

epiworld

0.0-1

Generated by Doxygen 1.9.1



<b>1 Main Page</b>	<b>1</b>
1.1 epiworld	1
1.2 Hello world	1
1.2.1 Tools	2
1.2.2 Contagion	2
<b>2 Class Index</b>	<b>3</b>
2.1 Class List	3
<b>3 Class Documentation</b>	<b>5</b>
3.1 Action< TSeq > Struct Template Reference	5
3.1.1 Detailed Description	6
3.1.2 Constructor & Destructor Documentation	6
3.1.2.1 Action()	6
3.2 AdjList Class Reference	7
3.2.1 Constructor & Destructor Documentation	7
3.2.1.1 AdjList()	7
3.3 Agent< TSeq > Class Template Reference	8
3.3.1 Detailed Description	9
3.4 DataBase< TSeq > Class Template Reference	10
3.4.1 Detailed Description	11
3.4.2 Member Function Documentation	11
3.4.2.1 record_variant()	11
3.5 LFMCMC< TData > Class Template Reference	11
3.5.1 Detailed Description	12
3.6 Location< TSeq > Class Template Reference	13
3.7 Model< TSeq > Class Template Reference	13
3.7.1 Detailed Description	19
3.7.2 Member Function Documentation	19
3.7.2.1 add_global_action()	19
3.7.2.2 reset()	19
3.7.2.3 run_multiple()	20
3.7.2.4 write_data()	20
3.8 PersonTools< TSeq > Class Template Reference	21
3.9 Progress Class Reference	21
3.9.1 Detailed Description	21
3.10 Queue< TSeq > Class Template Reference	21
3.10.1 Detailed Description	21
3.11 RandGraph Class Reference	22
3.12 Tool< TSeq > Class Template Reference	22
3.12.1 Detailed Description	23
3.13 Tools< TSeq > Class Template Reference	24
3.13.1 Detailed Description	24

---

3.14 Tools_const< TSeq > Class Template Reference . . . . .	24
3.14.1 Detailed Description . . . . .	25
3.15 UserData< TSeq > Class Template Reference . . . . .	25
3.15.1 Detailed Description . . . . .	26
3.15.2 Constructor & Destructor Documentation . . . . .	26
3.15.2.1 UserData() . . . . .	26
3.16 vecHasher< T > Struct Template Reference . . . . .	27
3.16.1 Detailed Description . . . . .	27
3.17 Virus< TSeq > Class Template Reference . . . . .	27
3.17.1 Detailed Description . . . . .	29
3.18 Viruses< TSeq > Class Template Reference . . . . .	29
3.18.1 Detailed Description . . . . .	30
3.19 Viruses_const< TSeq > Class Template Reference . . . . .	30
3.19.1 Detailed Description . . . . .	30
<b>Index</b>	<b>33</b>

# Chapter 1

## Main Page

### 1.1 epiworld

This C++ library provides a general framework for epidemiologic simulation. The core principle of `epiworld` is fast epidemiological prototyping for building complex models quickly. Here are some of its main features:

- It only depends on the standard library (C++11 required.)
- It is a template library.
- It is header-only ( [single file](#)).
- Models can have an arbitrary set of states.
- [Viruses](#) and tools (e.g., vaccines, mask-wearing) can be designed to have arbitrary features.
- Multiple tools and viruses can live in the same simulation.
- It is *FAST*: About 7.5 Million person/day simulations per second (see example below).

Various examples can be found in the `[examples](examples)` folder.

### 1.2 Hello world

Here is a simple SIR model implemented with `epiworld`. The source code can be found [here](#), and you can compile the code as follows:

```
g++ -std=c++17 -O2 readme.cpp -o readme.o
```

As you can see in [readme.cpp](#), to use `epiworld` you only need to incorporate the single header file `epiworld.hpp`:

```
#include "epiworld.hpp"
using namespace epiworld;
int main()
{
    // Creating a model with three statuses:
    // - Susceptible: Status 0
    // - Infected: Status 1
    // - Recovered: Status 2
```

```

Model<> model;
model.add_status("Susceptible", default_update_susceptible<>);
model.add_status("Infected", default_update_exposed<>);
model.add_status("Recovered");
// Desgining a virus: This virus will:
// - Have a 90% transmission rate
// - Have a 50% recovery rate
// - Infected individuals become "Infected" (status 1)
// - Recovered individuals become "Recovered" (status 2)
// Only five individuals will have the virus from the beginning.
Virus<> virus("covid 19");
virus.set_prob_infecting(.9);
virus.set_prob_recover(.5);

virus.set_status(1, 2);
model.add_virus_n(virus, 5);

// Generating a random pop from a smallworld network
model.population_smallworld(100000);
// Initializing setting days and seed
model.init(100, 123);
// Running the model
model.run();
model.print();
}

```

And you should get something like the following:

Running the model...

```

||||||||||||||||||||||||||||||||||||||||||||||||| done.
SIMULATION STUDY
Population size      : 100000
Days (duration)     : 100 (of 100)
Number of variants  : 1
Last run elapsed t   : 134.00ms
Rewiring             : off
Virus(es):
- covid 19 (baseline prevalence: 5 seeds)
Tool(s):
(none)
Model parameters:
(none)
Distribution of the population at time 100:
- (0) Total Susceptible : 99995 -> 95466
- (1) Total Infected    :      5 -> 70
- (2) Total Recovered   :      0 -> 4464

```

Which took about 0.134 seconds ( $\sim 7.5$  million ppl x day / second).

## 1.2.1 Tools

## 1.2.2 Contagion

Susceptible individuals can acquire a virus from any of their infected connections. The probability that susceptible individual  $i$  gets the virus  $v$  from individual  $j$  depends on how three things:

1. The transmissibility of the virus,  $\beta_v$ ,
2. The contagion reduction factor of  $i$ ,  $\gamma_i$ , and
3. The host's transmission reduction factor,  $\gamma_j$ .

The last two are computed from and  $i$ 's tools. Ultimately, the probability of getting virus  $v$  from  $j$  equals:

Nonetheless, the default behavior of the simulation model is to assume that individuals can acquire only one disease at a time, if any. This way, the actual probability is:

The latter is calculated using Bayes' rule

Where

This way, viruses with higher transmissibility will be more likely to be acquired when competing with other variants.

