here is the caller graph for this function:

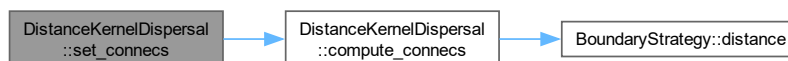


### 6.7.2.3 set\_connects()

```
void DistanceKernelDispersal::set_connects (
    std::vector< Patch * > & sites ) [override], [virtual]
```

Implements [Dispersal](#).

Here is the call graph for this function:



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

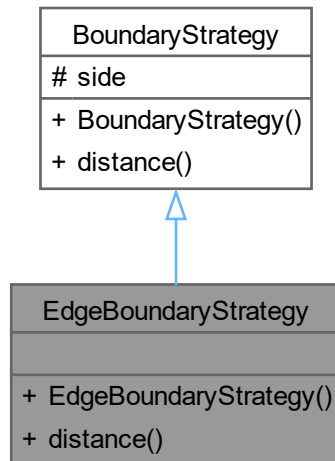
- [Dispersal.h](#)
- [Dispersal.cpp](#)



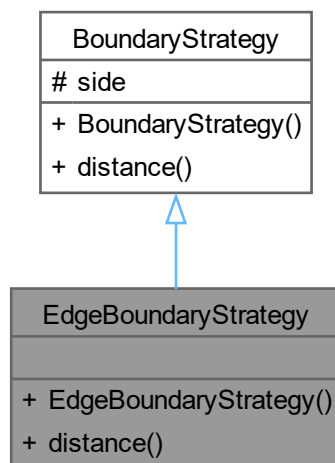
## 6.8 EdgeBoundaryStrategy Class Reference

```
#include <BoundaryStrategy.h>
```

Inheritance diagram for EdgeBoundaryStrategy:



Collaboration diagram for EdgeBoundaryStrategy:



### Public Member Functions

- [EdgeBoundaryStrategy](#) (double [side](#))
- double [distance](#) (const [Point](#) &p1, const [Point](#) &p2) override

## Public Member Functions inherited from [BoundaryStrategy](#)

- [BoundaryStrategy](#) (double [side](#))

## Additional Inherited Members

## Protected Attributes inherited from [BoundaryStrategy](#)

- double [side](#)

## 6.8.1 Constructor & Destructor Documentation

### 6.8.1.1 [EdgeBoundaryStrategy](#)()

```
EdgeBoundaryStrategy::EdgeBoundaryStrategy (
    double side ) [inline]
```

## 6.8.2 Member Function Documentation

### 6.8.2.1 [distance](#)()

```
double EdgeBoundaryStrategy::distance (
    const Point & p1,
    const Point & p2 ) [override], [virtual]
```

Implements [BoundaryStrategy](#).

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

- [BoundaryStrategy.h](#)
- [BoundaryStrategy.cpp](#)

## 6.9 GDRelease Class Reference

```
#include <GDRelease.h>
```

Collaboration diagram for GDRelease:

GDRelease
<ul style="list-style-type: none"> <li>- driver_start</li> <li>- num_driver_M</li> <li>- num_driver_sites</li> </ul>
<ul style="list-style-type: none"> <li>+ GDRelease()</li> <li>+ release_gene_drive()</li> <li>+ is_release_time()</li> <li>- select_driver_sites()</li> <li>- put_driver_sites()</li> </ul>

### Public Member Functions

- [GDRelease](#) ([ReleaseParams](#) \*params)
- void [release\\_gene\\_drive](#) (std::vector< [Patch](#) \* > &sites)
- bool [is\\_release\\_time](#) (int day)

### Private Member Functions

- std::vector< [Patch](#) \* > [select\\_driver\\_sites](#) (int num\_rel\_sites, const std::vector< [Patch](#) \* > &sites)
- void [put\\_driver\\_sites](#) (std::vector< [Patch](#) \* > &rel\_sites, std::vector< [Patch](#) \* > &sites)

### Private Attributes

- int [driver\\_start](#)
- int [num\\_driver\\_M](#)
- int [num\\_driver\\_sites](#)

## 6.9.1 Constructor & Destructor Documentation

### 6.9.1.1 GDRelease()

```
GDRelease::GDRelease (
    ReleaseParams * params )
```

## 6.9.2 Member Function Documentation

### 6.9.2.1 is\_release\_time()

```
bool GDRelease::is_release_time (
    int day )
```

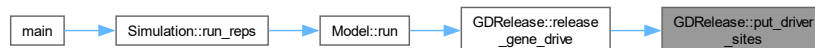
Here is the caller graph for this function:



### 6.9.2.2 put\_driver\_sites()

```
void GDRelease::put_driver_sites (
    std::vector< Patch * > & rel_sites,
    std::vector< Patch * > & sites ) [private]
```

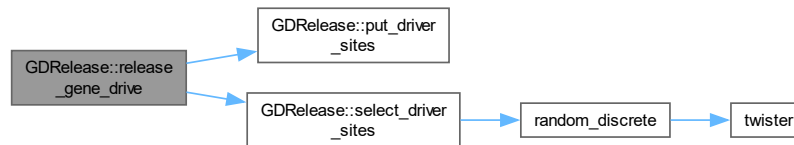
Here is the caller graph for this function:



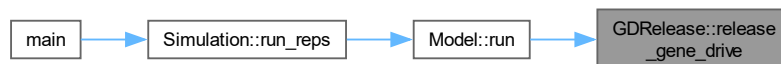
### 6.9.2.3 release\_gene\_drive()

```
void GDRelease::release_gene_drive (
    std::vector< Patch * > & sites )
```

Here is the call graph for this function:



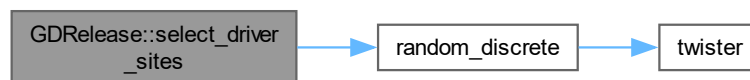
Here is the caller graph for this function:



### 6.9.2.4 select\_driver\_sites()

```
std::vector< Patch * > GDRelease::select_driver_sites (
    int num_rel_sites,
    const std::vector< Patch * > & sites ) [private]
```

Here is the call graph for this function:



Here is the caller graph for this function:



## 6.9.3 Member Data Documentation

### 6.9.3.1 driver\_start

```
int GDRelease::driver_start [private]
```

### 6.9.3.2 num\_driver\_M

```
int GDRelease::num_driver_M [private]
```

### 6.9.3.3 num\_driver\_sites

```
int GDRelease::num_driver_sites [private]
```

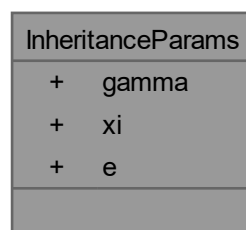
The documentation for this class was generated from the following files:

- [GDRelease.h](#)
- [GDRelease.cpp](#)

## 6.10 InheritanceParams Struct Reference

```
#include <Params.h>
```

Collaboration diagram for InheritanceParams:



### Public Attributes

- double [gamma](#) = 0.025
- double [xi](#) = 0.2
- double [e](#) = 0.95

## 6.10.1 Member Data Documentation

### 6.10.1.1 e

```
double InheritanceParams::e = 0.95
```

### 6.10.1.2 gamma

```
double InheritanceParams::gamma = 0.025
```

### 6.10.1.3 xi

```
double InheritanceParams::xi = 0.2
```

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

- [Params.h](#)

## 6.11 InitialPopsParams Struct Reference

```
#include <Params.h>
```

Collaboration diagram for InitialPopsParams:

InitialPopsParams
+ initial_WJ
+ initial_WM
+ initial_WV
+ initial_WF

### Public Attributes

- int [initial\\_WJ](#) = 10000
- int [initial\\_WM](#) = 50000
- int [initial\\_WV](#) = 10000
- int [initial\\_WF](#) = 40000

## 6.11.1 Member Data Documentation

### 6.11.1.1 initial\_WF

```
int InitialPopsParams::initial_WF = 40000
```

### 6.11.1.2 initial\_WJ

```
int InitialPopsParams::initial_WJ = 10000
```

### 6.11.1.3 initial\_WM

```
int InitialPopsParams::initial_WM = 50000
```

### 6.11.1.4 initial\_WV

```
int InitialPopsParams::initial_WV = 10000
```

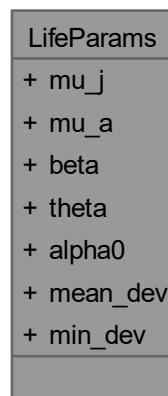
The documentation for this struct was generated from the following file:

- [Params.h](#)

## 6.12 LifeParams Struct Reference

```
#include <Params.h>
```

Collaboration diagram for LifeParams:



## Public Attributes

- double `mu_j` = 0.05
- double `mu_a` = 0.125
- double `beta` = 100.0
- double `theta` = 9.0
- double `alpha0` = 100000.0
- double `mean_dev` = 15.0
- int `min_dev` = 10

## 6.12.1 Member Data Documentation

### 6.12.1.1 `alpha0`

```
double LifeParams::alpha0 = 100000.0
```

### 6.12.1.2 `beta`

```
double LifeParams::beta = 100.0
```

### 6.12.1.3 `mean_dev`

```
double LifeParams::mean_dev = 15.0
```

### 6.12.1.4 `min_dev`

```
int LifeParams::min_dev = 10
```

### 6.12.1.5 `mu_a`

```
double LifeParams::mu_a = 0.125
```

### 6.12.1.6 `mu_j`

```
double LifeParams::mu_j = 0.05
```

### 6.12.1.7 `theta`

```
double LifeParams::theta = 9.0
```

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

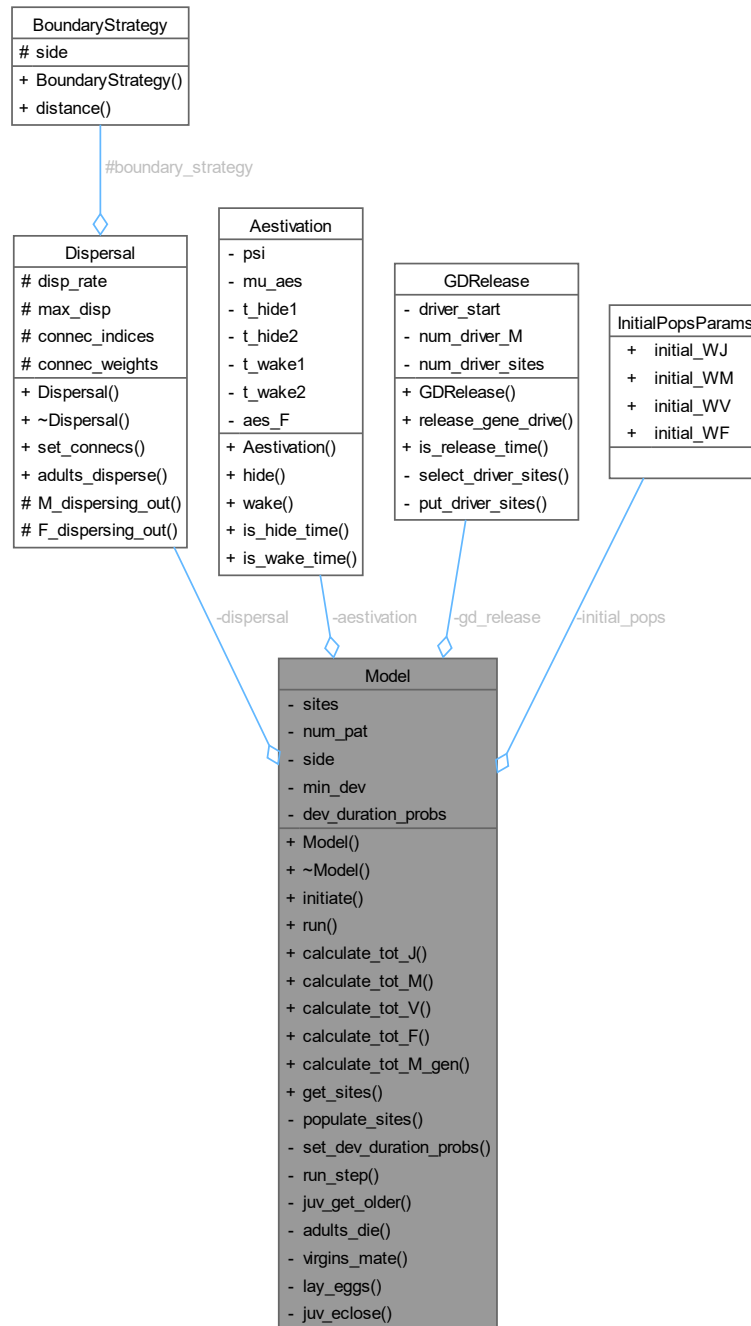
- [Params.h](#)



## 6.13 Model Class Reference

```
#include <Model.h>
```

Collaboration diagram for Model:



### Public Member Functions

- [Model](#) ([AreaParams](#) \*area, [InitialPopsParams](#) \*initial, [LifeParams](#) \*life, [AestivationParams](#) \*aes, [DispersalParams](#) \*disp, [ReleaseParams](#) \*rel, [BoundaryType](#) boundary=[BoundaryType::Toroid](#), [DispersalType](#) disp\_type=[DispersalType::DistanceKernel](#), [std::vector](#)< [Point](#) > coords={})

- `~Model ()`
- `void initiate ()`
- `void run (int day, const std::array< std::array< std::array< double, num_gen >, num_gen >, num_gen > &inher_fraction)`
- `long long int calculate_tot_J ()`
- `long long int calculate_tot_M ()`
- `long long int calculate_tot_V ()`
- `long long int calculate_tot_F ()`
- `std::array< long long int, num_gen > calculate_tot_M_gen ()`
- `std::vector< Patch * > get_sites () const`

### Private Member Functions

- `void populate_sites ()`
- `void set_dev_duration_probs (int min_time, int max_time)`
- `void run_step (int day, const std::array< std::array< std::array< double, num_gen >, num_gen >, num_gen > &inher_fraction)`
- `void juv_get_older ()`
- `void adults_die ()`
- `void virgins_mate ()`
- `void lay_eggs (const std::array< std::array< std::array< double, num_gen >, num_gen >, num_gen > &inher_fraction)`
- `void juv_eclose ()`

### Private Attributes

- `std::vector< Patch * > sites`
- `Dispersal * dispersal`
- `Aestivation * aestivation`
- `GDRRelease * gd_release`
- `int num_pat`
- `double side`
- `InitialPopsParams * initial_pops`
- `int min_dev`
- `std::array< double, max_dev+1 > dev_duration_probs`

## 6.13.1 Constructor & Destructor Documentation

### 6.13.1.1 Model()

```
Model::Model (
    AreaParams * area,
    InitialPopsParams * initial,
    LifeParams * life,
    AestivationParams * aes,
    DispersalParams * disp,
    ReleaseParams * rel,
    BoundaryType boundary = BoundaryType::Toroid,
    DispersalType disp_type = DispersalType::DistanceKernel,
    std::vector< Point > coords = {} )
```

### 6.13.1.2 ~Model()

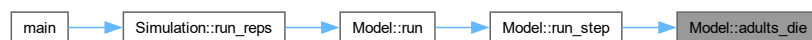
```
Model::~~Model ( )
```

## 6.13.2 Member Function Documentation

### 6.13.2.1 adults\_die()

```
void Model::adults_die ( ) [private]
```

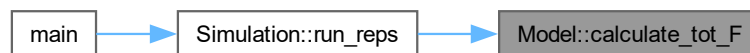
Here is the caller graph for this function:



### 6.13.2.2 calculate\_tot\_F()

```
long long int Model::calculate_tot_F ( )
```

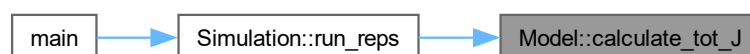
Here is the caller graph for this function:



### 6.13.2.3 calculate\_tot\_J()

```
long long int Model::calculate_tot_J ( )
```

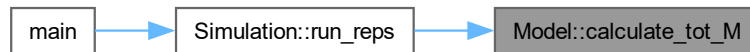
Here is the caller graph for this function:



#### 6.13.2.4 calculate\_tot\_M()

```
long long int Model::calculate_tot_M ( )
```

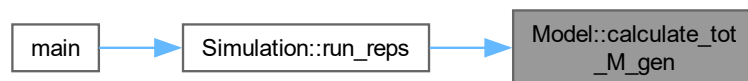
Here is the caller graph for this function:



#### 6.13.2.5 calculate\_tot\_M\_gen()

```
std::array< long long int, num_gen > Model::calculate_tot_M_gen ( )
```

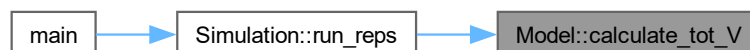
Here is the caller graph for this function:



#### 6.13.2.6 calculate\_tot\_V()

```
long long int Model::calculate_tot_V ( )
```

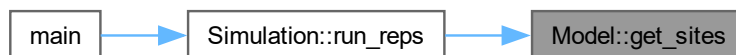
Here is the caller graph for this function:



### 6.13.2.7 get\_sites()

```
std::vector< Patch * > Model::get_sites ( ) const
```

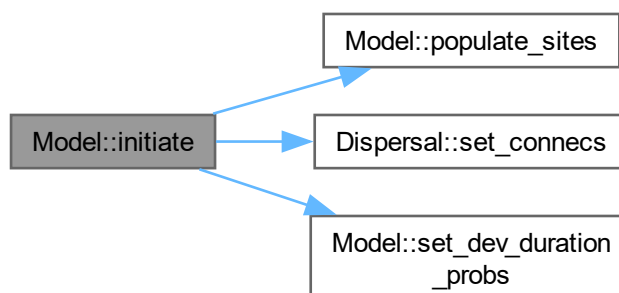
Here is the caller graph for this function:



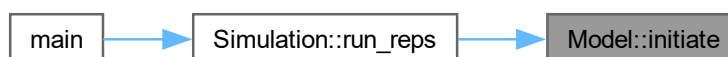
### 6.13.2.8 initiate()

```
void Model::initiate ( )
```

Here is the call graph for this function:



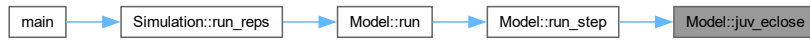
Here is the caller graph for this function:



### 6.13.2.9 juv\_eclose()

```
void Model::juv_eclose ( ) [private]
```

Here is the caller graph for this function:



### 6.13.2.10 juv\_get\_older()

```
void Model::juv_get_older ( ) [private]
```

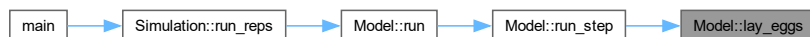
Here is the caller graph for this function:



### 6.13.2.11 lay\_eggs()

```
void Model::lay_eggs (
    const std::array< std::array< std::array< double, num_gen >, num_gen >, num_gen
    > & inher_fraction ) [private]
```

Here is the caller graph for this function:



### 6.13.2.12 populate\_sites()

```
void Model::populate_sites ( ) [private]
```

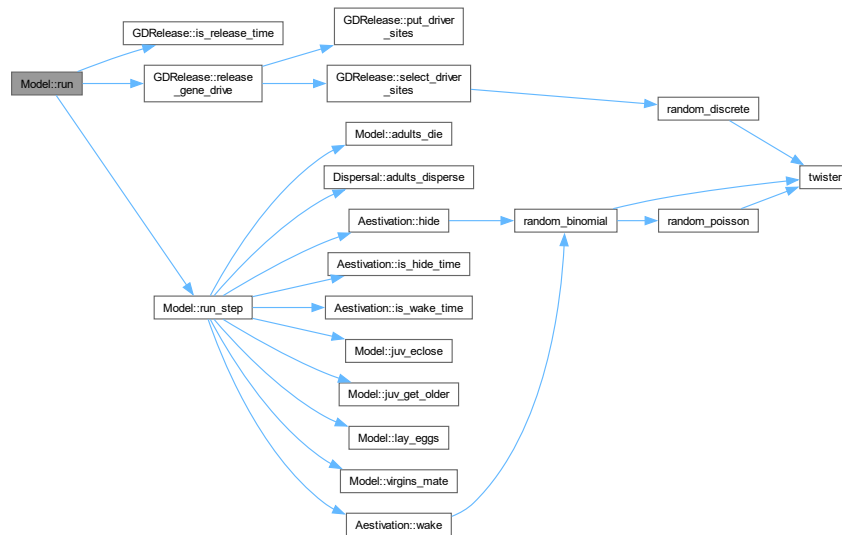
Here is the caller graph for this function:



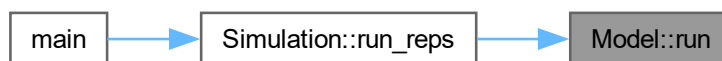
## 6.13.2.13 run()

```
void Model::run (
    int day,
    const std::array< std::array< std::array< double, num_gen >, num_gen >, num_gen
> & inher_fraction )
```

Here is the call graph for this function:



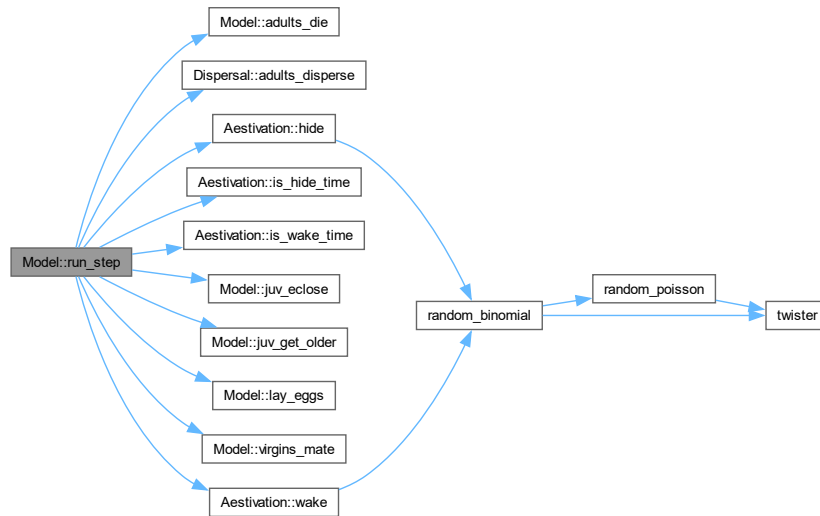
Here is the caller graph for this function:



## 6.13.2.14 run\_step()

```
void Model::run_step (
    int day,
    const std::array< std::array< std::array< double, num_gen >, num_gen >, num_gen
> & inher_fraction ) [private]
```

Here is the call graph for this function:



Here is the caller graph for this function:



#### 6.13.2.15 set\_dev\_duration\_probs()

```

void Model::set_dev_duration_probs (
    int min_time,
    int max_time ) [private]
  
```

Here is the caller graph for this function:

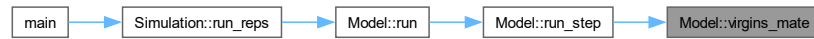




### 6.13.2.16 virgins\_mate()

```
void Model::virgins_mate ( ) [private]
```

Here is the caller graph for this function:



## 6.13.3 Member Data Documentation

### 6.13.3.1 aestivation

```
Aestivation* Model::aestivation [private]
```

### 6.13.3.2 dev\_duration\_probs

```
std::array<double, max_dev+1> Model::dev_duration_probs [private]
```

### 6.13.3.3 dispersal

```
Dispersal* Model::dispersal [private]
```

### 6.13.3.4 gd\_release

```
GDRelease* Model::gd_release [private]
```

### 6.13.3.5 initial\_pops

```
InitialPopsParams* Model::initial_pops [private]
```

### 6.13.3.6 min\_dev

```
int Model::min_dev [private]
```

### 6.13.3.7 num\_pat

```
int Model::num_pat [private]
```

#### 6.13.3.8 side

```
double Model::side [private]
```

#### 6.13.3.9 sites

```
std::vector<Patch*> Model::sites [private]
```

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

- [Model.h](#)
- [Model.cpp](#)

## 6.14 Patch Class Reference

```
#include <Patch.h>
```

Collaboration diagram for Patch:



### Public Member Functions

- [Patch](#) ([LifeParams](#) \*par, double side)
- [Patch](#) ([LifeParams](#) \*par, [Point](#) point)
- void [populate](#) (int initial\_WJ, int initial\_WM, int initial\_WV, int initial\_WF)
- [Point](#) [get\\_coords](#) () const
- std::array< long long int, [num\\_gen](#) > [get\\_M](#) () const

- `std::array< std::array< long long int, num_gen >, num_gen > get_F () const`
- `long long int calculate_tot_J ()`
- `long long int calculate_tot_M ()`
- `long long int calculate_tot_V ()`
- `long long int calculate_tot_F ()`
- `void juv_get_older (int max_dev)`
- `void adults_die ()`
- `void virgins_mate ()`
- `void lay_eggs (const std::array< std::array< std::array< double, num_gen >, num_gen >, num_gen > &f, const std::array< double, max_dev+1 > &dev_duration_probs)`
- `void juv_eclose ()`
- `void update_comp ()`
- `void update_mate ()`
- `void M_disperse_out (const std::array< long long int, num_gen > &m_out)`
- `void F_disperse_out (const std::array< std::array< long long int, num_gen >, num_gen > &f_out)`
- `void M_disperse_in (int gen, long long int m_in)`
- `void F_disperse_in (int f_gen, int m_gen, long long int f_disp)`
- `void F_hide (const std::array< std::array< long long int, num_gen >, num_gen > &f_try)`
- `void F_wake (const std::array< std::array< long long int, num_gen >, num_gen > &f_wake)`
- `void add_driver_M (int num_driver_M)`

### Private Attributes

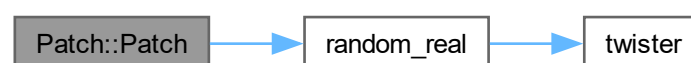
- `LifeParams * params`
- `Point coords`
- `std::array< std::array< long long int, max_dev+1 >, num_gen > J`
- `std::array< long long int, num_gen > M`
- `std::array< long long int, num_gen > V`
- `std::array< std::array< long long int, num_gen >, num_gen > F`
- `long double comp`
- `long double mate_rate`

## 6.14.1 Constructor & Destructor Documentation

### 6.14.1.1 Patch() [1/2]

```
Patch::Patch (
    LifeParams * par,
    double side )
```

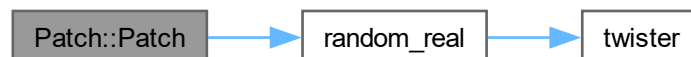
Here is the call graph for this function:



### 6.14.1.2 Patch() [2/2]

```
Patch::Patch (
    LifeParams * par,
    Point point )
```

Here is the call graph for this function:



## 6.14.2 Member Function Documentation

### 6.14.2.1 add\_driver\_M()

```
void Patch::add_driver_M (
    int num_driver_M )
```

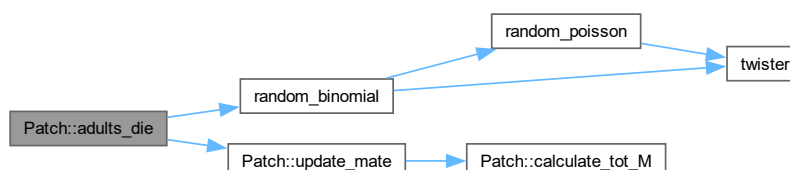
Here is the call graph for this function:



### 6.14.2.2 adults\_die()

```
void Patch::adults_die ( )
```

Here is the call graph for this function:



### 6.14.2.3 calculate\_tot\_F()

```
long long int Patch::calculate_tot_F ( )
```

### 6.14.2.4 calculate\_tot\_J()

```
long long int Patch::calculate_tot_J ( )
```

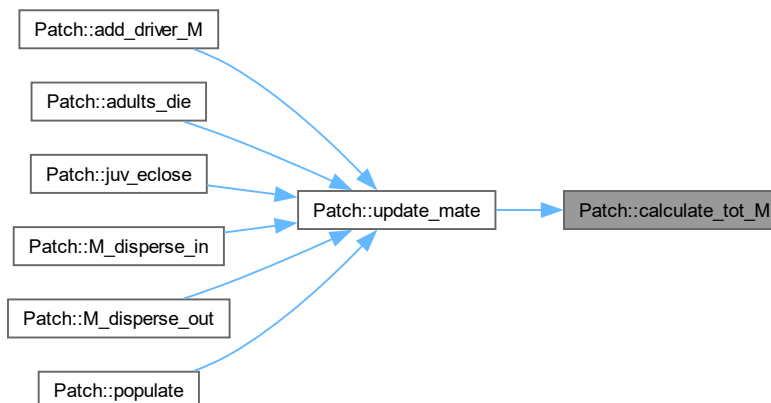
Here is the caller graph for this function:



### 6.14.2.5 calculate\_tot\_M()

```
long long int Patch::calculate_tot_M ( )
```

Here is the caller graph for this function:



### 6.14.2.6 calculate\_tot\_V()

```
long long int Patch::calculate_tot_V ( )
```

**6.14.2.7 F\_disperse\_in()**

```
void Patch::F_disperse_in (
    int f_gen,
    int m_gen,
    long long int f_disp )
```

**6.14.2.8 F\_disperse\_out()**

```
void Patch::F_disperse_out (
    const std::array< std::array< long long int, num_gen >, num_gen > & f_out )
```

**6.14.2.9 F\_hide()**

```
void Patch::F_hide (
    const std::array< std::array< long long int, num_gen >, num_gen > & f_try )
```

**6.14.2.10 F\_wake()**

```
void Patch::F_wake (
    const std::array< std::array< long long int, num_gen >, num_gen > & f_wake )
```

**6.14.2.11 get\_coords()**

```
Point Patch::get_coords ( ) const
```

**6.14.2.12 get\_F()**

```
std::array< std::array< long long int, num_gen >, num_gen > Patch::get_F ( ) const
```

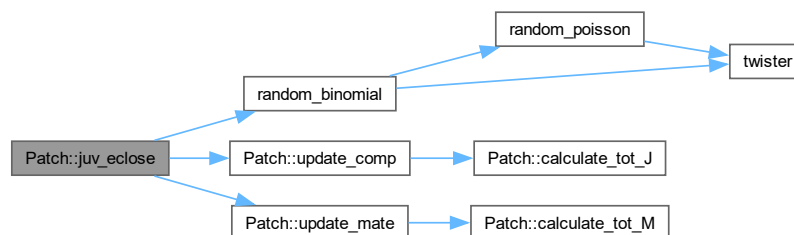
**6.14.2.13 get\_M()**

```
std::array< long long int, num_gen > Patch::get_M ( ) const
```

**6.14.2.14 juv\_eclose()**

```
void Patch::juv_eclose ( )
```

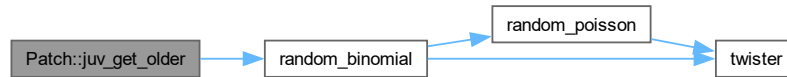
Here is the call graph for this function:



### 6.14.2.15 juv\_get\_older()

```
void Patch::juv_get_older (
    int max_dev )
```

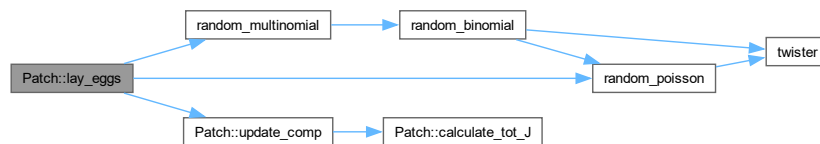
Here is the call graph for this function:



### 6.14.2.16 lay\_eggs()

```
void Patch::lay_eggs (
    const std::array< std::array< std::array< double, num_gen >, num_gen >, num_gen
    > & f,
    const std::array< double, max_dev+1 > & dev_duration_probs )
```

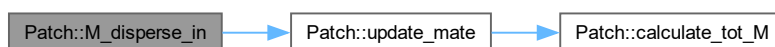
Here is the call graph for this function:



### 6.14.2.17 M\_disperse\_in()

```
void Patch::M_disperse_in (
    int gen,
    long long int m_in )
```

Here is the call graph for this function:

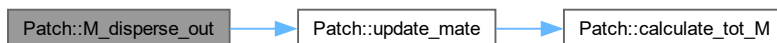




### 6.14.2.18 M\_disperse\_out()

```
void Patch::M_disperse_out (
    const std::array< long long int, num_gen > & m_out )
```

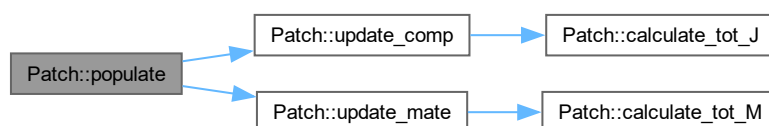
Here is the call graph for this function:



### 6.14.2.19 populate()

```
void Patch::populate (
    int initial_WJ,
    int initial_WM,
    int initial_WV,
    int initial_WF )
```

Here is the call graph for this function:



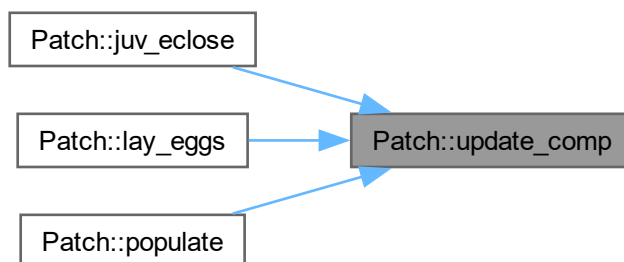
### 6.14.2.20 update\_comp()

```
void Patch::update_comp ( )
```

Here is the call graph for this function:



Here is the caller graph for this function:



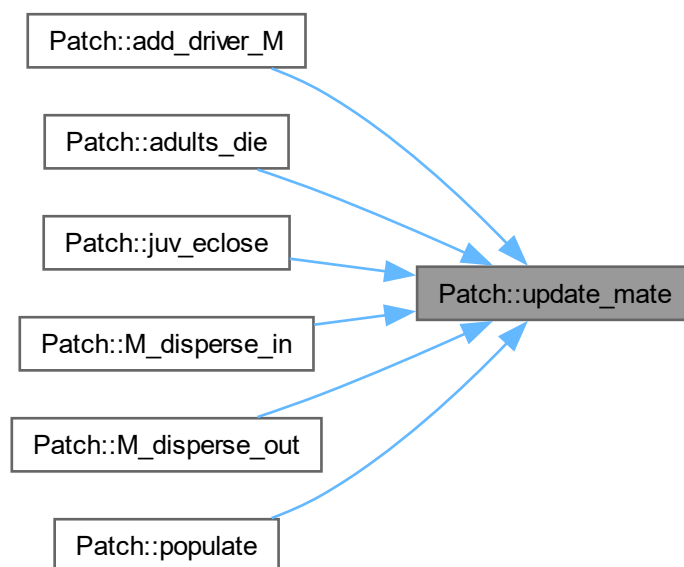
#### 6.14.2.21 `update_mate()`

```
void Patch::update_mate ( )
```

Here is the call graph for this function:



Here is the caller graph for this function:



#### 6.14.2.22 virgins\_mate()

```
void Patch::virgins_mate ( )
```

Here is the call graph for this function:



### 6.14.3 Member Data Documentation

#### 6.14.3.1 comp

```
long double Patch::comp [private]
```

#### 6.14.3.2 coords

```
Point Patch::coords [private]
```

#### 6.14.3.3 F

```
std::array<std::array<long long int, num_gen>, num_gen> Patch::F [private]
```

#### 6.14.3.4 J

```
std::array<std::array<long long int, max_dev+1>, num_gen> Patch::J [private]
```

#### 6.14.3.5 M

```
std::array<long long int, num_gen> Patch::M [private]
```

#### 6.14.3.6 mate\_rate

```
long double Patch::mate_rate [private]
```

#### 6.14.3.7 params

```
LifeParams* Patch::params [private]
```

#### 6.14.3.8 V

```
std::array<long long int, num_gen> Patch::V [private]
```

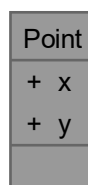
The documentation for this class was generated from the following files:

- [Patch.h](#)
- [Patch.cpp](#)

## 6.15 Point Struct Reference

```
#include <Point.h>
```

Collaboration diagram for Point:



### Public Attributes

- double [x](#) = 0
- double [y](#) = 0

## 6.15.1 Member Data Documentation

### 6.15.1.1 x

```
double Point::x = 0
```

### 6.15.1.2 y

```
double Point::y = 0
```

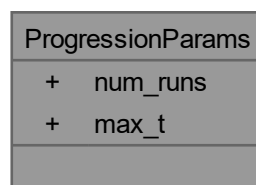
The documentation for this struct was generated from the following file:

- [Point.h](#)

## 6.16 ProgressionParams Struct Reference

```
#include <Params.h>
```

Collaboration diagram for ProgressionParams:



### Public Attributes

- int [num\\_runs](#) = 1
- int [max\\_t](#) = 1000

## 6.16.1 Member Data Documentation

### 6.16.1.1 max\_t

```
int ProgressionParams::max_t = 1000
```

### 6.16.1.2 num\_runs

```
int ProgressionParams::num_runs = 1
```

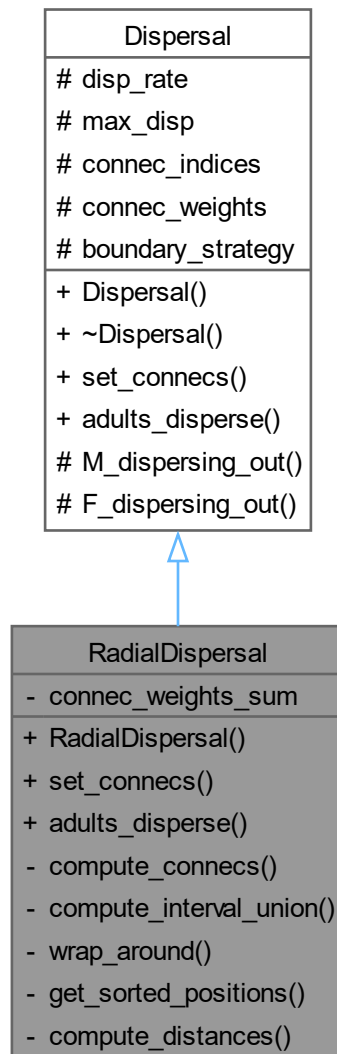
The documentation for this struct was generated from the following file:

- [Params.h](#)

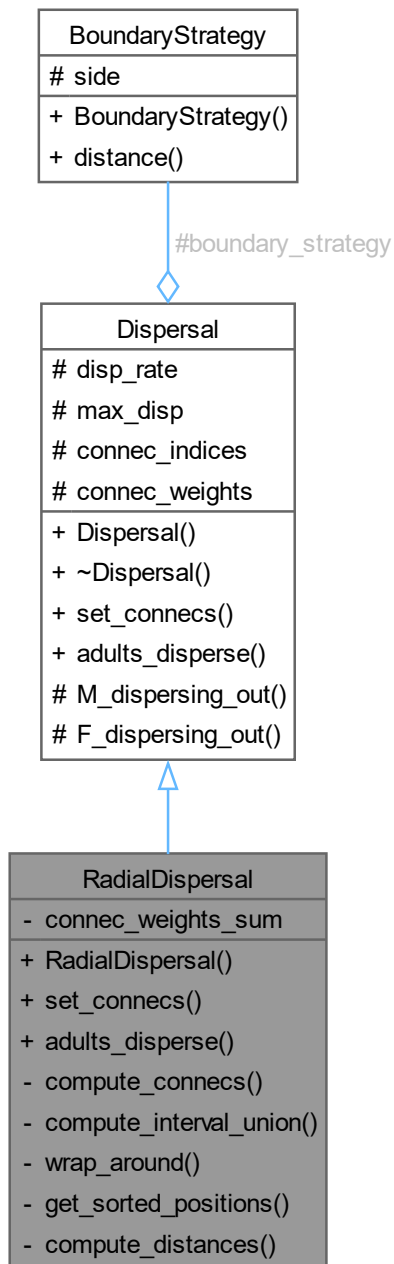
## 6.17 RadialDispersal Class Reference

```
#include <Dispersal.h>
```

Inheritance diagram for RadialDispersal:



Collaboration diagram for RadialDispersal:



### Public Member Functions

- [RadialDispersal](#) ([DispersalParams](#) \*params, [BoundaryType](#) boundary, double side)
- void [set\\_connecs](#) (std::vector< [Patch](#) \* > &sites) override
- void [adults\\_disperse](#) (std::vector< [Patch](#) \* > &sites) override

## Public Member Functions inherited from [Dispersal](#)

- [Dispersal](#) ([DispersalParams](#) \*params, [BoundaryType](#) boundary, double side)
- [~Dispersal](#) ()

## Private Member Functions

- `std::pair< std::vector< std::vector< int > >, std::vector< std::vector< double > > >` [compute\\_connecs](#) (`std::vector< Patch * >` &sites)
- `std::pair< std::vector< std::pair< double, double > >, double >` [compute\\_interval\\_union](#) (const `std::pair< double, double >` &qq, const `std::vector< std::pair< double, double > >` &input)
- double [wrap\\_around](#) (double value, double range)
- `std::vector< int >` [get\\_sorted\\_positions](#) (const `std::vector< double >` &numbers)
- `std::vector< std::vector< double > >` [compute\\_distances](#) (const `std::vector< Patch * >` &sites)

## Private Attributes

- `std::vector< double >` [connec\\_weights\\_sum](#)

## Additional Inherited Members

## Protected Member Functions inherited from [Dispersal](#)

- `std::vector< std::array< long long int, num\_gen > >` [M\\_dispersing\\_out](#) (const `std::vector< Patch * >` &sites)
- `std::vector< std::array< std::array< long long int, num\_gen >, num\_gen > >` [F\\_dispersing\\_out](#) (const `std::vector< Patch * >` &sites)

## Protected Attributes inherited from [Dispersal](#)

- double [disp\\_rate](#)
- double [max\\_disp](#)
- `std::vector< std::vector< int > >` [connec\\_indices](#)
- `std::vector< std::vector< double > >` [connec\\_weights](#)
- [BoundaryStrategy](#) \* [boundary\\_strategy](#)

## 6.17.1 Constructor & Destructor Documentation

### 6.17.1.1 RadialDispersal()

```
RadialDispersal::RadialDispersal (
    DispersalParams * params,
    BoundaryType boundary,
    double side )
```



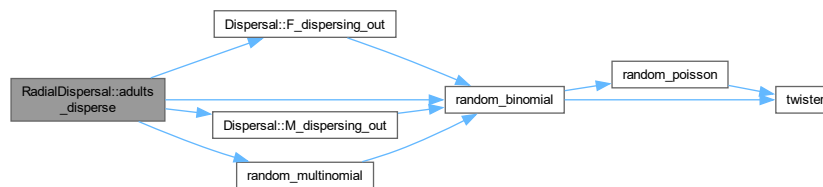
## 6.17.2 Member Function Documentation

### 6.17.2.1 adults\_disperse()

```
void RadialDispersal::adults_disperse (
    std::vector< Patch * > & sites ) [override], [virtual]
```

Implements [Dispersal](#).

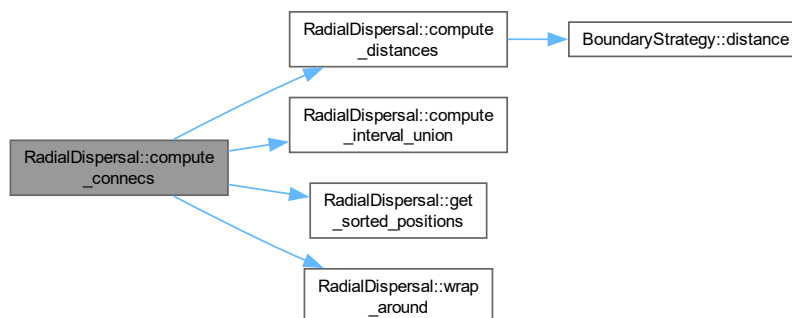
Here is the call graph for this function:



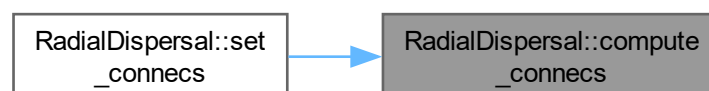
### 6.17.2.2 compute\_connects()

```
std::pair< std::vector< std::vector< int > >, std::vector< std::vector< double > > > > RadialDispersal::compute_connects (
    std::vector< Patch * > & sites ) [private]
```

Here is the call graph for this function:



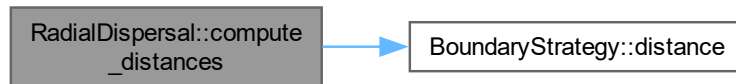
Here is the caller graph for this function:



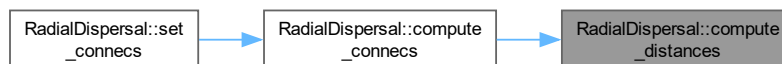
### 6.17.2.3 compute\_distances()

```
std::vector< std::vector< double > > RadialDispersal::compute_distances (
    const std::vector< Patch * > & sites ) [private]
```

Here is the call graph for this function:



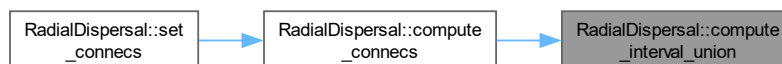
Here is the caller graph for this function:



### 6.17.2.4 compute\_interval\_union()

```
std::pair< std::vector< std::pair< double, double > >, double > RadialDispersal::compute_↵
interval_union (
    const std::pair< double, double > & qq,
    const std::vector< std::pair< double, double > > & input ) [private]
```

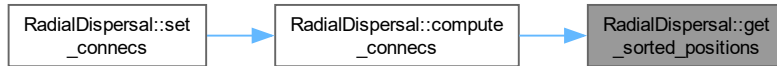
Here is the caller graph for this function:



### 6.17.2.5 get\_sorted\_positions()

```
std::vector< int > RadialDispersal::get_sorted_positions (
    const std::vector< double > & numbers ) [private]
```

Here is the caller graph for this function:

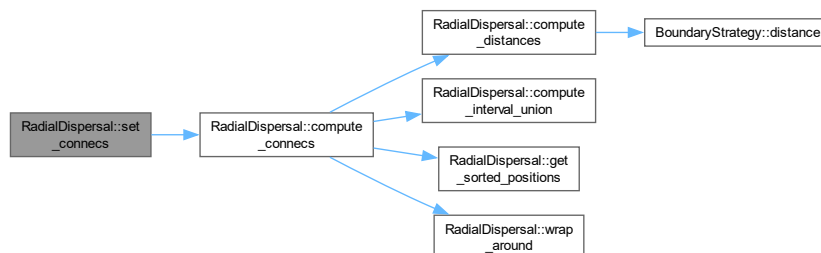


### 6.17.2.6 set\_connecs()

```
void RadialDispersal::set_connecs (
    std::vector< Patch * > & sites ) [override], [virtual]
```

Implements [Dispersal](#).

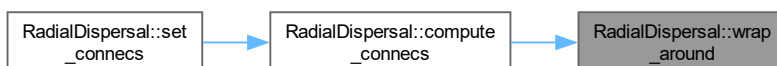
Here is the call graph for this function:



### 6.17.2.7 wrap\_around()

```
double RadialDispersal::wrap_around (
    double value,
    double range ) [private]
```

Here is the caller graph for this function:



### 6.17.3 Member Data Documentation

#### 6.17.3.1 connec\_weights\_sum

```
std::vector<double> RadialDispersal::connec_weights_sum [private]
```

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

- [Dispersal.h](#)
- [Dispersal.cpp](#)

## 6.18 Record Class Reference

```
#include <Record.h>
```

Collaboration diagram for Record:

Record
<ul style="list-style-type: none"><li>- rec_start</li><li>- rec_end</li><li>- rec_interval_global</li><li>- rec_interval_local</li><li>- rec_sites_freq</li><li>- set_label</li><li>- rep_label</li><li>- os1</li><li>- os2</li><li>- os3</li><li>- local_data</li><li>- global_data</li><li>- coord_list</li></ul>
<ul style="list-style-type: none"><li>+ Record()</li><li>+ ~Record()</li><li>+ record_coords()</li><li>+ record_global()</li><li>+ output_totals()</li><li>+ record_local()</li><li>+ is_rec_local_time()</li><li>+ is_rec_global_time()</li></ul>

**Public Member Functions**

- [Record](#) ([RecordParams](#) \*rec\_params, int rep)
- [~Record](#) ()
- void [record\\_coords](#) (const std::vector< [Patch](#) \* > &sites)
- void [record\\_global](#) (int day, const std::array< long long int, [num\\_gen](#) > &tot\_M\_gen)
- void [output\\_totals](#) (int day, long long int tot\_J, long long int tot\_M, long long int tot\_V, long long int tot\_F)
- void [record\\_local](#) (int day, const std::vector< [Patch](#) \* > &sites)
- bool [is\\_rec\\_local\\_time](#) (int day)
- bool [is\\_rec\\_global\\_time](#) (int day)

**Private Attributes**

- int [rec\\_start](#)
- int [rec\\_end](#)
- int [rec\\_interval\\_global](#)
- int [rec\\_interval\\_local](#)
- int [rec\\_sites\\_freq](#)
- int [set\\_label](#)
- int [rep\\_label](#)
- std::ostringstream [os1](#)
- std::ostringstream [os2](#)
- std::ostringstream [os3](#)
- std::ofstream [local\\_data](#)
- std::ofstream [global\\_data](#)
- std::ofstream [coord\\_list](#)

**6.18.1 Constructor & Destructor Documentation****6.18.1.1 Record()**

```
Record::Record (
    RecordParams * rec_params,
    int rep )
```

**6.18.1.2 ~Record()**

```
Record::~Record ( )
```

**6.18.2 Member Function Documentation****6.18.2.1 is\_rec\_global\_time()**

```
bool Record::is_rec_global_time (
    int day )
```

Here is the caller graph for this function:



### 6.18.2.2 is\_rec\_local\_time()

```
bool Record::is_rec_local_time (
    int day )
```

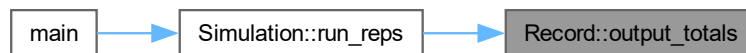
Here is the caller graph for this function:



### 6.18.2.3 output\_totals()

```
void Record::output_totals (
    int day,
    long long int tot_J,
    long long int tot_M,
    long long int tot_V,
    long long int tot_F )
```

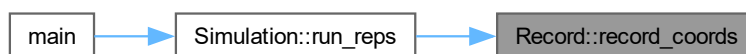
Here is the caller graph for this function:



### 6.18.2.4 record\_coords()

```
void Record::record_coords (
    const std::vector< Patch * > & sites )
```

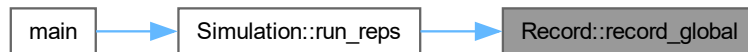
Here is the caller graph for this function:



### 6.18.2.5 record\_global()

```
void Record::record_global (
    int day,
    const std::array< long long int, num_gen > & tot_M_gen )
```

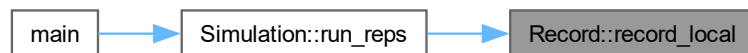
Here is the caller graph for this function:



### 6.18.2.6 record\_local()

```
void Record::record_local (
    int day,
    const std::vector< Patch * > & sites )
```

Here is the caller graph for this function:



## 6.18.3 Member Data Documentation

### 6.18.3.1 coord\_list

```
std::ofstream Record::coord_list [private]
```

### 6.18.3.2 global\_data

```
std::ofstream Record::global_data [private]
```

### 6.18.3.3 local\_data

```
std::ofstream Record::local_data [private]
```

**6.18.3.4 os1**

```
std::ostream Record::os1 [private]
```

**6.18.3.5 os2**

```
std::ostream Record::os2 [private]
```

**6.18.3.6 os3**

```
std::ostream Record::os3 [private]
```

**6.18.3.7 rec\_end**

```
int Record::rec_end [private]
```

**6.18.3.8 rec\_interval\_global**

```
int Record::rec_interval_global [private]
```

**6.18.3.9 rec\_interval\_local**

```
int Record::rec_interval_local [private]
```

**6.18.3.10 rec\_sites\_freq**

```
int Record::rec_sites_freq [private]
```

**6.18.3.11 rec\_start**

```
int Record::rec_start [private]
```

**6.18.3.12 rep\_label**

```
int Record::rep_label [private]
```

**6.18.3.13 set\_label**

```
int Record::set_label [private]
```

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

- [Record.h](#)
- [Record.cpp](#)



## 6.19 RecordParams Struct Reference

```
#include <Params.h>
```

Collaboration diagram for RecordParams:



### Public Attributes

- int [rec\\_start](#) = 0
- int [rec\\_end](#) = 1000
- int [rec\\_interval\\_global](#) = 1
- int [rec\\_interval\\_local](#) = 200
- int [rec\\_sites\\_freq](#) = 1
- int [set\\_label](#) = 1

### 6.19.1 Member Data Documentation

#### 6.19.1.1 [rec\\_end](#)

```
int RecordParams::rec_end = 1000
```

#### 6.19.1.2 [rec\\_interval\\_global](#)

```
int RecordParams::rec_interval_global = 1
```

#### 6.19.1.3 [rec\\_interval\\_local](#)

```
int RecordParams::rec_interval_local = 200
```

#### 6.19.1.4 rec\_sites\_freq

```
int RecordParams::rec_sites_freq = 1
```

#### 6.19.1.5 rec\_start

```
int RecordParams::rec_start = 0
```

#### 6.19.1.6 set\_label

```
int RecordParams::set_label = 1
```

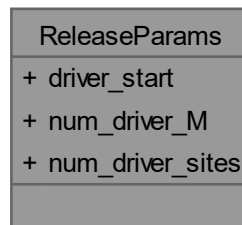
The documentation for this struct was generated from the following file:

- [Params.h](#)

## 6.20 ReleaseParams Struct Reference

```
#include <Params.h>
```

Collaboration diagram for ReleaseParams:



### Public Attributes

- int [driver\\_start](#) = 200
- int [num\\_driver\\_M](#) = 1000
- int [num\\_driver\\_sites](#) = 5

### 6.20.1 Member Data Documentation

#### 6.20.1.1 driver\_start

```
int ReleaseParams::driver_start = 200
```

### 6.20.1.2 num\_driver\_M

```
int ReleaseParams::num_driver_M = 1000
```

### 6.20.1.3 num\_driver\_sites

```
int ReleaseParams::num_driver_sites = 5
```

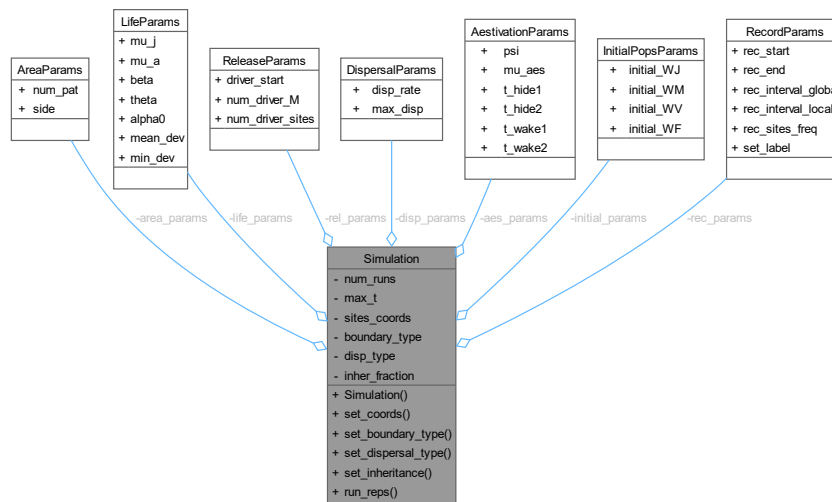
The documentation for this struct was generated from the following file:

- [Params.h](#)

## 6.21 Simulation Class Reference

```
#include <Simulation.h>
```

Collaboration diagram for Simulation:



### Public Member Functions

- [Simulation](#) ([ProgressionParams](#) &prog, [AreaParams](#) &area, [LifeParams](#) &life, [ReleaseParams](#) &rel, [DispersalParams](#) &disp, [AestivationParams](#) &aes, [InitialPopsParams](#) &initial, [RecordParams](#) &rec)
- void [set\\_coords](#) (const std::string &coords\_file)
- void [set\\_boundary\\_type](#) ([BoundaryType](#) boundary)
- void [set\\_dispersal\\_type](#) ([DispersalType](#) disp)
- void [set\\_inheritance](#) ([InheritanceParams](#) inher\_params)
- void [run\\_reps](#) ()

## Private Attributes

- int `num_runs`
- int `max_t`
- `AreaParams` \* `area_params`
- `LifeParams` \* `life_params`
- `ReleaseParams` \* `rel_params`
- `DispersalParams` \* `disp_params`
- `AestivationParams` \* `aes_params`
- `InitialPopsParams` \* `initial_params`
- `RecordParams` \* `rec_params`
- `std::vector< Point >` `sites_coords`
- `BoundaryType` `boundary_type`
- `DispersalType` `disp_type`
- `std::array< std::array< std::array< double, num_gen >, num_gen >, num_gen >` `inher_fraction`

## 6.21.1 Constructor & Destructor Documentation

### 6.21.1.1 Simulation()

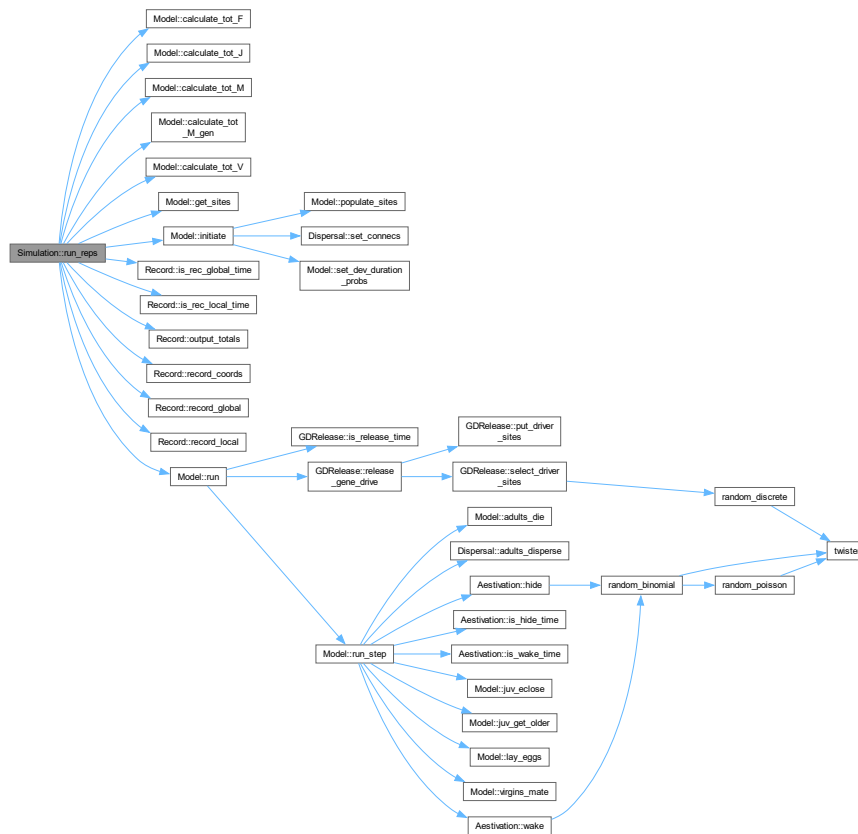
```
Simulation::Simulation (
    ProgressionParams & prog,
    AreaParams & area,
    LifeParams & life,
    ReleaseParams & rel,
    DispersalParams & disp,
    AestivationParams & aes,
    InitialPopsParams & initial,
    RecordParams & rec )
```

## 6.21.2 Member Function Documentation

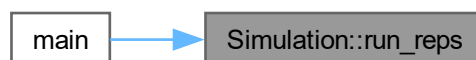
### 6.21.2.1 run\_reps()

```
void Simulation::run_reps ( )
```

Here is the call graph for this function:



Here is the caller graph for this function:

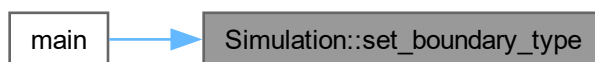


### 6.21.2.2 set\_boundary\_type()

```

void Simulation::set_boundary_type (
    BoundaryType boundary )
  
```

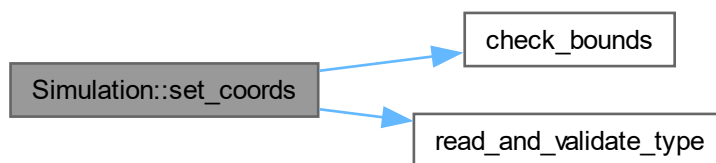
Here is the caller graph for this function:



### 6.21.2.3 set\_coords()

```
void Simulation::set_coords (
    const std::string & coords_file )
```

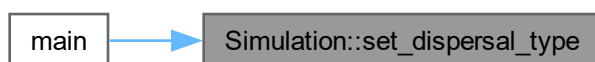
Here is the call graph for this function:



### 6.21.2.4 set\_dispersal\_type()

```
void Simulation::set_dispersal_type (
    DispersalType disp )
```

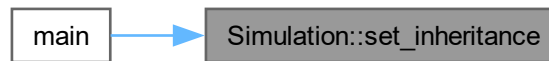
Here is the caller graph for this function:



### 6.21.2.5 set\_inheritance()

```
void Simulation::set_inheritance (
    InheritanceParams inher_params )
```

Here is the caller graph for this function:



## 6.21.3 Member Data Documentation

### 6.21.3.1 aes\_params

```
AestivationParams* Simulation::aes_params [private]
```

### 6.21.3.2 area\_params

```
AreaParams* Simulation::area_params [private]
```

### 6.21.3.3 boundary\_type

```
BoundaryType Simulation::boundary_type [private]
```

### 6.21.3.4 disp\_params

```
DispersalParams* Simulation::disp_params [private]
```

### 6.21.3.5 disp\_type

```
DispersalType Simulation::disp_type [private]
```

### 6.21.3.6 inher\_fraction

```
std::array<std::array<std::array<double, num_gen>, num_gen>, num_gen> Simulation::inher_↵
fraction [private]
```

#### 6.21.3.7 initial\_params

`InitialPopsParams*` `Simulation::initial_params` [private]

#### 6.21.3.8 life\_params

`LifeParams*` `Simulation::life_params` [private]

#### 6.21.3.9 max\_t

`int` `Simulation::max_t` [private]

#### 6.21.3.10 num\_runs

`int` `Simulation::num_runs` [private]

#### 6.21.3.11 rec\_params

`RecordParams*` `Simulation::rec_params` [private]

#### 6.21.3.12 rel\_params

`ReleaseParams*` `Simulation::rel_params` [private]

#### 6.21.3.13 sites\_coords

`std::vector<Point>` `Simulation::sites_coords` [private]

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

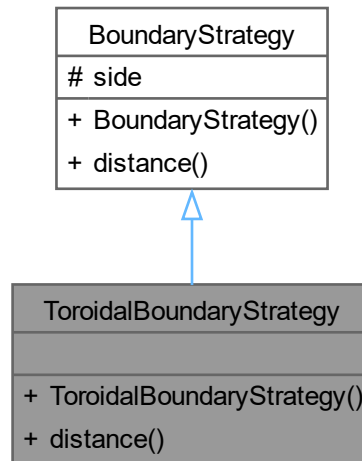
- [Simulation.h](#)
- [Simulation.cpp](#)



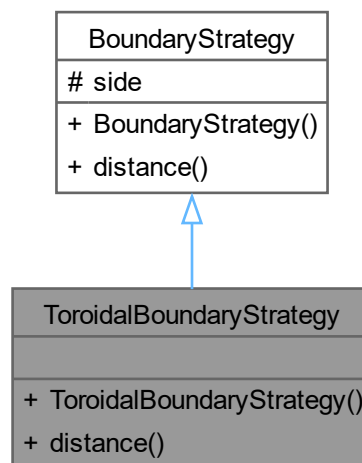
## 6.22 ToroidalBoundaryStrategy Class Reference

```
#include <BoundaryStrategy.h>
```

Inheritance diagram for ToroidalBoundaryStrategy:



Collaboration diagram for ToroidalBoundaryStrategy:



### Public Member Functions

- `ToroidalBoundaryStrategy` (double `side`)
- double `distance` (const `Point` &p1, const `Point` &p2) override

## Public Member Functions inherited from [BoundaryStrategy](#)

- [BoundaryStrategy](#) (double [side](#))

## Additional Inherited Members

## Protected Attributes inherited from [BoundaryStrategy](#)

- double [side](#)

## 6.22.1 Constructor & Destructor Documentation

### 6.22.1.1 [ToroidalBoundaryStrategy](#)()

```
ToroidalBoundaryStrategy::ToroidalBoundaryStrategy (  
    double side ) [inline]
```

## 6.22.2 Member Function Documentation

### 6.22.2.1 [distance](#)()

```
double ToroidalBoundaryStrategy::distance (  
    const Point & p1,  
    const Point & p2 ) [override], [virtual]
```

Implements [BoundaryStrategy](#).

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

- [BoundaryStrategy.h](#)
- [BoundaryStrategy.cpp](#)

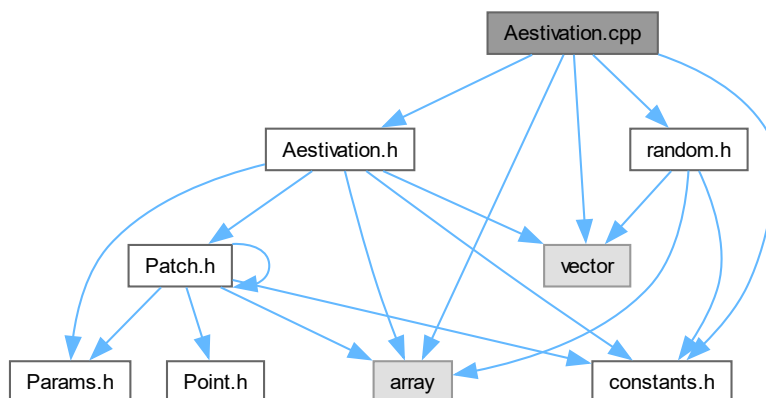
## Chapter 7

# File Documentation

### 7.1 Aestivation.cpp File Reference

```
#include <array>
#include <vector>
#include "Aestivation.h"
#include "random.h"
#include "constants.h"
```

Include dependency graph for Aestivation.cpp:

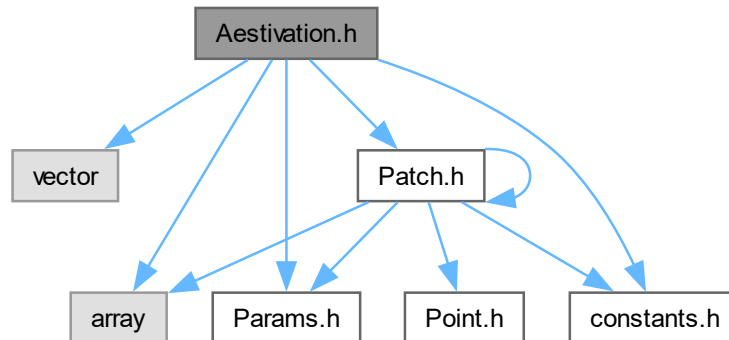


### 7.2 Aestivation.h File Reference

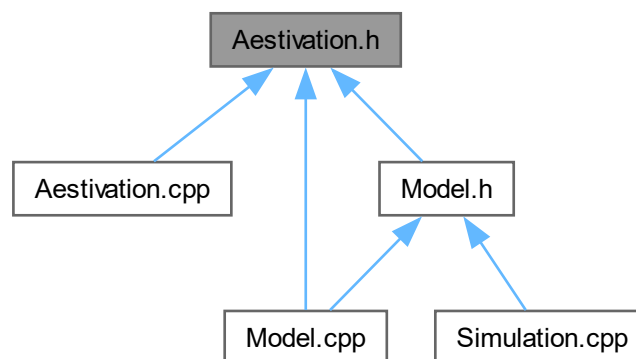
```
#include <vector>
#include <array>
#include "Params.h"
#include "Patch.h"
```

```
#include "constants.h"
```

Include dependency graph for Aestivation.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class [Aestivation](#)

## 7.3 Aestivation.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AESTIVATION_H
00002 #define AESTIVATION_H
00003
00004 #include <vector>

```

```

00005 #include <array>
00006 #include "Params.h"
00007 #include "Patch.h"
00008 #include "constants.h"
00009
00010 using namespace constants;
00011
00012 class Patch;
00013
00014 // Implement aestivative behaviour for the collection of mosquito sites.
00015 class Aestivation {
00016 public:
00017     Aestivation(AestivationParams *params, int sites_size);
00018     void hide(std::vector<Patch*> &sites);
00019     void wake(int day, std::vector<Patch*> &sites);
00020     bool is_hide_time(int day);
00021     bool is_wake_time(int day);
00022
00023 private:
00024     double psi; // aestivation rate
00025     double mu_aes; // aestivation mortality
00026     int t_hide1; // start day of aestivation-entering period (day number of the year), not included
00027     int t_hide2; // end day of aestivation-entering period (day number of the year)
00028     int t_wake1; // start day of aestivation-waking period (day number of the year), not included
00029     int t_wake2; // end day of aestivation-waking period (day number of the year)
00030
00031     // number of mated female mosquitoes F_{ij} with female genotype i and carrying mated male
    genotype j that have gone into
00032     // aestivation from each patch
00033     std::vector<std::array<std::array<long long int, num_gen>, num_gen>> aes_F;
00034 };
00035
00036 #endif //AESTIVATION_H

```

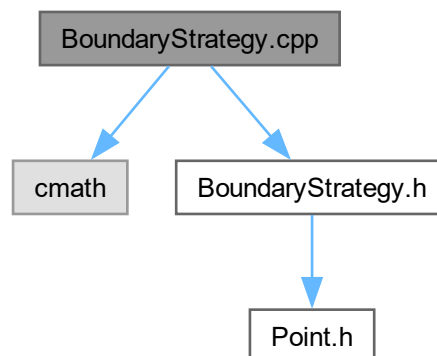
## 7.4 BoundaryStrategy.cpp File Reference

```

#include <cmath>
#include "BoundaryStrategy.h"

```

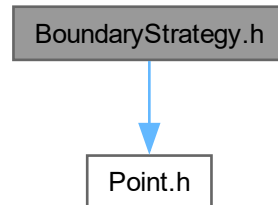
Include dependency graph for BoundaryStrategy.cpp:



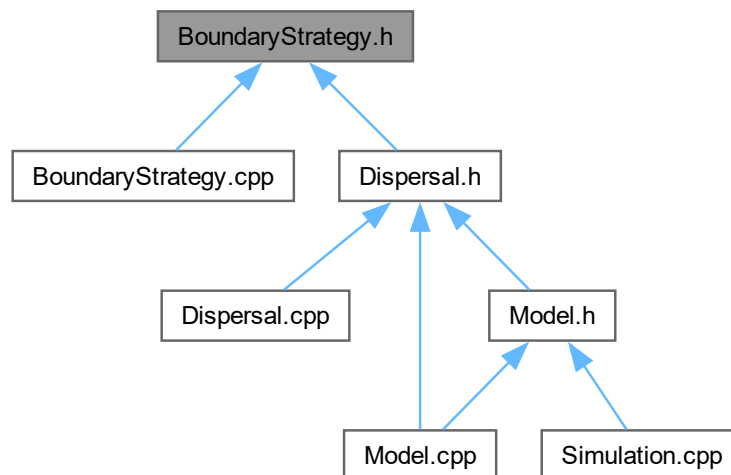
## 7.5 BoundaryStrategy.h File Reference

```
#include "Point.h"
```

Include dependency graph for BoundaryStrategy.h:



This graph shows which files directly or indirectly include this file:



### Classes

- class [BoundaryStrategy](#)
- class [ToroidalBoundaryStrategy](#)
- class [EdgeBoundaryStrategy](#)

## 7.6 BoundaryStrategy.h

[Go to the documentation of this file.](#)

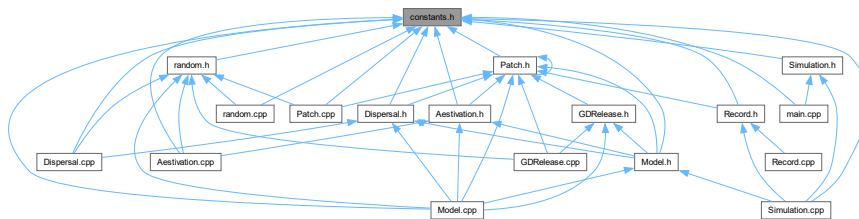
```

00001 #ifndef BOUNDARYSTRATEGY_H
00002 #define BOUNDARYSTRATEGY_H
00003
00004 #include "Point.h"
00005
00006 class BoundaryStrategy {
00007 public:
00008     BoundaryStrategy(double side): side(side) {};
00009     virtual double distance(const Point& p1, const Point& p2) = 0;
00010
00011 protected:
00012     double side;
00013 };
00014
00015 class ToroidalBoundaryStrategy: public BoundaryStrategy {
00016 public:
00017     ToroidalBoundaryStrategy(double side): BoundaryStrategy(side) {};
00018     double distance(const Point &p1, const Point &p2) override;
00019 };
00020
00021 class EdgeBoundaryStrategy: public BoundaryStrategy {
00022 public:
00023     EdgeBoundaryStrategy(double side): BoundaryStrategy(side) {};
00024     double distance(const Point& p1, const Point& p2) override;
00025 };
00026
00027 #endif //BOUNDARYSTRATEGY_H

```

## 7.7 constants.h File Reference

This graph shows which files directly or indirectly include this file:



### Namespaces

- namespace [constants](#)

### Enumerations

- enum [BoundaryType](#) { [Toroid](#) , [Edge](#) }
- enum [DispersalType](#) { [DistanceKernel](#) , [Radial](#) }

### Variables

- const int [constants::max\\_dev](#) = 20
- const int [constants::num\\_gen](#) = 6

## 7.7.1 Enumeration Type Documentation

### 7.7.1.1 BoundaryType

enum [BoundaryType](#)

**Enumerator**

Toroid	
Edge	

**7.7.1.2 DispersalType**