## Chapter 2

# Class Index

### 2.1 Class List

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

<a href="#">Action&lt; TSeq &gt;</a>	
<a href="#">Action</a> data for update an agent	5
<a href="#">AdjList</a>	7
<a href="#">Agent&lt; TSeq &gt;</a>	
<a href="#">Agent</a> (agents)	8
<a href="#">DataBase&lt; TSeq &gt;</a>	
Statistical data about the process	10
<a href="#">LFMCMC&lt; TData &gt;</a>	
Likelihood-Free Markov Chain Monte Carlo	11
<a href="#">Location&lt; TSeq &gt;</a>	13
<a href="#">Model&lt; TSeq &gt;</a>	
Core class of epiworld	13
<a href="#">PersonTools&lt; TSeq &gt;</a>	21
<a href="#">Progress</a>	
A simple progress bar	21
<a href="#">Queue&lt; TSeq &gt;</a>	
Controls which agents are verified at each step	21
<a href="#">RandGraph</a>	22
<a href="#">Tool&lt; TSeq &gt;</a>	
Tools for defending the agent against the virus	22
<a href="#">Tools&lt; TSeq &gt;</a>	
Set of tools (useful for building iterators)	24
<a href="#">Tools_const&lt; TSeq &gt;</a>	
Set of <a href="#">Tools</a> (const) (useful for iterators)	24
<a href="#">UserData&lt; TSeq &gt;</a>	
Personalized data by the user	25
<a href="#">vecHasher&lt; T &gt;</a>	
Vector hasher	27
<a href="#">Virus&lt; TSeq &gt;</a>	
<a href="#">Virus</a>	27
<a href="#">Viruses&lt; TSeq &gt;</a>	
Set of viruses (useful for building iterators)	29
<a href="#">Viruses_const&lt; TSeq &gt;</a>	
Set of <a href="#">Viruses</a> (const) (useful for iterators)	30





## Chapter 3

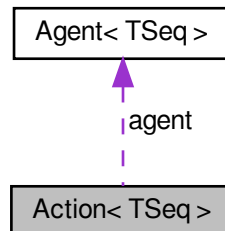
# Class Documentation

### 3.1 Action< TSeq > Struct Template Reference

[Action](#) data for update an agent.

```
#include <config.hpp>
```

Collaboration diagram for Action< TSeq >:



#### Public Member Functions

- [Action](#) ([Agent](#)< TSeq > \*agent\_, VirusPtr< TSeq > virus\_, ToolPtr< TSeq > tool\_, epiworld\_fast\_int new\_status\_, epiworld\_fast\_int queue\_, ActionFun< TSeq > call\_)

Construct a new [Action](#) object.

#### Public Attributes

- [Agent](#)< TSeq > \* **agent**
- VirusPtr< TSeq > **virus**
- ToolPtr< TSeq > **tool**
- epiworld\_fast\_int **new\_status**
- epiworld\_fast\_int **queue**
- ActionFun< TSeq > **call**

### 3.1.1 Detailed Description

```
template<typename TSeq>
struct Action< TSeq >
```

[Action](#) data for update an agent.

Template Parameters

<i>TSeq</i>	
-------------	--

### 3.1.2 Constructor & Destructor Documentation

#### 3.1.2.1 Action()

```
template<typename TSeq >
Action< TSeq >::Action (
    Agent< TSeq > * agent_,
    VirusPtr< TSeq > virus_,
    ToolPtr< TSeq > tool_,
    epiworld_fast_int new_status_,
    epiworld_fast_int queue_,
    ActionFun< TSeq > call_ ) [inline]
```

Construct a new [Action](#) object.

All the parameters are rather optional.

Parameters

<i>agent_</i>	<a href="#">Agent</a> over who the action will happen
<i>virus_</i>	<a href="#">Virus</a> to add
<i>tool_</i>	<a href="#">Tool</a> to add
<i>virus_idx</i>	Index of virus to be removed (if needed)
<i>tool_idx</i>	Index of tool to be removed (if needed)
<i>new_↔ status_</i>	Next status
<i>queue_</i>	Effect on the queue
<i>call_</i>	The action call (if needed)

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

- include/epiworld/agent-bones.hpp
- include/epiworld/config.hpp

## 3.2 AdjList Class Reference

### Public Member Functions

- [AdjList](#) (const std::vector< unsigned int > &source, const std::vector< unsigned int > &target, bool directed, int min\_id=-1, int max\_id=-1)  
*Construct a new Adj List object.*
- void **read\_edgelist** (std::string fn, int skip=0, bool directed=true, int min\_id=-1, int max\_id=-1)
- std::map< unsigned int, unsigned int > **operator()** (unsigned int i) const
- void **print** (unsigned int limit=20u) const
- unsigned int **get\_id\_max** () const
- unsigned int **get\_id\_min** () const
- size\_t **vcount** () const
- size\_t **ecount** () const
- std::map< unsigned int, std::map< unsigned int, unsigned int > > & **get\_dat** ()
- bool **is\_directed** () const

### 3.2.1 Constructor & Destructor Documentation

#### 3.2.1.1 AdjList()

```
AdjList::AdjList (
    const std::vector< unsigned int > & source,
    const std::vector< unsigned int > & target,
    bool directed,
    int min_id = -1,
    int max_id = -1 ) [inline]
```

Construct a new Adj List object.

It will create an adjacency list object with `maxid - minid + 1` nodes. If `min_id` and `max_id` are not specified (both < 0), then the program will try to figure them out automatically by looking at the range of the observed ids.

#### Parameters

<i>source</i>	Unsigned int vector with the source
<i>target</i>	Unsigned int vector with the target
<i>directed</i>	Bool true if the network is directed
<i>min_id</i>	int min id.
<i>max_id</i>	int max id.

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

- include/epiworld/adjlist-bones.hpp
- include/epiworld/adjlist-meat.hpp

### 3.3 Agent< TSeq > Class Template Reference

[Agent](#) (agents)

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

#### Public Member Functions

- **Agent** (const [Agent](#)< TSeq > &p)
- int [get\\_id](#) () const  
*Id of the individual.*
- unsigned int [get\\_index](#) () const  
*Location (0, ..., n-1).*
- std::mt19937 \* [get\\_rand\\_engine](#) ()
- [Model](#)< TSeq > \* [get\\_model](#) ()
- VirusPtr< TSeq > & [get\\_virus](#) (int i)
- [Viruses](#)< TSeq > [get\\_viruses](#) ()
- const [Viruses\\_const](#)< TSeq > [get\\_viruses](#) () const
- size\_t [get\\_n\\_viruses](#) () const noexcept
- ToolPtr< TSeq > & [get\\_tool](#) (int i)
- [Tools](#)< TSeq > [get\\_tools](#) ()
- const [Tools\\_const](#)< TSeq > [get\\_tools](#) () const
- size\_t [get\\_n\\_tools](#) () const noexcept
- void [mutate\\_variant](#) ()
- void [add\\_neighbor](#) ([Agent](#)< TSeq > \*p, bool check\_source=true, bool check\_target=true)
- std::vector< [Agent](#)< TSeq > \* > & [get\\_neighbors](#) ()
- void [change\\_status](#) (epiworld\_fast\_uint new\_status, epiworld\_fast\_int queue=0)
- const epiworld\_fast\_uint & [get\\_status](#) () const
- void [reset](#) ()
- bool [has\\_tool](#) (unsigned int t) const
- bool [has\\_tool](#) (std::string name) const
- bool [has\\_virus](#) (unsigned int t) const
- bool [has\\_virus](#) (std::string name) const

#### Add/Remove Virus/Tool

*Any of these is ultimately reflected at the end of the iteration.*

#### Parameters

tool	<a href="#">Tool</a> to add
virus	<a href="#">Virus</a> to add
status_new	Status after the change
queue	

- void [add\\_tool](#) (ToolPtr< TSeq > tool, epiworld\_fast\_int status\_new=-99, epiworld\_fast\_int queue=-99)
- void [add\\_tool](#) ([Tool](#)< TSeq > tool, epiworld\_fast\_int status\_new=-99, epiworld\_fast\_int queue=-99)
- void [add\\_virus](#) (VirusPtr< TSeq > virus, epiworld\_fast\_int status\_new=-99, epiworld\_fast\_int queue=-99)
- void [add\\_virus](#) ([Virus](#)< TSeq > virus, epiworld\_fast\_int status\_new=-99, epiworld\_fast\_int queue=-99)
- void [rm\\_tool](#) (epiworld\_fast\_uint tool\_idx, epiworld\_fast\_int status\_new=-99, epiworld\_fast\_int queue=-99)

- void **rm\_tool** (ToolPtr< TSeq > &tool, epiworld\_fast\_int status\_new=-99, epiworld\_fast\_int queue=-99)
- void **rm\_virus** (epiworld\_fast\_uint virus\_idx, epiworld\_fast\_int status\_new=-99, epiworld\_fast\_int queue=-99)
- void **rm\_virus** (VirusPtr< TSeq > &virus, epiworld\_fast\_int status\_new=-99, epiworld\_fast\_int queue=-99)

#### Get the rates (multipliers) for the agent

##### Parameters

v	A pointer to a virus.
---	-----------------------

##### Returns

*epiworld\_double*

- epiworld\_double **get\_susceptibility\_reduction** (VirusPtr< TSeq > v)
- epiworld\_double **get\_transmission\_reduction** (VirusPtr< TSeq > v)
- epiworld\_double **get\_recovery\_enhancer** (VirusPtr< TSeq > v)
- epiworld\_double **get\_death\_reduction** (VirusPtr< TSeq > v)

#### Friends

- class **Model**< TSeq >
- class **Virus**< TSeq >
- class **Viruses**< TSeq >
- class **Viruses\_const**< TSeq >
- class **Tool**< TSeq >
- class **Tools**< TSeq >
- class **Queue**< TSeq >
- void **default\_add\_virus** (Action< TSeq > &a, Model< TSeq > \*m)
- void **default\_add\_tool** (Action< TSeq > &a, Model< TSeq > \*m)
- void **default\_rm\_virus** (Action< TSeq > &a, Model< TSeq > \*m)
- void **default\_rm\_tool** (Action< TSeq > &a, Model< TSeq > \*m)

### 3.3.1 Detailed Description

```
template<typename TSeq = int>
class Agent< TSeq >
```

[Agent](#) (agents)

##### Template Parameters

<i>TSeq</i>	Sequence type (should match TSeq across the model)
-------------	--

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

- include/epiworld/agent-bones.hpp

### 3.4 DataBase< TSeq > Class Template Reference

Statistical data about the process.

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

#### Public Member Functions

- **DataBase** ([Model](#)< TSeq > &m)
- void [record\\_variant](#) ([Virus](#)< TSeq > &v)  
*Registering a new variant.*
- void **record\_tool** ([Tool](#)< TSeq > &t)
- void **set\_seq\_hasher** (std::function< std::vector< int >(TSeq)> fun)
- void **set\_model** ([Model](#)< TSeq > &m)
- [Model](#)< TSeq > \* **get\_model** ()
- void **record** ()
- const std::vector< TSeq > & **get\_sequence** () const
- const std::vector< int > & **get\_nexposed** () const
- size\_t **size** () const
- void **write\_data** (std::string fn\_variant\_info, std::string fn\_variant\_hist, std::string fn\_tool\_info, std::string fn\_tool\_hist, std::string fn\_total\_hist, std::string fn\_transmission, std::string fn\_transition) const
- void **record\_transmission** (int i, int j, int variant)
- size\_t **get\_n\_variants** () const
- size\_t **get\_n\_tools** () const
- void **reset** ()
- void **set\_user\_data** (std::vector< std::string > names)
- void **add\_user\_data** (std::vector< epiworld\_double > x)
- void **add\_user\_data** (unsigned int j, epiworld\_double x)
- [UserData](#)< TSeq > & **get\_user\_data** ()

#### Get recorded information from the model

##### Parameters

what	<i>std::string, The status, e.g., 0, 1, 2, ...</i>
------	--

##### Returns

*In **get\_today\_total**, the current counts of what.*

*In **get\_today\_variant**, the current counts of what for each variant.*

*In **get\_hist\_total**, the time series of what*

*In **get\_hist\_variant**, the time series of what for each variant.*

*In **get\_hist\_total\_date** and **get\_hist\_variant\_date** the corresponding dates*