```
enum DispersalType
```

**Enumerator**

DistanceKernel	
Radial	

**7.8 constants.h**

[Go to the documentation of this file.](#)

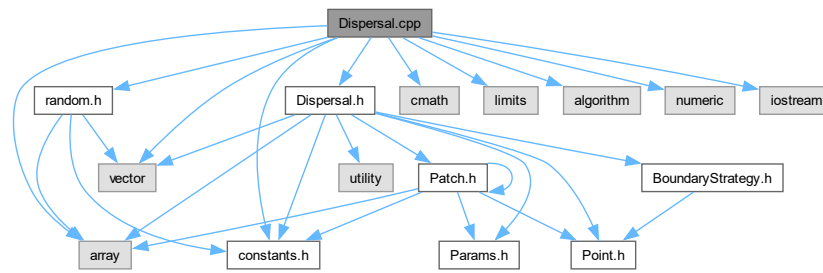
```
00001
00002 #ifndef CONSTANTS_H
00003 #define CONSTANTS_H
00004
00005 namespace constants
00006 {
00007     const int max_dev = 20; // juvenile development time (egg to adult) expressed as days left till
        eclosion (eclosion on day 0)
00008     const int num_gen = 6; // number of different genotypes in the mosquito population
00009 }
00010
00011 enum BoundaryType{Toroid, Edge};
00012 enum DispersalType{DistanceKernel, Radial};
00013
00014 #endif //CONSTANTS_H
```

**7.9 Dispersal.cpp File Reference**

```
#include <vector>
#include <array>
#include <cmath>
#include <limits>
#include <algorithm>
#include <numeric>
#include "Dispersal.h"
#include "random.h"
#include "constants.h"
#include <iostream>
```



Include dependency graph for Dispersal.cpp:



## Variables

- const double `PI` = 3.14159265
- const double `TWOPI` = 2 \* `PI`

## 7.9.1 Variable Documentation

### 7.9.1.1 `PI`

```
const double PI = 3.14159265
```

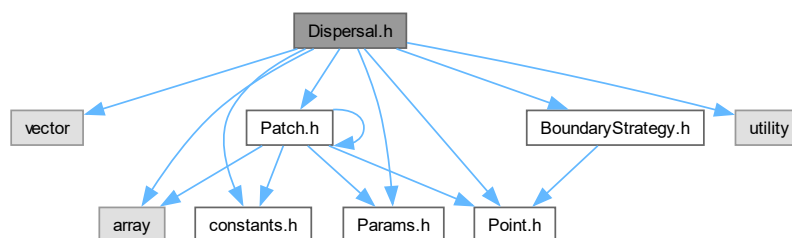
### 7.9.1.2 `TWOPI`

```
const double TWOPI = 2 * PI
```

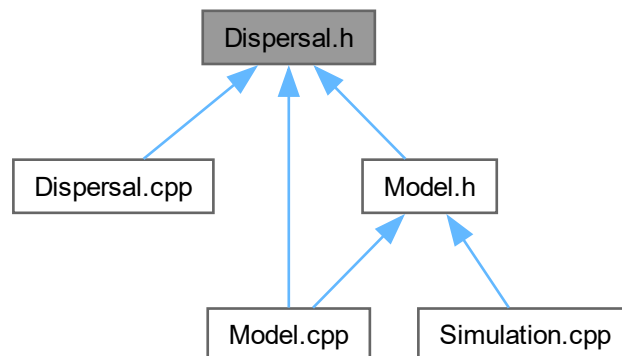
## 7.10 Dispersal.h File Reference

```
#include <vector>
#include <array>
#include <utility>
#include "constants.h"
#include "Params.h"
#include "Patch.h"
#include "Point.h"
#include "BoundaryStrategy.h"
```

Include dependency graph for Dispersal.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class [Dispersal](#)
- class [DistanceKernelDispersal](#)
- class [RadialDispersal](#)

## 7.11 Dispersal.h

[Go to the documentation of this file.](#)

```

00001 #ifndef DISPERSAL_H
00002 #define DISPERSAL_H
00003
00004 #include <vector>
00005 #include <array>
00006 #include <utility>
00007 #include "constants.h"
00008 #include "Params.h"
00009 #include "Patch.h"
00010 #include "Point.h"
00011 #include "BoundaryStrategy.h"
00012
00013 using namespace constants;
00014
00015 class Patch;
00016
00017 // Implements dispersion across all mosquito sites in the collection.
00018 class Dispersal {
00019 public:
00020     Dispersal(DispersalParams* params, BoundaryType boundary, double side);
00021     ~Dispersal();
00022     virtual void set_connects(std::vector<Patch*> &sites) = 0;
00023     virtual void adults_disperse(std::vector<Patch*> &sites) = 0;
00024
00025 protected:
00026     double disp_rate; // adult dispersal rate
00027     double max_disp; // maximum distance at which two sites are connected (km)
00028
00029     // connected patch indices ordered by each patch in sites, such that the first element contains
00030     // the indices of all the patches
00031     // connected to the first sites patch, second element has all connection indices to the second
00032     // sites patch, etc.
00033     std::vector<std::vector<int>> connec_indices;
00034     // connection weights of the connected patches ordered by each patch in sites, such that the first
00035     // element contains the connection
  
```

```

00033     // weights between the first patch in sites and all the patches connected to it, the second
00034     // element has all connection weights
00035     // between the second sites element and all other patches connected to it, etc.
00036     std::vector<std::vector<double>> connec_weights;
00037     BoundaryStrategy* boundary_strategy;
00038     std::vector<std::array<long long int, num_gen>> M_dispersing_out(const std::vector<Patch*> &sites);
00039     std::vector<std::array<long long int, num_gen>, num_gen>> F_dispersing_out(const
00040     std::vector<Patch*> &sites);
00041 };
00042
00043
00044 class DistanceKernelDispersal: public Dispersal {
00045 public:
00046     DistanceKernelDispersal(DispersalParams* params, BoundaryType boundary, double side):
00047     Dispersal(params, boundary, side) {};
00048     void set_connecs(std::vector<Patch*> &sites) override;
00049     void adults_disperse(std::vector<Patch*> &sites) override;
00050 private:
00051     std::pair<std::vector<std::vector<int>>, std::vector<std::vector<double>>>
00052     compute_connecs(std::vector<Patch*> &sites);
00053 };
00054
00055 class RadialDispersal: public Dispersal {
00056 public:
00057     RadialDispersal(DispersalParams* params, BoundaryType boundary, double side);
00058     void set_connecs(std::vector<Patch*> &sites) override;
00059     void adults_disperse(std::vector<Patch*> &sites) override;
00060 private:
00061     std::vector<double> connec_weights_sum;
00062
00063     std::pair<std::vector<std::vector<int>>, std::vector<std::vector<double>>>
00064     compute_connecs(std::vector<Patch*> &sites);
00065     std::pair<std::vector<std::pair<double, double>>, double> compute_interval_union(const
00066     std::pair<double, double>& qq,
00067     const std::vector<std::pair<double, double>& input);
00068     double wrap_around(double value, double range);
00069     std::vector<int> get_sorted_positions(const std::vector<double>& numbers);
00070     std::vector<std::vector<double>> compute_distances(const std::vector<Patch*> &sites);
00071 };
00072 #endif //DISPERSAL_H

```

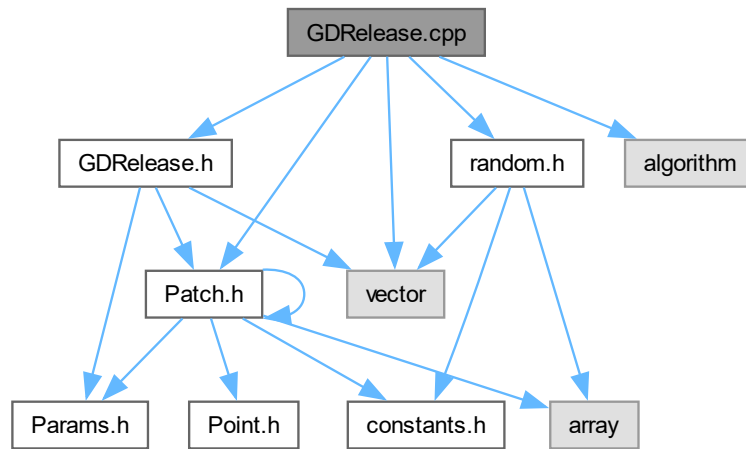
## 7.12 GDRelease.cpp File Reference

```

#include <vector>
#include <algorithm>
#include "GDRelease.h"
#include "Patch.h"
#include "random.h"

```

Include dependency graph for GDRelease.cpp:



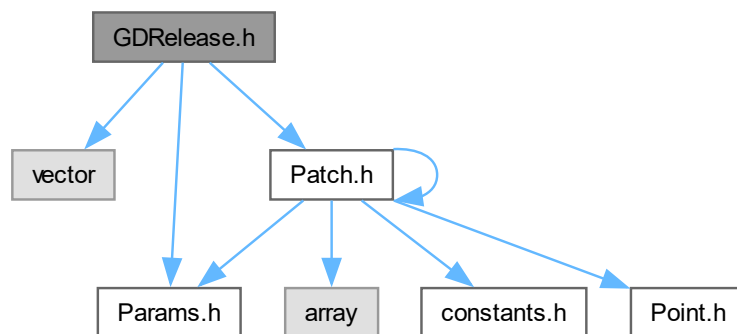
## 7.13 GDRelease.h File Reference

```

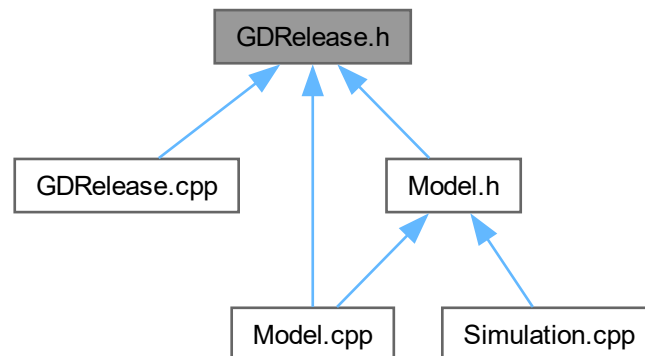
#include <vector>
#include "Params.h"
#include "Patch.h"

```

Include dependency graph for GDRelease.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class [GDRelease](#)

## 7.14 GDRelease.h

[Go to the documentation of this file.](#)

```

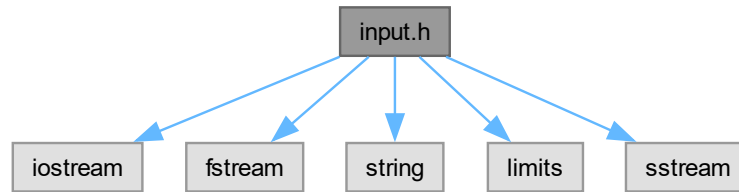
00001 #ifndef GDRELEASE_H
00002 #define GDRELEASE_H
00003
00004 #include <vector>
00005 #include "Params.h"
00006 #include "Patch.h"
00007
00008 class Patch;
00009
00010 // Implements gene drive release of mosquitoes into the collection of mosquito sites.
00011 class GDRelease {
00012 public:
00013     GDRelease(ReleaseParams* params);
00014     void release_gene_drive(std::vector<Patch*> &sites);
00015     bool is_release_time(int day);
00016
00017 private:
00018     int driver_start; // time to start releasing drive alleles into the mosquito population
00019     int num_driver_M; // number of drive heterozygous (WD) male mosquitoes per release
00020     int num_driver_sites; // number of gene drive release sites per year
00021
00022     std::vector<Patch*> select_driver_sites(int num_rel_sites, const std::vector<Patch*> &sites);
00023     void put_driver_sites(std::vector<Patch*> &rel_sites, std::vector<Patch*> &sites);
00024 };
00025
00026 #endif //GDRELEASE_H
  
```

## 7.15 input.h File Reference

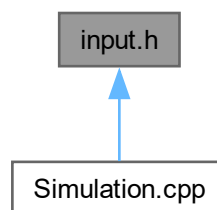
```

#include <iostream>
#include <fstream>
#include <string>
  
```

```
#include <limits>
#include <sstream>
Include dependency graph for input.h:
```



This graph shows which files directly or indirectly include this file:



## Functions

- `template<typename T>`  
`bool check_bounds (const std::string &par_name, T value, T lower_bound, bool inclusive_lower=true, T upper_bound=std::numeric_limits< T >::max(), bool inclusive_upper=true)`
- `template<typename T>`  
`bool read_and_validate_type (std::ifstream &file, T &par, const std::string &par_name, const std::string &par_type)`
- `template<typename T>`  
`bool read_and_validate_type (std::stringstream &linestream, T &par, const std::string &par_name, const std::string &par_type)`

## 7.15.1 Function Documentation

### 7.15.1.1 check\_bounds()

```
template<typename T>
bool check_bounds (
    const std::string & par_name,
```

```
T value,  
T lower_bound,  
bool inclusive_lower = true,  
T upper_bound = std::numeric_limits<T>::max(),  
bool inclusive_upper = true )
```

Here is the caller graph for this function:



#### 7.15.1.2 read\_and\_validate\_type() [1/2]

```
template<typename T >  
bool read_and_validate_type (  
    std::ifstream & file,  
    T & par,  
    const std::string & par_name,  
    const std::string & par_type )
```

Here is the caller graph for this function:



#### 7.15.1.3 read\_and\_validate\_type() [2/2]

```
template<typename T >  
bool read_and_validate_type (  
    std::stringstream & linestream,  
    T & par,  
    const std::string & par_name,  
    const std::string & par_type )
```

## 7.16 input.h

[Go to the documentation of this file.](#)

```

00001 #ifndef INPUT_H
00002 #define INPUT_H
00003
00004 #include <iostream>
00005 #include <fstream>
00006 #include <string>
00007 #include <limits>
00008 #include <sstream>
00009
00010 // Checks if a value falls within bounds.
00011 template <typename T>
00012 bool check_bounds(const std::string &par_name, T value, T lower_bound, bool inclusive_lower = true,
00013 T upper_bound = std::numeric_limits<T>::max(), bool inclusive_upper = true)
00014 {
00015     if (lower_bound >= upper_bound) {
00016         std::cerr << "Error: lower_bound must be less than upper_bound." << std::endl;
00017         return false;
00018     }
00019
00020     bool lower_check = inclusive_lower ? value >= lower_bound : value > lower_bound;
00021     bool upper_check = inclusive_upper ? value <= upper_bound : value < upper_bound;
00022
00023     std::string lower_bound_char = inclusive_lower ? "\u2264" : "<";
00024     std::string upper_bound_char = inclusive_upper ? "\u2264" : "<";
00025     if (!lower_check || !upper_check) {
00026         std::cout << "The parameter " << par_name << " is out of bounds. ";
00027         std::cout << par_name << " should be " << lower_bound << " " << lower_bound_char << " " << par_name <<
00028 " ";
00029         std::cout << upper_bound_char << " " << upper_bound << ". " << std::endl;
00030     }
00031     return lower_check && upper_check;
00032 }
00033
00034 // Reads a value from filestream and assigns it to the parameter variable if the types match.
00035 template <typename T>
00036 bool read_and_validate_type(std::ifstream &file, T &par, const std::string &par_name, const
std::string &par_type)
00037 {
00038     if (!(file >> par)) {
00039         std::cerr << "Error: invalid type for " << par_name << ". Expected " << par_type << "." <<
std::endl;
00040         return false;
00041     }
00042     return true;
00043 }
00044
00045 template <typename T>
00046 bool read_and_validate_type(std::stringstream &linestream, T &par, const std::string &par_name, const
std::string &par_type)
00047 {
00048     if (!(linestream >> par)) {
00049         std::cerr << "Error: invalid type for " << par_name << ". Expected " << par_type << "." <<
std::endl;
00050         return false;
00051     }
00052     return true;
00053 }
00054
00055 #endif //INPUT_H

```

## 7.17 main.cpp File Reference

```

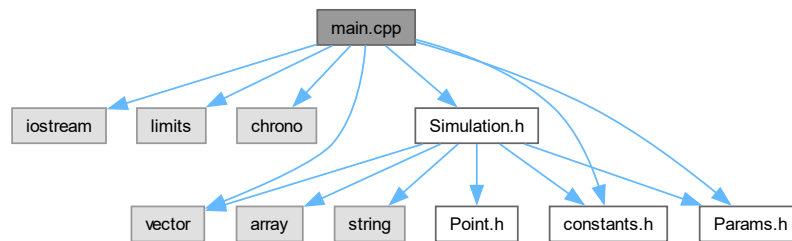
#include <iostream>
#include <limits>
#include <chrono>
#include <vector>
#include "Simulation.h"
#include "Params.h"

```



```
#include "constants.h"
```

Include dependency graph for main.cpp:



## Functions

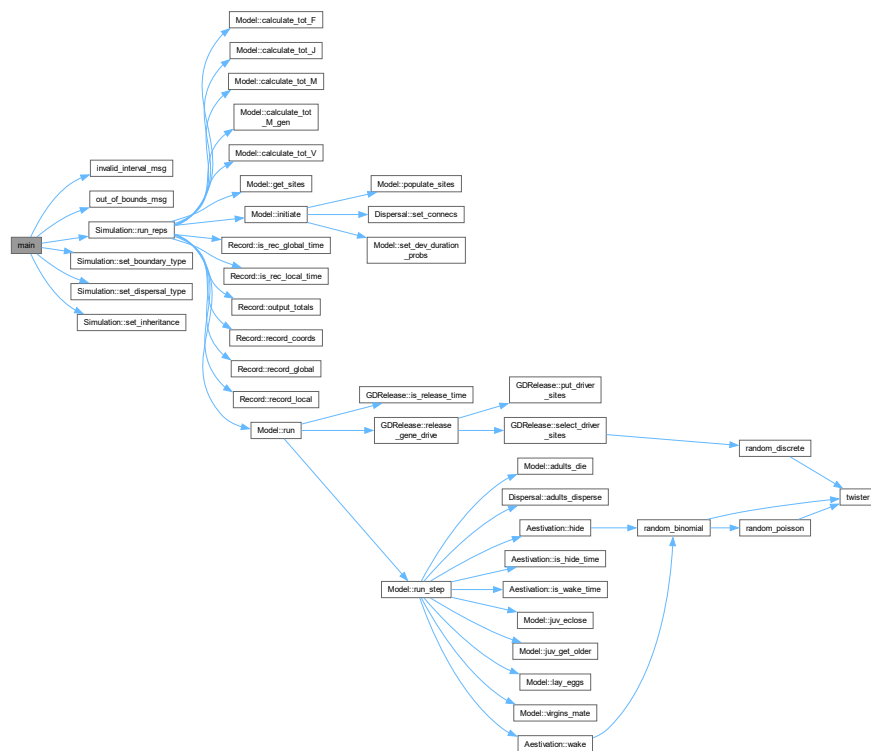
- int [main](#) ()

### 7.17.1 Function Documentation

#### 7.17.1.1 main()