- int **get\_today\_total** (std::string what) const
- int **get\_today\_total** (epiworld\_fast\_uint what) const
- void **get\_today\_total** (std::vector< std::string > \*status=nullptr, std::vector< int > \*counts=nullptr) const
- void **get\_today\_variant** (std::vector< std::string > &status, std::vector< int > &id, std::vector< int > &counts) const
- void **get\_hist\_total** (std::vector< int > \*date, std::vector< std::string > \*status, std::vector< int > \*counts) const
- void **get\_hist\_variant** (std::vector< int > &date, std::vector< int > &id, std::vector< std::string > &status, std::vector< int > &counts) const

## Friends

- class **Model**< TSeq >
- void **default\_add\_virus** ([Action](#)< TSeq > &a, [Model](#)< TSeq > \*m)
- void **default\_add\_tool** ([Action](#)< TSeq > &a, [Model](#)< TSeq > \*m)
- void **default\_rm\_virus** ([Action](#)< TSeq > &a, [Model](#)< TSeq > \*m)
- void **default\_rm\_tool** ([Action](#)< TSeq > &a, [Model](#)< TSeq > \*m)

### 3.4.1 Detailed Description

```
template<typename TSeq>
class DataBase< TSeq >
```

Statistical data about the process.

#### Template Parameters

<i>TSeq</i>	
-------------	--

### 3.4.2 Member Function Documentation

#### 3.4.2.1 record\_variant()

```
template<typename TSeq >
void DataBase< TSeq >::record_variant (
    Virus< TSeq > & v ) [inline]
```

Registering a new variant.

#### Parameters

<i>v</i>	Pointer to the new variant. Since variants are originated in the agent, the numbers simply move around. From the parent variant to the new variant. And the total number of infected does not change.
----------	---

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

- include/epiworld/database-bones.hpp
- include/epiworld/database-meat.hpp

## 3.5 LFMCMC< TData > Class Template Reference

Likelihood-Free Markov Chain Monte Carlo.

```
#include <lfmcmc.hpp>
```

## Public Member Functions

- void **run** (VEC(epiworld\_double) param\_init, size\_t n\_samples\_, epiworld\_double epsilon\_)
- **LFMCMC** (TData &observed\_data\_)
- void **set\_observed\_data** (TData &observed\_data\_)
- void **set\_proposal\_fun** (FUN< void(VEC(epiworld\_double)&, LFMCMC< TData > \*)> fun)
- void **set\_simulation\_fun** (FUN< TData(VEC(epiworld\_double)&, LFMCMC< TData > \*)> fun)
- void **set\_summary\_fun** (FUN< VEC(epiworld\_double)(TData &, LFMCMC< TData > \*)> fun)
- void **set\_kernel\_fun** (FUN< epiworld\_double(VEC(epiworld\_double)&, epiworld\_double, LFMCMC< TData > \*)> fun)
- const size\_t **get\_n\_samples** ()
- const size\_t **get\_n\_statistics** ()
- const size\_t **get\_n\_parameters** ()
- const epiworld\_double **get\_epsilon** ()
- const **VEC** (epiworld\_double) &get\_params\_now()
- const **VEC** (epiworld\_double) &get\_params\_prev()
- const **VEC** (epiworld\_double) &get\_params\_init()
- const **VEC** (epiworld\_double) &get\_statistics\_obs()
- const **VEC** (epiworld\_double) &get\_statistics\_hist()
- const **VEC** (bool) &get\_statistics\_accepted()
- const **VEC** (epiworld\_double) &get\_posterior\_if\_prob()
- const **VEC** (epiworld\_double) &get\_acceptance\_prob()
- const **VEC** (epiworld\_double) &get\_drawn\_prob()
- **VEC** (TData) \*get\_sampled\_data()

## Random number generation

### Parameters

eng	
-----	--

- void **set\_rand\_engine** (std::mt19937 &eng)
- std::mt19937 \* **get\_rand\_engine** ()
- void **seed** (unsigned int s)
- void **set\_rand\_gamma** (epiworld\_double alpha, epiworld\_double beta)
- epiworld\_double **runif** ()
- epiworld\_double **rnorm** ()
- epiworld\_double **rnorm** (epiworld\_double mean, epiworld\_double sd)
- epiworld\_double **rgamma** ()
- epiworld\_double **rgamma** (epiworld\_double alpha, epiworld\_double beta)

## 3.5.1 Detailed Description

```
template<typename TData>
class LFMCMC< TData >
```

Likelihood-Free Markov Chain Monte Carlo.

### Template Parameters

<i>TData</i>	Type of data that is generated
--------------	--------------------------------



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

- include/epiworld/math/lfmcmmc.hpp

## 3.6 Location< TSeq > Class Template Reference

### Public Member Functions

- **add\_agent** ([Agent](#)< TSeq > &p)
- **add\_agent** ([Agent](#)< TSeq > \*p)
- **size\_t count** () const
- **void reset** ()

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

- include/epiworld/location-bones.hpp

## 3.7 Model< TSeq > Class Template Reference

Core class of epiworld.

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

### Public Member Functions

- [DataBase](#)< TSeq > & **get\_db** ()
- epiworld\_double & **operator()** (std::string pname)
- **size\_t size** () const
- **size\_t get\_n\_variants** () const
- **size\_t get\_n\_tools** () const
- **unsigned int get\_ndays** () const
- **unsigned int get\_n\_replicates** () const
- **void set\_ndays** (unsigned int ndays)
- **bool get\_verbose** () const
- **void verbose\_off** ()
- **void verbose\_on** ()
- **int today** () const
- *The current time of the model.*
- **void write\_data** (std::string fn\_variant\_info, std::string fn\_variant\_hist, std::string fn\_tool\_info, std::string fn\_tool\_hist, std::string fn\_total\_hist, std::string fn\_transmission, std::string fn\_transition) const
- *Wrapper of DataBase::write\_data*
- **std::map< std::string, epiworld\_double > & params** ()
- **void reset** ()
- *Reset the model.*
- **void print** () const
- [Model](#)< TSeq > && **clone** () const
- **void get\_elapsed** (std::string unit="auto", epiworld\_double \*last\_elapsed=nullptr, epiworld\_double \*total\_elapsed=nullptr, std::string \*unit\_abbr=nullptr, bool print=true) const

- void **add\_global\_action** (std::function< void([Model](#)< TSeq > \*)> fun, int date=-99)  
*Set a global action.*
- void **run\_global\_actions** ()
- void **clear\_status\_set** ()
- const std::vector< VirusPtr< TSeq > > & **get\_viruses** () const
- const std::vector< ToolPtr< TSeq > > & **get\_tools** () const

### Set the backup object

*backup* can be used to restore the entire object after a run. This can be useful if the user wishes to have individuals start with the same network from the beginning.

- void **set\_backup** ()
- void **restore\_backup** ()

### Random number generation

#### Parameters

eng	<i>Random number generator</i>
s	<i>Seed</i>

- void **set\_rand\_engine** (std::mt19937 &eng)
- std::mt19937 \* **get\_rand\_engine** ()
- void **seed** (unsigned int s)
- void **set\_rand\_gamma** (epiworld\_double alpha, epiworld\_double beta)
- epiworld\_double **runif** ()
- epiworld\_double **rnorm** ()
- epiworld\_double **rnorm** (epiworld\_double mean, epiworld\_double sd)
- epiworld\_double **rgamma** ()
- epiworld\_double **rgamma** (epiworld\_double alpha, epiworld\_double beta)

### Add Virus/Tool to the model

*This is done before the model has been initialized.*

#### Parameters

v	<i><a href="#">Virus</a> to be added</i>
t	<i><a href="#">Tool</a> to be added</i>
preval	<i>Initial prevalence (initial state.) It can be specified as a proportion (between zero and one,) or an integer indicating number of individuals.</i>

- void **add\_virus** ([Virus](#)< TSeq > v, epiworld\_double preval)
- void **add\_virus\_n** ([Virus](#)< TSeq > v, unsigned int preval)
- void **add\_tool** ([Tool](#)< TSeq > t, epiworld\_double preval)
- void **add\_tool\_n** ([Tool](#)< TSeq > t, unsigned int preval)

### Accessing population of the model

#### Parameters

fn	<i>std::string Filename of the edgelist file.</i>
skip	<i>int Number of lines to skip in fn.</i>

*Parameters*

directed	<i>bool Whether the graph is directed or not.</i>
min_id	<i>int Minimum id number (if negative, the program will try to guess from the data.)</i>
max_id	<i>int Maximum id number (if negative, the program will try to guess from the data.)</i>
al	<i><a href="#">AdjList</a> to read into the model.</i>

- void **population\_from\_adjlist** (std::string fn, int skip=0, bool directed=false, int min\_id=-1, int max\_id=-1)
- void **population\_from\_adjlist** ([AdjList](#) al)
- bool **is\_directed** () const
- std::vector< [Agent](#)< TSeq > > \* **get\_population** ()
- void **population\_smallworld** (unsigned int n=1000, unsigned int k=5, bool d=false, epiworld\_double p=.01)

**Functions to run the model***Parameters*

seed	<i>Seed to be used for Pseudo-RNG.</i>
ndays	<i>Number of days (steps) of the simulation.</i>
fun	<i>In the case of <code>run_multiple</code>, a function that is called after each experiment.</i>

- void **init** (unsigned int ndays, unsigned int seed)
- void **update\_status** ()
- void **mutate\_variant** ()
- void **next** ()
- void **run** ()  
*Runs the simulation (after initialization)*
- void **run\_multiple** (unsigned int nexperiments, std::function< void([Model](#)< TSeq > \*)> fun, bool [reset](#), bool verbose)

**Rewire the network preserving the degree sequence.**

*This implementation assumes an undirected network, thus if  $\{(i,j), (k,l)\} \rightarrow \{(i,l), (k,j)\}$ , the reciprocal is also true, i.e.,  $\{(j,i), (l,k)\} \rightarrow \{(j,k), (l,i)\}$ .*

*Parameters*

proportion	<i>Proportion of ties to be rewired.</i>
------------	--

*Returns*

*A rewired version of the network.*

- void **set\_rewire\_fun** (std::function< void(std::vector< [Agent](#)< TSeq > > \*, [Model](#)< TSeq > \*, epiworld\_double)> fun)
- void **set\_rewire\_prop** (epiworld\_double prop)
- epiworld\_double **get\_rewire\_prop** () const
- void **rewire** ()

**Export the network data in edgelist form***Parameters*

fn	<i>std::string. File name.</i>
----	--------------------------------

**Parameters**

source	<i>Integer vector</i>
target	<i>Integer vector</i>

When passing the source and target, the function will write the edgelist on those.

- void **write\_edgelist** (std::string fn) const
- void **write\_edgelist** (std::vector< unsigned int > &source, std::vector< unsigned int > &target) const

**Manage status (states) in the model**

The functions *get\_status* return the current values for the statuses included in the model.

**Parameters**

lab	<i>std::string Name of the status.</i>
-----	--

**Returns**

*add\_status\** returns nothing.  
*get\_status\_\** returns a vector of pairs with the statuses and their labels.

- void **add\_status** (std::string lab, UpdateFun< TSeq > fun=nullptr)
- const std::vector< std::string > & **get\_status** () const
- const std::vector< UpdateFun< TSeq > > & **get\_status\_fun** () const
- void **print\_status\_codes** () const

**Set the user data object****Parameters**

names	<i>string vector with the names of the variables.</i>
-------	---

- void **set\_user\_data** (std::vector< std::string > names)
- void **add\_user\_data** (unsigned int j, epiworld\_double x)
- void **add\_user\_data** (std::vector< epiworld\_double > x)
- **UserData**< TSeq > & **get\_user\_data** ()

**Queuing system**

When queueing is on, the model will keep track of which agents are either in risk of exposure or exposed. This then is used at each step to act only on the aforementioned agents.

- void **queueing\_on** ()  
*Activates the queueing system (default.)*
- void **queueing\_off** ()  
*Deactivates the queueing system.*
- bool **is\_queueing\_on** () const  
*Query if the queueing system is on.*
- **Queue**< TSeq > & **get\_queue** ()  
*Retrieve the **Queue** object.*

**Get the susceptibility reduction object**

#### Parameters

v	
---	--

#### Returns

*epiworld\_double*

- void **set\_susceptibility\_reduction\_mixer** (MixerFun< TSeq > fun)
- void **set\_transmission\_reduction\_mixer** (MixerFun< TSeq > fun)
- void **set\_recovery\_enhancer\_mixer** (MixerFun< TSeq > fun)
- void **set\_death\_reduction\_mixer** (MixerFun< TSeq > fun)

#### Friends

- class **Agent**< TSeq >
- class **DataBase**< TSeq >
- class **Queue**< TSeq >

#### Tool Mixers

These functions combine the effects tools have to deliver a single effect. For example, wearing a mask, been vaccinated, and the immune system combine together to jointly reduce the susceptibility for a given virus.