```
int main ( )
```

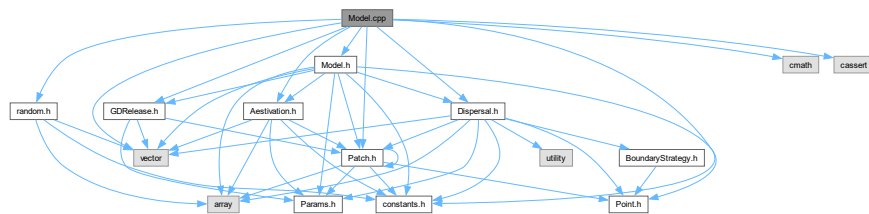
Here is the call graph for this function:



## 7.18 Model.cpp File Reference

```
#include <vector>
#include <cmath>
#include <cassert>
#include "Model.h"
#include "random.h"
#include "constants.h"
#include "Patch.h"
#include "Dispersal.h"
#include "GDRelease.h"
#include "Aestivation.h"
```

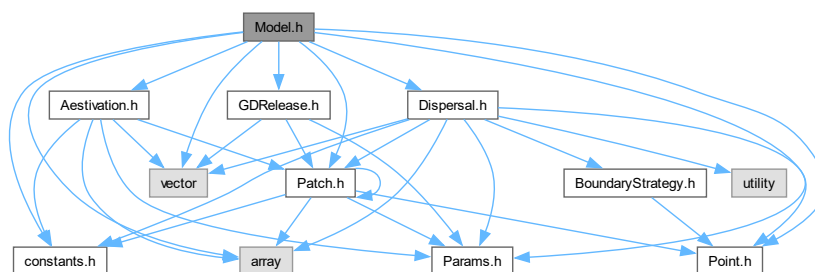
Include dependency graph for Model.cpp:



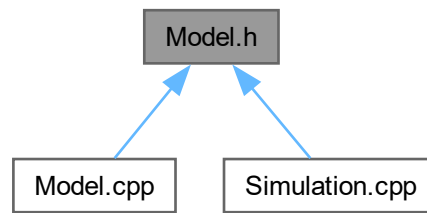
## 7.19 Model.h File Reference

```
#include <vector>
#include <array>
#include "constants.h"
#include "Params.h"
#include "Patch.h"
#include "Dispersal.h"
#include "Aestivation.h"
#include "GDRelease.h"
#include "Point.h"
```

Include dependency graph for Model.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class [Model](#)

## 7.20 Model.h

[Go to the documentation of this file.](#)

```

00001 #ifndef MODEL_H
00002 #define MODEL_H
00003
00004 #include <vector>
00005 #include <array>
00006 #include "constants.h"
00007 #include "Params.h"
00008 #include "Patch.h"
00009 #include "Dispersal.h"
00010 #include "Aestivation.h"
00011 #include "GDRelease.h"
00012 #include "Point.h"
00013
00014 using namespace constants;
00015
00016 // Runs the model.
00017 class Model {
00018 public:
00019     Model(AreaParams *area, InitialPopsParams *initial, LifeParams *life, AestivationParams *aes,
00020           DispersalParams *disp,
00021           ReleaseParams *rel, BoundaryType boundary = BoundaryType::Toroid, DispersalType disp_type =
00022           DispersalType::DistanceKernel,
00023           std::vector<Point> coords = {});
00024     ~Model();
00025     void initiate();
00026     void run(int day, const std::array<std::array<std::array<double, num_gen>, num_gen>, num_gen>
00027             &inher_fraction);
00028
00029     long long int calculate_tot_J();
00030     long long int calculate_tot_M();
00031     long long int calculate_tot_V();
00032     long long int calculate_tot_F();
00033     std::array<long long int, num_gen> calculate_tot_M_gen();
00034     std::vector<Patch*> get_sites() const;
00035
00036 private:
00037     std::vector<Patch*> sites;
00038     Dispersal* dispersal;
00039     Aestivation* aestivation;
00040     GDRelease* gd_release;
00041
00042     // simulation area parameters
00043     int num_pat; // number of population sites chosen for the simulation
00044     double side; // size of the square simulation area (side x side) (km)
00045
00046     // initial population values - common for all Patches
  
```

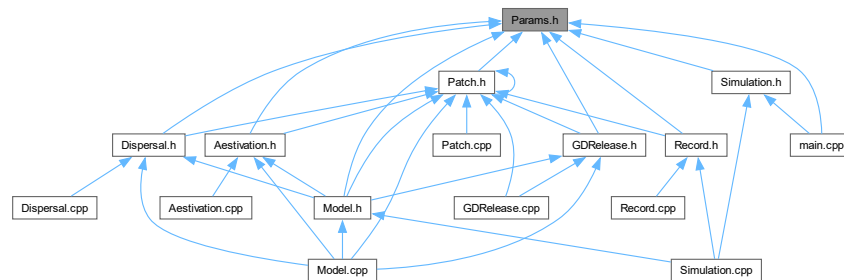
```

00044     InitialPopsParams *initial_pops;
00045
00046     // juvenile development parameters - common for all Patches
00047     int min_dev; // minimum development time for a juvenile (in days)
00048     std::array<double, max_dev+1> dev_duration_probs; // array of probabilities of juvenile
development duration for a new juvenile
00049     // (index indicates the number of days to develop or, equivalently, the age class the new juvenile
starts at)
00050
00051     // initiation methods
00052     void populate_sites();
00053     void set_dev_duration_probs(int min_time, int max_time);
00054
00055     // life-processes - interface with Patch
00056     void run_step(int day, const std::array<std::array<std::array<double, num_gen>, num_gen>,
num_gen> &inher_fraction);
00057     void juv_get_older();
00058     void adults_die();
00059     void virgins_mate();
00060     void lay_eggs(const std::array<std::array<std::array<double, num_gen>, num_gen>, num_gen>
&inher_fraction);
00061     void juv_eclose();
00062 };
00063
00064 #endif //MODEL_H

```

## 7.21 Params.h File Reference

This graph shows which files directly or indirectly include this file:



### Classes

- struct [ProgressionParams](#)
- struct [AreaParams](#)
- struct [LifeParams](#)
- struct [InheritanceParams](#)
- struct [ReleaseParams](#)
- struct [DispersalParams](#)
- struct [AestivationParams](#)
- struct [InitialPopsParams](#)
- struct [RecordParams](#)

## 7.22 Params.h

[Go to the documentation of this file.](#)

```

00001 #ifndef PARAMS_H
00002 #define PARAMS_H
00003
00004 // ** These parameters should not be modified after being passed to the Simulation or Model **
00005
00006 // Simulation progression parameters
00007 struct ProgressionParams {
00008     int num_runs = 1; // number of simulation replicates to run
00009     int max_t = 1000; // maximum simulated time (in days)
00010 };
00011
00012 // Model area parameters
00013 struct AreaParams {
00014     int num_pat = 50; // number of population sites chosen for the simulation
00015     double side = 1.0; // size of the square simulation area (side x side) (km)
00016 };
00017
00018 // Model life-process parameters
00019 struct LifeParams {
00020     double mu_j = 0.05; // juvenile density independent mortality rate per day
00021     double mu_a = 0.125; // adult mortality rate per day
00022     double beta = 100.0; // parameter that controls mating rate
00023     double theta = 9.0; // average egg laying rate of wildtype females (eggs per day)
00024     double alpha0 = 100000.0; // baseline contribution to carrying capacity
00025     double mean_dev = 15.0; // mean juvenile development time (in days)
00026     int min_dev = 10; // minimum development time for a juvenile (in days)
00027 };
00028
00029 // Gene drive inheritance parameters
00030 struct InheritanceParams {
00031     double gamma = 0.025; // rate of r2 allele formation from W/D meiosis
00032     double xi = 0.2; // somatic Cas9 expression fitness cost
00033     double e = 0.95; // homing rate in females
00034 };
00035
00036 // Gene drive release model parameters
00037 struct ReleaseParams {
00038     int driver_start = 200; // time to start releasing drive alleles into the mosquito population
00039     int num_driver_M = 1000; // number of drive heterozygous (WD) male mosquitoes per release
00040     int num_driver_sites = 5; // number of gene drive release sites per year
00041 };
00042
00043 // Dispersal model parameters
00044 struct DispersalParams {
00045     double disp_rate = 0.01; // adult dispersal rate
00046     double max_disp = 0.2; // maximum distance at which two sites are connected (km)
00047 };
00048
00049 // Aestivation model parameters
00050 struct AestivationParams {
00051     double psi = 0.0; // aestivation rate
00052     double mu_aes = 0.0; // aestivation mortality
00053     int t_hidel = 0; // start day of aestivation-entering period (day number of the year), not
    included
00054     int t_hide2 = 0; // end day of aestivation-entering period (day number of the year)
00055     int t_wakel = 0; // start day of aestivation-waking period (day number of the year), not included
00056     int t_wake2 = 0; // end day of aestivation-waking period (day number of the year)
00057 };
00058
00059 // Initial population values for the model
00060 struct InitialPopsParams {
00061     int initial_WJ = 10000; // array of number of initial juvenile mosquitoes with wild homozygous
    (WW) genotype for each age group
00062     int initial_WM = 50000; // number of initial adult male mosquitoes with wild homozygous (WW)
    genotype
00063     int initial_WV = 10000; // number of initial adult unmated female (virgin) mosquitoes with wild
    homozygous (WW) genotype
00064     int initial_WF = 40000; // number of initial adult mated female mosquitoes with wild homozygous
    (WW) genotype
00065 };
00066
00067 // Data-recording parameters
00068 struct RecordParams {
00069     // recording window and intervals
00070     int rec_start = 0; // start time for the data recording window (in days) (non-inclusive)
00071     int rec_end = 1000; // end time for the data recording window (in days) (inclusive)
00072     int rec_interval_global = 1; // time interval for global data recording/output
00073     int rec_interval_local = 200; // time interval at which to collect/record local data (in days)
00074     int rec_sites_freq = 1; // fraction of sites to collect local data for (1 is all sites, 10 is 1 in
    10 etc)
00075
00076     // output filename labels

```

```

00077     int set_label = 1; // 'set of runs' index label for output files
00078 };
00079
00080 #endif //PARAMS_H

```

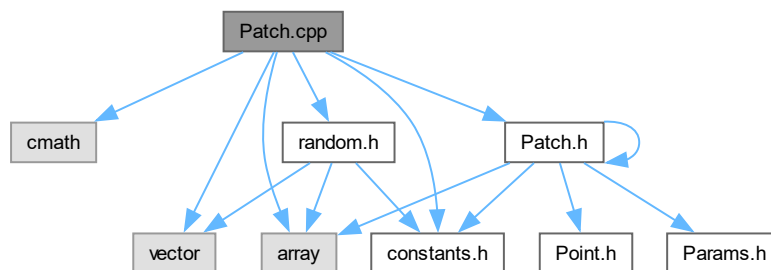
## 7.23 Patch.cpp File Reference

```

#include <cmath>
#include <array>
#include <vector>
#include "Patch.h"
#include "random.h"
#include "constants.h"

```

Include dependency graph for Patch.cpp:



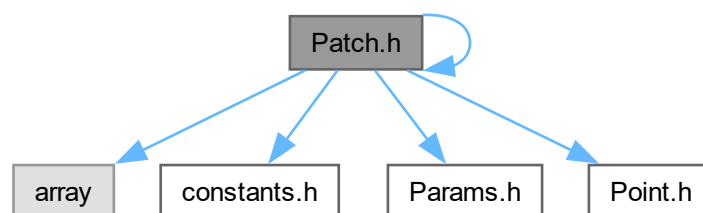
## 7.24 Patch.h File Reference

```

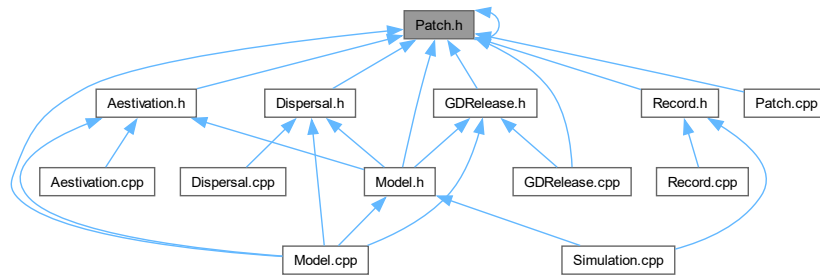
#include <array>
#include "Patch.h"
#include "constants.h"
#include "Params.h"
#include "Point.h"

```

Include dependency graph for Patch.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class [Patch](#)

## 7.25 Patch.h

[Go to the documentation of this file.](#)

```

00001 #ifndef PATCH_H
00002 #define PATCH_H
00003
00004 #include <array>
00005 #include "Patch.h"
00006 #include "constants.h"
00007 #include "Params.h"
00008 #include "Point.h"
00009
00010 using namespace constants;
00011
00012 // Contains the information of a local mosquito population
00013 class Patch {
00014 public:
00015     Patch(LifeParams* par, double side);
00016     Patch(LifeParams* par, Point point);
00017     void populate(int initial_WJ, int initial_WM, int initial_WV, int initial_WF);
00018
00019     Point get_coords() const;
00020     std::array<long long int, num_gen> get_M() const;
00021     std::array<std::array<long long int, num_gen>, num_gen> get_F() const;
00022
00023     long long int calculate_tot_J();
00024     long long int calculate_tot_M();
00025     long long int calculate_tot_V();
00026     long long int calculate_tot_F();
00027
00028     // life-processes for the local site
00029     void juv_get_older(int max_dev);
00030     void adults_die();
00031     void virgins_mate();
00032     void lay_eggs(const std::array<std::array<double, num_gen>, num_gen> &f,
00033                 const std::array<double, max_dev+1> &dev_duration_probs);
00034     void juv_eclose();
00035     void update_comp();
00036     void update_mate();
00037
00038     // interface to Dispersal
00039     void M_disperse_out(const std::array<long long int, num_gen> &m_out);
00040     void F_disperse_out(const std::array<std::array<long long int, num_gen>, num_gen> &f_out);
00041     void M_disperse_in(int gen, long long int m_in);
00042     void F_disperse_in(int f_gen, int m_gen, long long int f_disp);
00043
00044     // interface to Aestivation
00045     void F_hide(const std::array<std::array<long long int, num_gen>, num_gen> &f_try);
00046     void F_wake(const std::array<std::array<long long int, num_gen>, num_gen> &f_wake);
00047
00048     // interface to GDRelease
  
```

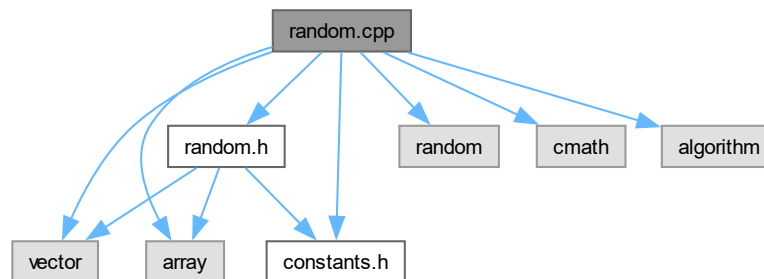




## 7.28 random.cpp File Reference

```
#include <vector>
#include <array>
#include <random>
#include <cmath>
#include <algorithm>
#include "random.h"
#include "constants.h"
```

Include dependency graph for random.cpp:



### Functions

- std::mt19937 [twister](#) (1)
- double [random\\_real](#) ()
- int [random\\_discrete](#) (int a, int b)
- long long int [random\\_poisson](#) (double lambda)
- long long int [random\\_binomial](#) (long long int n, double p)
- std::vector< long long int > [random\\_multinomial](#) (long long int n, const std::vector< double > &probs)
- std::vector< long long int > [random\\_multinomial](#) (long long int n, const std::array< long long int, num\_gen > &probs)
- std::vector< long long int > [random\\_multinomial](#) (long long int n, const std::array< double, max\_dev+1 > &probs)

### Variables

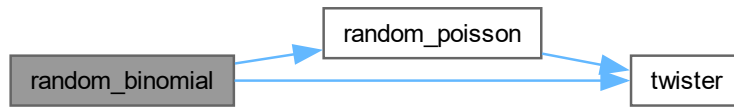
- std::random\_device [rd](#)

### 7.28.1 Function Documentation

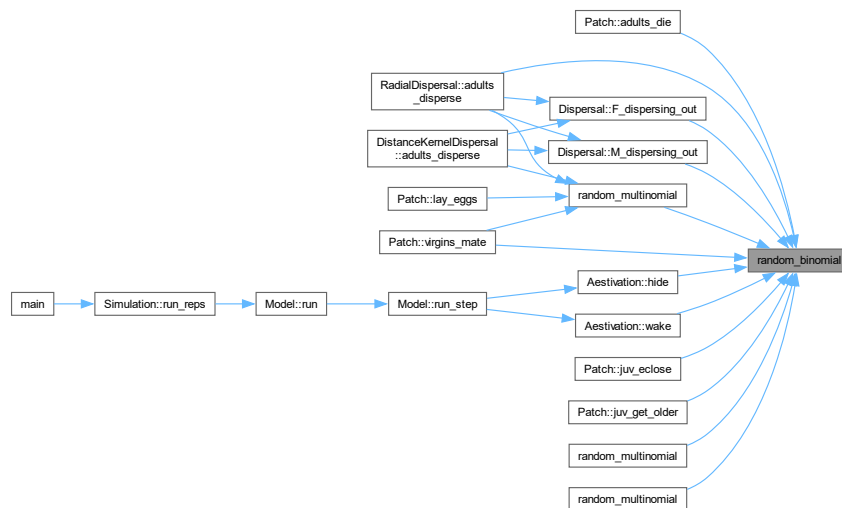
#### 7.28.1.1 random\_binomial()

```
long long int random_binomial (
    long long int n,
    double p )
```

Here is the call graph for this function:



Here is the caller graph for this function:



### 7.28.1.2 random\_discrete()

```

int random_discrete (
    int a,
    int b )
  
```

Here is the call graph for this function:



Here is the caller graph for this function:



### 7.28.1.3 random\_multinomial() [1/3]

```

std::vector< long long int > random_multinomial (
    long long int n,
    const std::array< double, max_dev+1 > & probs )
  
```

Here is the call graph for this function:



### 7.28.1.4 random\_multinomial() [2/3]