- std::vector< epiworld\_double > **array\_double\_tmp**
- std::vector< [Virus](#)< TSeq > \* > **array\_virus\_tmp**
- **Model** ()
- **Model** (const [Model](#)< TSeq > &m)
- **Model** ([Model](#)< TSeq > &&m)
- [Model](#)< TSeq > & **operator=** (const [Model](#)< TSeq > &m)
- void **clone\_population** (std::vector< [Agent](#)< TSeq > > &p, std::map< int, int > &p\_ids, bool &d, [Model](#)< TSeq > \*m=nullptr) const
- void **clone\_population** (const [Model](#)< TSeq > &m)

#### Setting and accessing parameters from the model

[Tools](#) can incorporate parameters included in the model. Internally, parameters in the tool are stored as pointers to an std::map<> of parameters in the model. Using the `unsigned int` method directly fetches the parameters in the order these were added to the tool. Accessing parameters via the `std::string` method involves searching the parameter directly in the std::map<> member of the model (so it is not recommended.)

The function `set_param()` can be used when the parameter already exists in the model.

The `par()` function members are aliases for `get_param()`.

#### Parameters

<i>initial_val</i>	
<i>pname</i>	Name of the parameter to add or to fetch

## Returns

The current value of the parameter in the model.

- `epiworld_double * p0`
- `epiworld_double * p1`
- `epiworld_double * p2`
- `epiworld_double * p3`
- `epiworld_double * p4`
- `epiworld_double * p5`
- `epiworld_double * p6`
- `epiworld_double * p7`
- `epiworld_double * p8`
- `epiworld_double * p9`
- `epiworld_double * p10`
- `epiworld_double * p11`
- `epiworld_double * p12`
- `epiworld_double * p13`
- `epiworld_double * p14`
- `epiworld_double * p15`
- `epiworld_double * p16`
- `epiworld_double * p17`
- `epiworld_double * p18`
- `epiworld_double * p19`
- `epiworld_double * p20`
- `epiworld_double * p21`
- `epiworld_double * p22`
- `epiworld_double * p23`
- `epiworld_double * p24`
- `epiworld_double * p25`
- `epiworld_double * p26`
- `epiworld_double * p27`
- `epiworld_double * p28`
- `epiworld_double * p29`
- `epiworld_double * p30`
- `epiworld_double * p31`
- `epiworld_double * p32`
- `epiworld_double * p33`
- `epiworld_double * p34`
- `epiworld_double * p35`
- `epiworld_double * p36`
- `epiworld_double * p37`
- `epiworld_double * p38`
- `epiworld_double * p39`
- `unsigned int npar_used = 0u`
- `epiworld_double add_param (epiworld_double initial_val, std::string pname)`
- `epiworld_double set_param (std::string pname)`
- `epiworld_double get_param (unsigned int k)`
- `epiworld_double get_param (std::string pname)`
- `epiworld_double par (unsigned int k)`
- `epiworld_double par (std::string pname)`

### 3.7.1 Detailed Description

```
template<typename TSeq = int>
class Model< TSeq >
```

Core class of epiworld.

The model class provides the wrapper that puts together [Agent](#), [Virus](#), and [Tools](#).

#### Template Parameters

<i>TSeq</i>	Type of sequence. In principle, users can build models in which virus and human sequence is represented as numeric vectors (if needed.)
-------------	---

### 3.7.2 Member Function Documentation

#### 3.7.2.1 add\_global\_action()

```
template<typename TSeq = int>
void Model< TSeq >::add_global_action (
    std::function< void (Model< TSeq > *)> fun,
    int date = -99 )
```

Set a global action.

#### Parameters

<i>fun</i>	A function to be called on the prescribed dates
<i>date</i>	Integer indicating when the function is called (see details)

When date is less than zero, then the function is called at the end of every day. Otherwise, the function will be called only at the end of the indicated date.

#### 3.7.2.2 reset()

```
template<typename TSeq = int>
void Model< TSeq >::reset ( )
```

Reset the model.

Resetting the model will:

- clear the database
- restore the population (if `set_backup()` was called before)

- re-distribute tools
- re-distribute viruses
- set the date to 0

### 3.7.2.3 run\_multiple()

```
template<typename TSeq = int>
void Model< TSeq >::run_multiple (
    unsigned int n_experiments,
    std::function< void(Model< TSeq > *)> fun,
    bool reset,
    bool verbose )
```

#### Parameters

<i>n_experiments</i>	Multiple runs of the simulation
----------------------	---------------------------------

### 3.7.2.4 write\_data()

```
template<typename TSeq = int>
void Model< TSeq >::write_data (
    std::string fn_variant_info,
    std::string fn_variant_hist,
    std::string fn_tool_info,
    std::string fn_tool_hist,
    std::string fn_total_hist,
    std::string fn_transmission,
    std::string fn_transition ) const
```

Wrapper of DataBase::write\_data

#### Parameters

<i>fn_variant_info</i>	Filename. Information about the variant.
<i>fn_variant_hist</i>	Filename. History of the variant.
<i>fn_tool_info</i>	Filename. Information about the tool.
<i>fn_tool_hist</i>	Filename. History of the tool.
<i>fn_total_hist</i>	Filename. Aggregated history (status)
<i>fn_transmission</i>	Filename. Transmission history.
<i>fn_transition</i>	Filename. Markov transition history.

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

- include/epiworld/agent-meat-status.hpp
- include/epiworld/model-bones.hpp



## 3.8 PersonTools< TSeq > Class Template Reference

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

- include/epiworld/config.hpp

## 3.9 Progress Class Reference

A simple progress bar.

```
#include <progress.hpp>
```

### Public Member Functions

- **Progress** (int n\_, int width\_)
- void **start** ()
- void **next** ()
- void **end** ()

### 3.9.1 Detailed Description

A simple progress bar.

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

- include/epiworld/progress.hpp

## 3.10 Queue< TSeq > Class Template Reference

Controls which agents are verified at each step.

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

### Public Member Functions

- void **operator+=** ([Agent](#)< TSeq > \*p)
- void **operator-=** ([Agent](#)< TSeq > \*p)
- epiworld\_fast\_int **operator[]** (unsigned int i) const
- void **set\_model** ([Model](#)< TSeq > \*m)

### 3.10.1 Detailed Description

```
template<typename TSeq = int>
class Queue< TSeq >
```

Controls which agents are verified at each step.

The idea is that only agents who are either in an infected state or have an infected neighbor should be checked. Otherwise it makes no sense (no chance to recover or capture the disease).

### Template Parameters

<i>TSeq</i>	
-------------	--

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

- include/epiworld/agent-bones.hpp
- include/epiworld/queue-bones.hpp

## 3.11 RandGraph Class Reference

### Public Member Functions

- **RandGraph** (int N\_)
- void **init** (int s)
- void **set\_rand\_engine** (std::mt19937 &e)
- epiworld\_double **runif** ()

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

- include/epiworld/random\_graph.hpp

## 3.12 Tool< TSeq > Class Template Reference

[Tools](#) for defending the agent against the virus.

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

### Public Member Functions

- **Tool** (std::string name="unknown tool")
- void **set\_sequence** (TSeq d)
- void **set\_sequence\_unique** (TSeq d)
- void **set\_sequence** (std::shared\_ptr< TSeq > d)
- std::shared\_ptr< TSeq > **get\_sequence** ()
- TSeq & **get\_sequence\_unique** ()
- void **set\_name** (std::string name)
- std::string **get\_name** () const
- [Agent](#)< TSeq > \* **get\_agent** ()
- int **get\_id** () const
- void **set\_id** (int id)
- void **set\_date** (int d)
- int **get\_date** () const
- void **set\_status** (epiworld\_fast\_int init, epiworld\_fast\_int post)
- void **set\_queue** (epiworld\_fast\_int init, epiworld\_fast\_int post)
- void **get\_status** (epiworld\_fast\_int \*init, epiworld\_fast\_int \*post)
- void **get\_queue** (epiworld\_fast\_int \*init, epiworld\_fast\_int \*post)

**Get and set the tool functions**

*Parameters*

v	<i>The virus over which to operate</i>
fun	<i>the function to be used</i>

*Returns**epiworld\_double*

- epiworld\_double **get\_susceptibility\_reduction** (VirusPtr< TSeq > v)
- epiworld\_double **get\_transmission\_reduction** (VirusPtr< TSeq > v)
- epiworld\_double **get\_recovery\_enhancer** (VirusPtr< TSeq > v)
- epiworld\_double **get\_death\_reduction** (VirusPtr< TSeq > v)
- void **set\_susceptibility\_reduction\_fun** (ToolFun< TSeq > fun)
- void **set\_transmission\_reduction\_fun** (ToolFun< TSeq > fun)
- void **set\_recovery\_enhancer\_fun** (ToolFun< TSeq > fun)
- void **set\_death\_reduction\_fun** (ToolFun< TSeq > fun)
- void **set\_susceptibility\_reduction** (epiworld\_double \*prob)
- void **set\_transmission\_reduction** (epiworld\_double \*prob)
- void **set\_recovery\_enhancer** (epiworld\_double \*prob)
- void **set\_death\_reduction** (epiworld\_double \*prob)
- void **set\_susceptibility\_reduction** (epiworld\_double prob)
- void **set\_transmission\_reduction** (epiworld\_double prob)
- void **set\_recovery\_enhancer** (epiworld\_double prob)
- void **set\_death\_reduction** (epiworld\_double prob)

**Friends**

- class **Agent**< TSeq >
- class **Model**< TSeq >
- void **default\_add\_virus** ([Action](#)< TSeq > &a, [Model](#)< TSeq > \*m)
- void **default\_add\_tool** ([Action](#)< TSeq > &a, [Model](#)< TSeq > \*m)
- void **default\_rm\_virus** ([Action](#)< TSeq > &a, [Model](#)< TSeq > \*m)
- void **default\_rm\_tool** ([Action](#)< TSeq > &a, [Model](#)< TSeq > \*m)

**3.12.1 Detailed Description**

```
template<typename TSeq = int>
class Tool< TSeq >
```

[Tools](#) for defending the agent against the virus.

**Template Parameters**

<i>TSeq</i>	Type of sequence
-------------	------------------

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

- include/epiworld/agent-bones.hpp
- include/epiworld/tool-bones.hpp
- include/epiworld/tool-meat.hpp

### 3.13 Tools< TSeq > Class Template Reference

Set of tools (useful for building iterators)

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

#### Public Member Functions

- **Tools** ([Agent](#)< TSeq > &p)
- std::vector< TOOLPTR >::iterator **begin** ()
- std::vector< TOOLPTR >::iterator **end** ()
- TOOLPTR & **operator**() (size\_t i)
- TOOLPTR & **operator**[] (size\_t i)
- size\_t **size** () const noexcept

#### Friends

- class **Tool**< TSeq >
- class **Agent**< TSeq >

#### 3.13.1 Detailed Description

```
template<typename TSeq>
class Tools< TSeq >
```

Set of tools (useful for building iterators)

Template Parameters

<i>TSeq</i>	
-------------	--

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

- include/epiworld/agent-bones.hpp
- include/epiworld/tools-bones.hpp

### 3.14 Tools\_const< TSeq > Class Template Reference

Set of [Tools](#) (const) (useful for iterators)

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

#### Public Member Functions

- **Tools\_const** (const [Agent](#)< TSeq > &p)
- std::vector< TOOLPTR >::const\_iterator **begin** ()
- std::vector< TOOLPTR >::const\_iterator **end** ()
- const TOOLPTR & **operator**() (size\_t i)
- const TOOLPTR & **operator**[] (size\_t i)
- size\_t **size** () const noexcept

## Friends

- class **Tool**< TSeq >
- class **Agent**< TSeq >

### 3.14.1 Detailed Description

```
template<typename TSeq>
class Tools_const< TSeq >
```

Set of [Tools](#) (const) (useful for iterators)

#### Template Parameters

<i>TSeq</i>	
-------------	--

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

- include/epiworld/agent-bones.hpp
- include/epiworld/tools-bones.hpp

## 3.15 UserData< TSeq > Class Template Reference

Personalized data by the user.