```

std::vector< long long int > random_multinomial (
    long long int n,
    const std::array< long long int, num_gen > & probs )
  
```

Here is the call graph for this function:

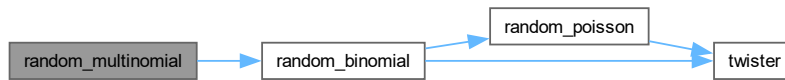


### 7.28.1.5 random\_multinomial() [3/3]

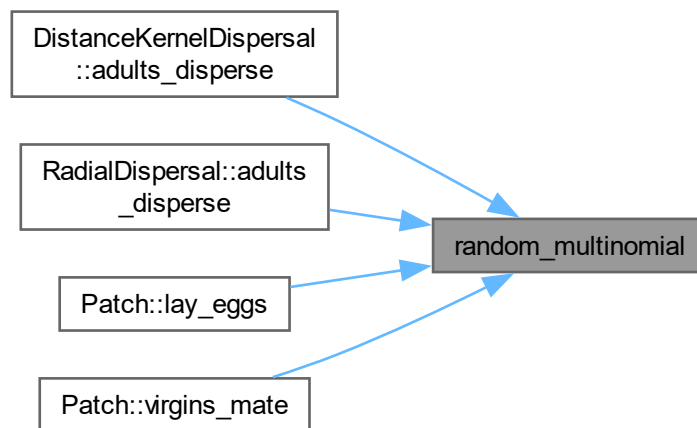
```

std::vector< long long int > random_multinomial (
    long long int n,
    const std::vector< double > & probs )
  
```

Here is the call graph for this function:



Here is the caller graph for this function:

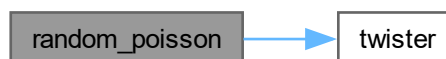


#### 7.28.1.6 random\_poisson()

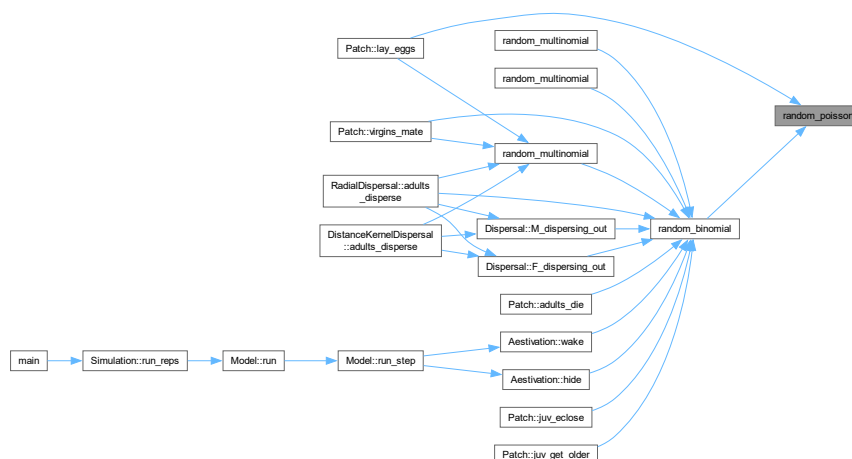
```

long long int random_poisson (
    double lambda )
  
```

Here is the call graph for this function:



Here is the caller graph for this function:



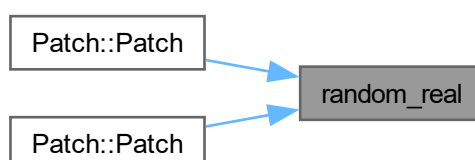
### 7.28.1.7 random\_real()

```
double random_real ( )
```

Here is the call graph for this function:



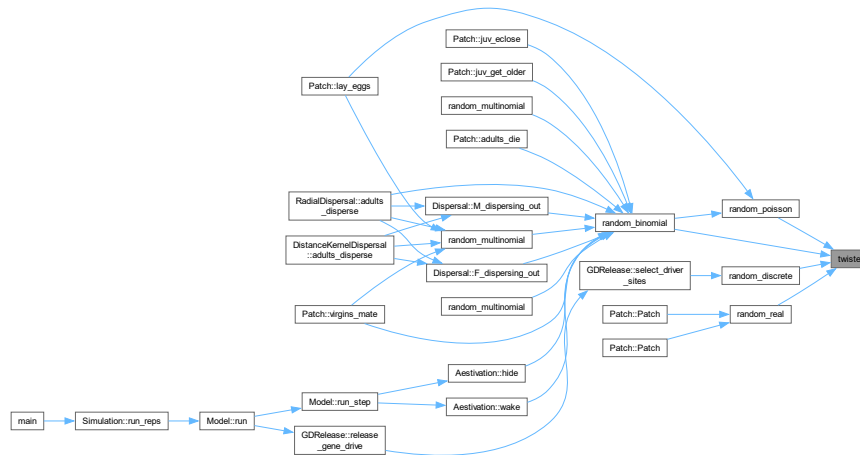
Here is the caller graph for this function:



### 7.28.1.8 twister()

```
std::mt19937 twister (
    1 )
```

Here is the caller graph for this function:



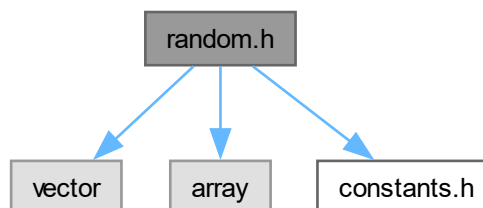
## 7.28.2 Variable Documentation

### 7.28.2.1 rd

```
std::random_device rd
```

## 7.29 random.h File Reference

```
#include <vector>
#include <array>
#include "constants.h"
Include dependency graph for random.h:
```



This graph shows which files directly or indirectly include this file:



## Functions

- double [random\\_real](#) ()
- int [random\\_discrete](#) (int a, int b)
- long long int [random\\_poisson](#) (double lambda)
- long long int [random\\_binomial](#) (long long int n, double p)
- std::vector< long long int > [random\\_multinomial](#) (long long int n, const std::vector< double > &probs)
- std::vector< long long int > [random\\_multinomial](#) (long long int n, const std::array< long long int, num\_gen > &probs)
- std::vector< long long int > [random\\_multinomial](#) (long long int n, const std::array< double, max\_dev+1 > &probs)

### 7.29.1 Function Documentation

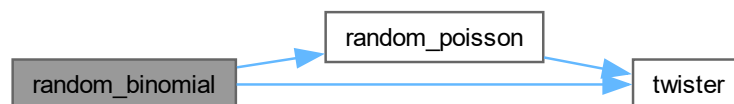
#### 7.29.1.1 random\_binomial()

```

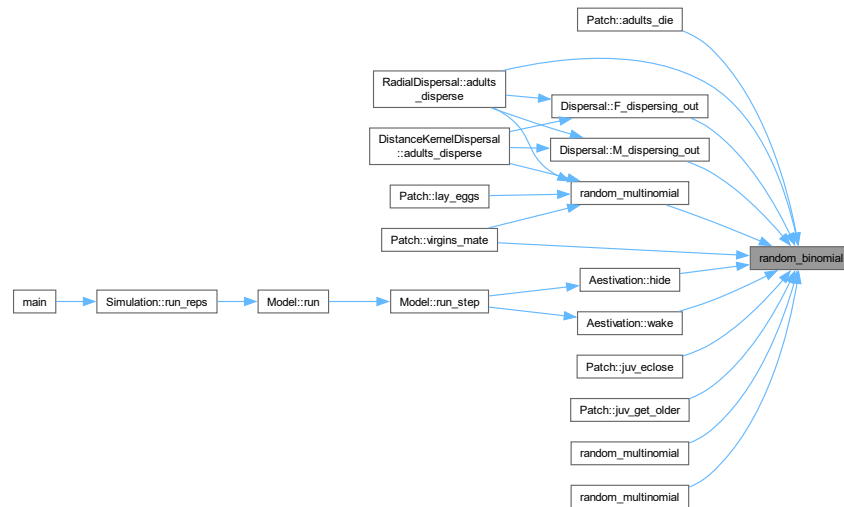
long long int random_binomial (
    long long int n,
    double p )

```

Here is the call graph for this function:



Here is the caller graph for this function:



### 7.29.1.2 random\_discrete()

```

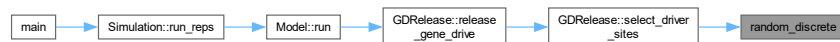
int random_discrete (
    int a,
    int b )

```

Here is the call graph for this function:



Here is the caller graph for this function:





### 7.29.1.3 random\_multinomial() [1/3]

```
std::vector< long long int > random_multinomial (
    long long int n,
    const std::array< double, max_dev+1 > & probs )
```

Here is the call graph for this function:



### 7.29.1.4 random\_multinomial() [2/3]

```
std::vector< long long int > random_multinomial (
    long long int n,
    const std::array< long long int, num_gen > & probs )
```

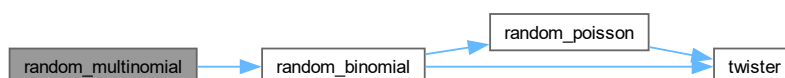
Here is the call graph for this function:



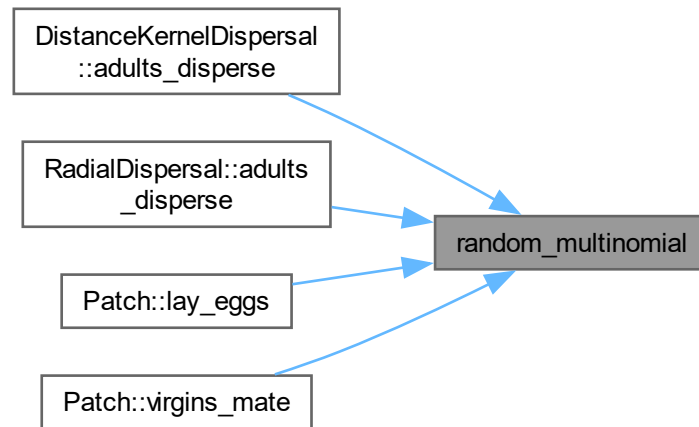
### 7.29.1.5 random\_multinomial() [3/3]

```
std::vector< long long int > random_multinomial (
    long long int n,
    const std::vector< double > & probs )
```

Here is the call graph for this function:



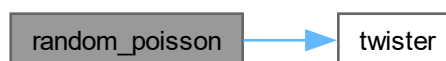
Here is the caller graph for this function:



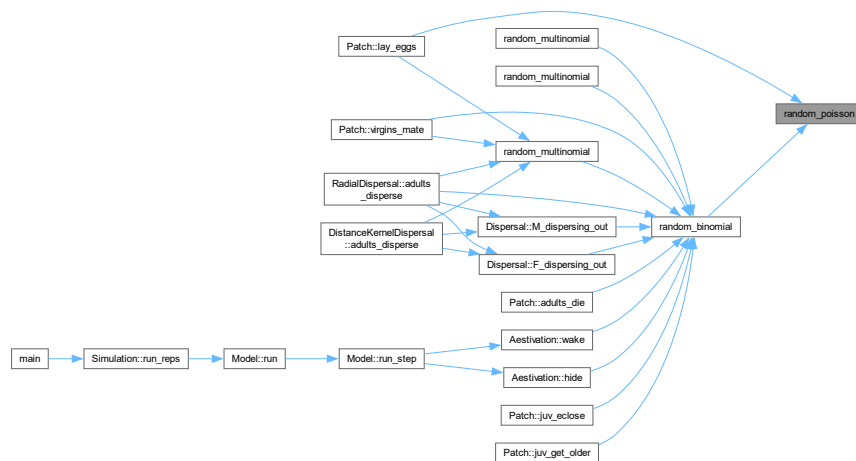
#### 7.29.1.6 `random_poisson()`

```
long long int random_poisson (  
    double lambda )
```

Here is the call graph for this function:



Here is the caller graph for this function:



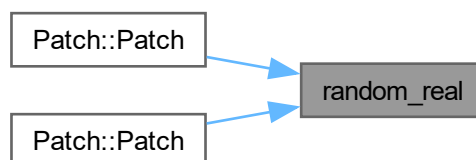
### 7.29.1.7 random\_real()

```
double random_real ( )
```

Here is the call graph for this function:



Here is the caller graph for this function:



## 7.30 random.h

[Go to the documentation of this file.](#)

```

00001 #include <vector>
00002 #include <array>
00003 #include "constants.h"
00004
00005 using namespace constants;
00006
00007 // Random number generator functions
00008
00009 double random_real();
00010 int random_discrete(int a, int b);
00011 long long int random_poisson(double lambda);
00012 long long int random_binomial(long long int n, double p);
00013 std::vector<long long int> random_multinomial(long long int n, const std::vector<double>& probs);
00014 std::vector<long long int> random_multinomial(long long int n, const std::array<long long int,
num_gen>& probs);
00015 std::vector<long long int> random_multinomial(long long int n, const std::array<double, max_dev+1>&
probs);

```

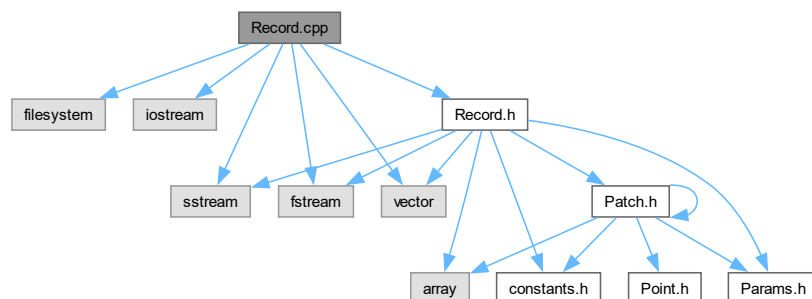
## 7.31 Record.cpp File Reference

```

#include <filesystem>
#include <iostream>
#include <sstream>
#include <fstream>
#include <vector>
#include "Record.h"

```

Include dependency graph for Record.cpp:



## 7.32 Record.h File Reference

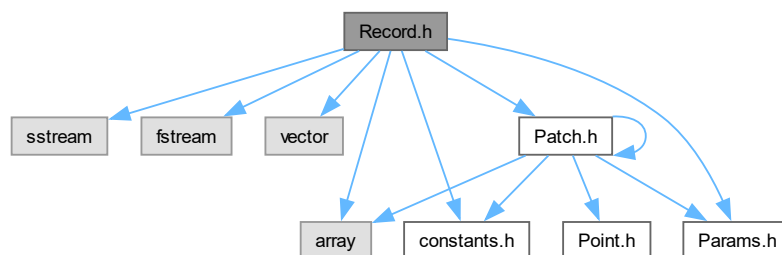
```

#include <sstream>
#include <fstream>
#include <vector>
#include <array>
#include "constants.h"
#include "Patch.h"

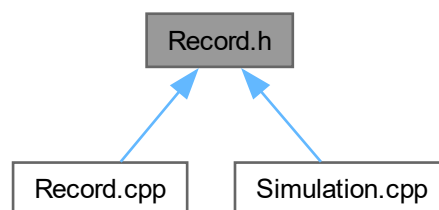
```

```
#include "Params.h"
```

Include dependency graph for Record.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class [Record](#)

## 7.33 Record.h

[Go to the documentation of this file.](#)

```

00001 #ifndef RECORD_H
00002 #define RECORD_H
00003
00004 #include <sstream>
00005 #include <fstream>
00006 #include <vector>
00007 #include <array>
00008 #include "constants.h"
00009 #include "Patch.h"
00010 #include "Params.h"
00011
00012 using namespace constants;
00013
00014 // Records model data.
00015 class Record {
00016 public:
00017     Record(RecordParams *rec_params, int rep);
00018     ~Record();
  
```

```

00019     void record_coords(const std::vector<Patch*> &sites);
00020     void record_global(int day, const std::array<long long int, num_gen> &tot_M_gen);
00021     void output_totals(int day, long long int tot_J, long long int tot_M, long long int tot_V, long
long int tot_F);
00022     void record_local(int day, const std::vector<Patch*> &sites);
00023
00024     bool is_rec_local_time(int day);
00025     bool is_rec_global_time(int day);
00026
00027 private:
00028     // recording window and intervals
00029     int rec_start; // start time for the data recording window (in days) (non-inclusive)
00030     int rec_end; // end time for the data recording window (in days) (inclusive)
00031     int rec_interval_global; // time interval for global data recording/output
00032     int rec_interval_local; // time interval at which to collect/record local data (in days)
00033     int rec_sites_freq; // fraction of sites to collect local data for (1 is all sites, 10 is 1 in 10
etc)
00034
00035     // output filename labels
00036     int set_label; // 'set of repetitions' index label for output files
00037     int rep_label; // 'repetition' index label in given set of repetitions for output files
00038
00039     std::ostringstream os1, os2, os3; // filenames
00040     std::ofstream local_data, global_data, coord_list; // file objects
00041 };
00042
00043 #endif //RECORD_H

```

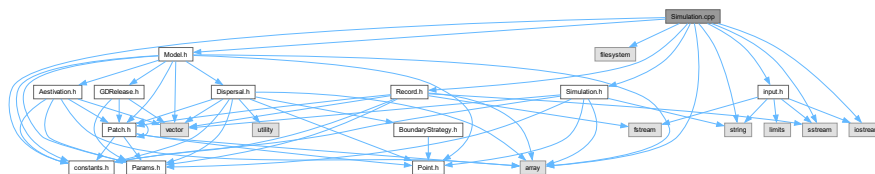
## 7.34 Simulation.cpp File Reference

```

#include <array>
#include <iostream>
#include <filesystem>
#include <sstream>
#include <string>
#include "Simulation.h"
#include "constants.h"
#include "Model.h"
#include "Record.h"
#include "input.h"

```

Include dependency graph for Simulation.cpp:



### Functions

- void [out\\_of\\_bounds\\_msg](#) (const std::string &par)
- void [invalid\\_interval\\_msg](#) (const std::string &param1, const std::string &param2)

### 7.34.1 Function Documentation

#### 7.34.1.1 invalid\_interval\_msg()

```

void invalid_interval_msg (
    const std::string & param1,
    const std::string & param2 )

```

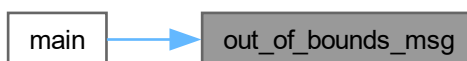
Here is the caller graph for this function:



#### 7.34.1.2 out\_of\_bounds\_msg()

```
void out_of_bounds_msg (  
    const std::string & par )
```

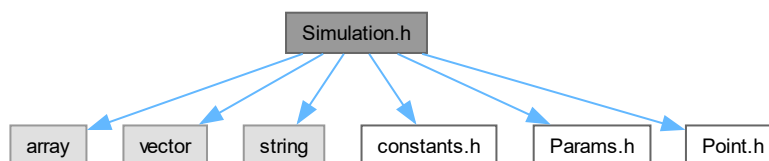
Here is the caller graph for this function:



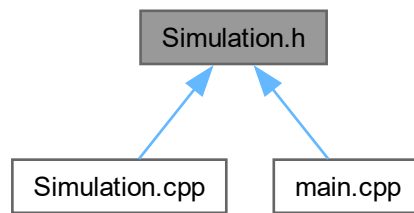
## 7.35 Simulation.h File Reference

```
#include <array>  
#include <vector>  
#include <string>  
#include "constants.h"  
#include "Params.h"  
#include "Point.h"
```

Include dependency graph for Simulation.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class [Simulation](#)

## Functions

- void [out\\_of\\_bounds\\_msg](#) (const std::string &par)
- void [invalid\\_interval\\_msg](#) (const std::string &param1, const std::string &param2)

## 7.35.1 Function Documentation

### 7.35.1.1 invalid\_interval\_msg()

```
void invalid_interval_msg (  
    const std::string & param1,  
    const std::string & param2 )
```

Here is the caller graph for this function:





## 7.35.1.2 out\_of\_bounds\_msg()

```
void out_of_bounds_msg (
    const std::string & par )
```

Here is the caller graph for this function:



## 7.36 Simulation.h

[Go to the documentation of this file.](#)

```
00001 #ifndef SIMULATION_H
00002 #define SIMULATION_H
00003
00004 #include <array>
00005 #include <vector>
00006 #include <string> // for error messages
00007 #include "constants.h"
00008 #include "Params.h"
00009 #include "Point.h"
00010
00011 using namespace constants;
00012
00013 void out_of_bounds_msg(const std::string& par);
00014 void invalid_interval_msg(const std::string& param1, const std::string& param2);
00015
00016 // Sets up and controls the flow of the simulation.
00017 class Simulation {
00018 public:
00019     Simulation(ProgressionParams &prog, AreaParams &area, LifeParams &life, ReleaseParams &rel,
00020               DispersalParams &disp,
00021               AestivationParams &aes, InitialPopsParams &initial, RecordParams &rec);
00022     void set_coords(const std::string& coords_file);
00023     void set_boundary_type(BoundaryType boundary);
00024     void set_dispersal_type(DispersalType disp);
00025     void set_inheritance(InheritanceParams inher_params);
00026     void run_reps();
00027 private:
00028     int num_runs; // number of simulation replicates to run
00029     int max_t; // maximum simulated time (in days)
00030
00031     AreaParams *area_params; // model area parameters
00032     LifeParams *life_params; // model life-process parameters
00033     ReleaseParams *rel_params; // gene drive release model parameters
00034     DispersalParams *disp_params; // dispersal model parameters
00035     AestivationParams *aes_params; // aestivation model parameters
00036     InitialPopsParams *initial_params; // initial population values
00037     RecordParams *rec_params; // data-recording parameters
00038
00039     // additional parameter options
00040     std::vector<Point> sites_coords; // 2D coordinates for the sites on the simulated square
00041     BoundaryType boundary_type;
00042     DispersalType disp_type;
00043
00044     // inheritance
00045     // f_ijk is the fraction of genotype k offspring from mother with genotype i mated to father with
00046     // genotype j
00047     std::array<std::array<std::array<double, num_gen>, num_gen>, num_gen> inher_fraction;
00048 };
00049 #endif //SIMULATION_H
```



# Index

- ~Dispersal
  - Dispersal, [21](#)
- ~Model
  - Model, [38](#)
- ~Record
  - Record, [65](#)
- add\_driver\_M
  - Patch, [49](#)
- adults\_die
  - Model, [39](#)
  - Patch, [49](#)
- adults\_disperse
  - Dispersal, [21](#)
  - DistanceKernelDispersal, [27](#)
  - RadialDispersal, [61](#)
- aes\_F
  - Aestivation, [14](#)
- aes\_params
  - Simulation, [75](#)
- Aestivation, [11](#)
  - aes\_F, [14](#)
  - Aestivation, [12](#)
  - hide, [12](#)
  - is\_hide\_time, [12](#)
  - is\_wake\_time, [13](#)
  - mu\_aes, [14](#)
  - psi, [14](#)
  - t\_hide1, [14](#)
  - t\_hide2, [14](#)
  - t\_wake1, [14](#)
  - t\_wake2, [14](#)
  - wake, [13](#)
- aestivation
  - Model, [45](#)
- Aestivation.cpp, [79](#)
- Aestivation.h, [79](#)
- AestivationParams, [15](#)
  - mu\_aes, [15](#)
  - psi, [15](#)
  - t\_hide1, [15](#)
  - t\_hide2, [15](#)
  - t\_wake1, [16](#)
  - t\_wake2, [16](#)
- alpha0
  - LifeParams, [36](#)
- area\_params
  - Simulation, [75](#)
- AreaParams, [16](#)
  - num\_pat, [16](#)
  - side, [16](#)
- beta
  - LifeParams, [36](#)
- boundary\_strategy
  - Dispersal, [23](#)
- boundary\_type
  - Simulation, [75](#)
- BoundaryStrategy, [17](#)
  - BoundaryStrategy, [18](#)
  - distance, [18](#)
  - side, [18](#)
- BoundaryStrategy.cpp, [81](#)
- BoundaryStrategy.h, [82](#)
- BoundaryType
  - constants.h, [83](#)
- calculate\_tot\_F
  - Model, [39](#)
  - Patch, [49](#)
- calculate\_tot\_J
  - Model, [39](#)
  - Patch, [50](#)
- calculate\_tot\_M
  - Model, [39](#)
  - Patch, [50](#)
- calculate\_tot\_M\_gen
  - Model, [40](#)
- calculate\_tot\_V
  - Model, [40](#)
  - Patch, [50](#)
- check\_bounds
  - input.h, [90](#)
- comp
  - Patch, [55](#)
- compute\_connecs
  - DistanceKernelDispersal, [27](#)
  - RadialDispersal, [61](#)
- compute\_distances
  - RadialDispersal, [61](#)
- compute\_interval\_union
  - RadialDispersal, [62](#)
- connec\_indices
  - Dispersal, [23](#)
- connec\_weights
  - Dispersal, [23](#)
- connec\_weights\_sum
  - RadialDispersal, [64](#)
- constants, [9](#)
  - max\_dev, [9](#)

- num\_gen, 9
- constants.h, 83
  - BoundaryType, 83
  - DispersalType, 84
  - DistanceKernel, 84
  - Edge, 84
  - Radial, 84
  - Toroid, 84
- coord\_list
  - Record, 67
- coords
  - Patch, 55
- dev\_duration\_probs
  - Model, 45
- disp\_params
  - Simulation, 75
- disp\_rate
  - Dispersal, 23
  - DispersalParams, 24
- disp\_type
  - Simulation, 75
- Dispersal, 19
  - ~Dispersal, 21
  - adults\_disperse, 21
  - boundary\_strategy, 23
  - connec\_indices, 23
  - connec\_weights, 23
  - disp\_rate, 23
  - Dispersal, 21
  - F\_dispersing\_out, 21
  - M\_dispersing\_out, 22
  - max\_disp, 23
  - set\_connecs, 22
- dispersal
  - Model, 45
- Dispersal.cpp, 84
  - PI, 85
  - TWOPI, 85
- Dispersal.h, 85
- DispersalParams, 24
  - disp\_rate, 24
  - max\_disp, 24
- DispersalType
  - constants.h, 84
- distance
  - BoundaryStrategy, 18
  - EdgeBoundaryStrategy, 30
  - ToroidalBoundaryStrategy, 78
- DistanceKernel
  - constants.h, 84
- DistanceKernelDispersal, 25
  - adults\_disperse, 27
  - compute\_connecs, 27
  - DistanceKernelDispersal, 27
  - set\_connecs, 28
- driver\_start
  - GDRelease, 33
  - ReleaseParams, 70
- e
  - InheritanceParams, 34
- Edge
  - constants.h, 84
- EdgeBoundaryStrategy, 29
  - distance, 30
  - EdgeBoundaryStrategy, 30
- F
  - Patch, 55
- F\_disperse\_in
  - Patch, 50
- F\_disperse\_out
  - Patch, 51
- F\_dispersing\_out
  - Dispersal, 21
- F\_hide
  - Patch, 51
- F\_wake
  - Patch, 51
- gamma
  - InheritanceParams, 34
- gd\_release
  - Model, 45
- GDRelease, 30
  - driver\_start, 33
  - GDRelease, 31
  - is\_release\_time, 31
  - num\_driver\_M, 33
  - num\_driver\_sites, 33
  - put\_driver\_sites, 31
  - release\_gene\_drive, 31
  - select\_driver\_sites, 32
- GDRelease.cpp, 87
- GDRelease.h, 88
- get\_coords
  - Patch, 51
- get\_F
  - Patch, 51
- get\_M
  - Patch, 51
- get\_sites
  - Model, 40
- get\_sorted\_positions
  - RadialDispersal, 62
- global\_data
  - Record, 67
- hide
  - Aestivation, 12
- inher\_fraction
  - Simulation, 75
- InheritanceParams, 33
  - e, 34
  - gamma, 34
  - xi, 34
- initial\_params

- Simulation, 75
- initial\_pops
  - Model, 45
- initial\_WF
  - InitialPopsParams, 35
- initial\_WJ
  - InitialPopsParams, 35
- initial\_WM
  - InitialPopsParams, 35
- initial\_WV
  - InitialPopsParams, 35
- InitialPopsParams, 34
  - initial\_WF, 35
  - initial\_WJ, 35
  - initial\_WM, 35
  - initial\_WV, 35
- initiate
  - Model, 41
- input.h, 89
  - check\_bounds, 90
  - read\_and\_validate\_type, 91
- invalid\_interval\_msg
  - Simulation.cpp, 114
  - Simulation.h, 116
- is\_hide\_time
  - Aestivation, 12
- is\_rec\_global\_time
  - Record, 65
- is\_rec\_local\_time
  - Record, 65
- is\_release\_time
  - GDRelease, 31
- is\_wake\_time
  - Aestivation, 13
- J
  - Patch, 56
- juv\_eclose
  - Model, 41
  - Patch, 51
- juv\_get\_older
  - Model, 42
  - Patch, 51
- lay\_eggs
  - Model, 42
  - Patch, 52
- life\_params
  - Simulation, 76
- LifeParams, 35
  - alpha0, 36
  - beta, 36
  - mean\_dev, 36
  - min\_dev, 36
  - mu\_a, 36
  - mu\_j, 36
  - theta, 36
- local\_data
  - Record, 67

- M
  - Patch, 56
- M\_disperse\_in
  - Patch, 52
- M\_disperse\_out
  - Patch, 52
- M\_dispersing\_out
  - Dispersal, 22
- main
  - main.cpp, 93
- main.cpp, 92
  - main, 93
- mate\_rate
  - Patch, 56
- max\_dev
  - constants, 9
- max\_disp
  - Dispersal, 23
  - DispersalParams, 24
- max\_t
  - ProgressionParams, 57
  - Simulation, 76
- mean\_dev
  - LifeParams, 36
- min\_dev
  - LifeParams, 36
  - Model, 45
- Model, 37
  - ~Model, 38
  - adults\_die, 39
  - aestivation, 45
  - calculate\_tot\_F, 39
  - calculate\_tot\_J, 39
  - calculate\_tot\_M, 39
  - calculate\_tot\_M\_gen, 40
  - calculate\_tot\_V, 40
  - dev\_duration\_probs, 45
  - dispersal, 45
  - gd\_release, 45
  - get\_sites, 40
  - initial\_pops, 45
  - initiate, 41
  - juv\_eclose, 41
  - juv\_get\_older, 42
  - lay\_eggs, 42
  - min\_dev, 45
  - Model, 38
  - num\_pat, 45
  - populate\_sites, 42
  - run, 42
  - run\_step, 43
  - set\_dev\_duration\_probs, 44
  - side, 45
  - sites, 46
  - virgins\_mate, 44
- Model.cpp, 94
- Model.h, 94
- mu\_a

- LifeParams, 36
- mu\_aes
  - Aestivation, 14
  - AestivationParams, 15
- mu\_j
  - LifeParams, 36
- num\_driver\_M
  - GDRelease, 33
  - ReleaseParams, 70
- num\_driver\_sites
  - GDRelease, 33
  - ReleaseParams, 71
- num\_gen
  - constants, 9
- num\_pat
  - AreaParams, 16
  - Model, 45
- num\_runs
  - ProgressionParams, 57
  - Simulation, 76
- os1
  - Record, 67
- os2
  - Record, 68
- os3
  - Record, 68
- out\_of\_bounds\_msg
  - Simulation.cpp, 115
  - Simulation.h, 116
- output\_totals
  - Record, 66
- params
  - Patch, 56
- Params.h, 96
- Patch, 46
  - add\_driver\_M, 49
  - adults\_die, 49
  - calculate\_tot\_F, 49
  - calculate\_tot\_J, 50
  - calculate\_tot\_M, 50
  - calculate\_tot\_V, 50
  - comp, 55
  - coords, 55
  - F, 55
  - F\_disperse\_in, 50
  - F\_disperse\_out, 51
  - F\_hide, 51
  - F\_wake, 51
  - get\_coords, 51
  - get\_F, 51
  - get\_M, 51
  - J, 56
  - juv\_eclose, 51
  - juv\_get\_older, 51
  - lay\_eggs, 52
  - M, 56
  - M\_disperse\_in, 52
  - M\_disperse\_out, 52
  - mate\_rate, 56
  - params, 56
  - Patch, 48
  - populate, 53
  - update\_comp, 53
  - update\_mate, 54
  - V, 56
  - virgins\_mate, 55
- Patch.cpp, 98
- Patch.h, 98
- PI
  - Dispersal.cpp, 85
- Point, 56
  - x, 57
  - y, 57
- Point.h, 100
- populate
  - Patch, 53
- populate\_sites
  - Model, 42
- ProgressionParams, 57
  - max\_t, 57
  - num\_runs, 57
- psi
  - Aestivation, 14
  - AestivationParams, 15
- put\_driver\_sites
  - GDRelease, 31
- Radial
  - constants.h, 84
- RadialDispersal, 58
  - adults\_disperse, 61
  - compute\_connecs, 61
  - compute\_distances, 61
  - compute\_interval\_union, 62
  - connec\_weights\_sum, 64
  - get\_sorted\_positions, 62
  - RadialDispersal, 60
  - set\_connecs, 63
  - wrap\_around, 63
- random.cpp, 101
  - random\_binomial, 101
  - random\_discrete, 102
  - random\_multinomial, 103
  - random\_poisson, 104
  - random\_real, 105
  - rd, 106
  - twister, 105
- random.h, 106
  - random\_binomial, 107
  - random\_discrete, 108
  - random\_multinomial, 108, 109
  - random\_poisson, 110
  - random\_real, 111
- random\_binomial
  - random.cpp, 101

- random.h, 107
- random\_discrete
  - random.cpp, 102
  - random.h, 108
- random\_multinomial
  - random.cpp, 103
  - random.h, 108, 109
- random\_poisson
  - random.cpp, 104
  - random.h, 110
- random\_real
  - random.cpp, 105
  - random.h, 111
- rd
  - random.cpp, 106
- read\_and\_validate\_type
  - input.h, 91
- rec\_end
  - Record, 68
  - RecordParams, 69
- rec\_interval\_global
  - Record, 68
  - RecordParams, 69
- rec\_interval\_local
  - Record, 68
  - RecordParams, 69
- rec\_params
  - Simulation, 76
- rec\_sites\_freq
  - Record, 68
  - RecordParams, 69
- rec\_start
  - Record, 68
  - RecordParams, 70
- Record, 64
  - ~Record, 65
  - coord\_list, 67
  - global\_data, 67
  - is\_rec\_global\_time, 65
  - is\_rec\_local\_time, 65
  - local\_data, 67
  - os1, 67
  - os2, 68
  - os3, 68
  - output\_totals, 66
  - rec\_end, 68
  - rec\_interval\_global, 68
  - rec\_interval\_local, 68
  - rec\_sites\_freq, 68
  - rec\_start, 68
  - Record, 65
  - record\_coords, 66
  - record\_global, 66
  - record\_local, 67
  - rep\_label, 68
  - set\_label, 68
- Record.cpp, 112
- Record.h, 112
- record\_coords
  - Record, 66
- record\_global
  - Record, 66
- record\_local
  - Record, 67
- RecordParams, 69
  - rec\_end, 69
  - rec\_interval\_global, 69
  - rec\_interval\_local, 69
  - rec\_sites\_freq, 69
  - rec\_start, 70
  - set\_label, 70
- rel\_params
  - Simulation, 76
- release\_gene\_drive
  - GDRelease, 31
- ReleaseParams, 70
  - driver\_start, 70
  - num\_driver\_M, 70
  - num\_driver\_sites, 71
- rep\_label
  - Record, 68
- run
  - Model, 42
- run\_reps
  - Simulation, 72
- run\_step
  - Model, 43
- select\_driver\_sites
  - GDRelease, 32
- set\_boundary\_type
  - Simulation, 73
- set\_connecs
  - Dispersal, 22
  - DistanceKernelDispersal, 28
  - RadialDispersal, 63
- set\_coords
  - Simulation, 74
- set\_dev\_duration\_probs
  - Model, 44
- set\_dispersal\_type
  - Simulation, 74
- set\_inheritance
  - Simulation, 74
- set\_label
  - Record, 68
  - RecordParams, 70
- side
  - AreaParams, 16
  - BoundaryStrategy, 18
  - Model, 45
- Simulation, 71
  - aes\_params, 75
  - area\_params, 75
  - boundary\_type, 75
  - disp\_params, 75
  - disp\_type, 75

- inher\_fraction, [75](#)
  - initial\_params, [75](#)
  - life\_params, [76](#)
  - max\_t, [76](#)
  - num\_runs, [76](#)
  - rec\_params, [76](#)
  - rel\_params, [76](#)
  - run\_reps, [72](#)
  - set\_boundary\_type, [73](#)
  - set\_coords, [74](#)
  - set\_dispersal\_type, [74](#)
  - set\_inheritance, [74](#)
  - Simulation, [72](#)
  - sites\_coords, [76](#)
- Simulation.cpp, [114](#)
  - invalid\_interval\_msg, [114](#)
  - out\_of\_bounds\_msg, [115](#)
- Simulation.h, [115](#)
  - invalid\_interval\_msg, [116](#)
  - out\_of\_bounds\_msg, [116](#)
- sites
  - Model, [46](#)
- sites\_coords
  - Simulation, [76](#)
- t\_hide1
  - Aestivation, [14](#)
  - AestivationParams, [15](#)
- t\_hide2
  - Aestivation, [14](#)
  - AestivationParams, [15](#)
- t\_wake1
  - Aestivation, [14](#)
  - AestivationParams, [16](#)
- t\_wake2
  - Aestivation, [14](#)
  - AestivationParams, [16](#)
- theta
  - LifeParams, [36](#)
- Toroid
  - constants.h, [84](#)
- ToroidalBoundaryStrategy, [77](#)
  - distance, [78](#)
  - ToroidalBoundaryStrategy, [78](#)
- twister
  - random.cpp, [105](#)
- TWOPI
  - Dispersal.cpp, [85](#)
- update\_comp
  - Patch, [53](#)
- update\_mate
  - Patch, [54](#)
- V
  - Patch, [56](#)
- virgins\_mate
  - Model, [44](#)
  - Patch, [55](#)
- wake
  - Aestivation, [13](#)
- wrap\_around
  - RadialDispersal, [63](#)
- x
  - Point, [57](#)
- xi
  - InheritanceParams, [34](#)
- y
  - Point, [57](#)