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

## Public Member Functions

- **UserData** ([Model](#)< TSeq > &m)
- **UserData** (std::vector< std::string > names)
 

*Construct a new User Data object.*
- std::vector< std::string > & **get\_names** ()
- std::vector< int > & **get\_dates** ()
- std::vector< epiworld\_double > & **get\_data** ()
- void **get\_all** (std::vector< std::string > \*names=nullptr, std::vector< int > \*date=nullptr, std::vector< epiworld\_double > \*data=nullptr)
- unsigned int **nrow** () const
- unsigned int **ncol** () const
- void **write** (std::string fn)
- void **print** () const

### Append data

#### Parameters

x	A vector of length <code>ncol()</code> (if vector), otherwise a <code>epiworld_double</code> .
j	Index of the data point, from 0 to <code>ncol()</code> - 1.

- void **add** (std::vector< epiworld\_double > x)
- void **add** (unsigned int j, epiworld\_double x)

### Access data

#### Parameters

i	Row (0 through <i>ndays</i> - 1.)
j	Column (0 through <i>ncols</i> ()).

#### Returns

*epiworld\_double*&

- *epiworld\_double* & **operator()** (unsigned int i, unsigned int j)
- *epiworld\_double* & **operator()** (unsigned int i, std::string name)

### Friends

- class **Model**< TSeq >
- class **DataBase**< TSeq >

## 3.15.1 Detailed Description

```
template<typename TSeq>
class UserData< TSeq >
```

Personalized data by the user.

#### Template Parameters

<i>TSeq</i>	
-------------	--

## 3.15.2 Constructor & Destructor Documentation

### 3.15.2.1 UserData()

```
template<typename TSeq >
UserData< TSeq >::UserData (
    std::vector< std::string > names ) [inline]
```

Construct a new User Data object.

#### Parameters

<i>names</i>	A vector of names. The length of the vector sets the number of columns to record.
--------------	---

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

- include/epiworld/database-bones.hpp
- include/epiworld/userdata-bones.hpp
- include/epiworld/userdata-meat.hpp

## 3.16 vecHasher< T > Struct Template Reference

Vector hasher.

```
#include <misc.hpp>
```

### Public Member Functions

- `std::size_t operator() (std::vector< T > const &dat) const` noexcept

#### 3.16.1 Detailed Description

```
template<typename T>
struct vecHasher< T >
```

Vector hasher.

Template Parameters

<i>T</i>	
----------	--

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

- include/epiworld/misc.hpp

## 3.17 Virus< TSeq > Class Template Reference

[Virus](#).

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

### Public Member Functions

- **Virus** (std::string name="unknown virus")
- void **mutate** ()
- void **set\_mutation** (MutFun< TSeq > fun)
- const TSeq \* **get\_sequence** ()
- void **set\_sequence** (TSeq sequence)

- [Agent](#)< TSeq > \* **get\_agent** ()
- void **set\_agent** ([Agent](#)< TSeq > \*p, epiworld\_fast\_uint idx)
- [Model](#)< TSeq > \* **get\_model** ()
- void **set\_date** (int d)
- int **get\_date** () const
- void **set\_id** (int idx)
- int **get\_id** () const
- void **set\_name** (std::string name)
- std::string **get\_name** () const
- std::vector< epiworld\_double > & **get\_data** ()

### Get and set the tool functions

#### Parameters

v	<i>The virus over which to operate</i>
fun	<i>the function to be used</i>

#### Returns

*epiworld\_double*

- epiworld\_double **get\_prob\_infecting** ()
- epiworld\_double **get\_prob\_recovery** ()
- epiworld\_double **get\_prob\_death** ()
- void **post\_recovery** ()
- void **set\_post\_recovery** (PostRecoveryFun< TSeq > fun)
- void **set\_post\_immunity** (epiworld\_double prob)
- void **set\_post\_immunity** (epiworld\_double \*prob)
- void **set\_prob\_infecting\_fun** (VirusFun< TSeq > fun)
- void **set\_prob\_recovery\_fun** (VirusFun< TSeq > fun)
- void **set\_prob\_death\_fun** (VirusFun< TSeq > fun)
- void **set\_prob\_infecting** (epiworld\_double \*prob)
- void **set\_prob\_recovery** (epiworld\_double \*prob)
- void **set\_prob\_death** (epiworld\_double \*prob)
- void **set\_prob\_infecting** (epiworld\_double prob)
- void **set\_prob\_recovery** (epiworld\_double prob)
- void **set\_prob\_death** (epiworld\_double prob)

### Get and set the status and queue

After applied, viruses can change the status and affect the queue of agents. These function sets the default values, which are retrieved when adding or removing a virus does not specify a change in status or in queue.

#### Parameters

init	<i>After the virus/tool is added to the agent.</i>
end	<i>After the virus/tool is removed.</i>
removed	<i>After the agent (<a href="#">Agent</a>) is removed.</i>

- void **set\_status** (epiworld\_fast\_int init, epiworld\_fast\_int end, epiworld\_fast\_int removed=-99)
- void **set\_queue** (epiworld\_fast\_int init, epiworld\_fast\_int end, epiworld\_fast\_int removed=-99)
- void **get\_status** (epiworld\_fast\_int \*init, epiworld\_fast\_int \*end, epiworld\_fast\_int \*removed=-99)
- void **get\_queue** (epiworld\_fast\_int \*init, epiworld\_fast\_int \*end, epiworld\_fast\_int \*removed=-99)



## Friends

- class **Agent**< TSeq >
- class **Model**< TSeq >
- class **DataBase**< TSeq >
- void **default\_add\_virus** ([Action](#)< TSeq > &a, [Model](#)< TSeq > \*m)
- void **default\_add\_tool** ([Action](#)< TSeq > &a, [Model](#)< TSeq > \*m)
- void **default\_rm\_virus** ([Action](#)< TSeq > &a, [Model](#)< TSeq > \*m)
- void **default\_rm\_tool** ([Action](#)< TSeq > &a, [Model](#)< TSeq > \*m)

### 3.17.1 Detailed Description

```
template<typename TSeq = int>
class Virus< TSeq >
```

[Virus](#).

Template Parameters

<i>TSeq</i>	
-------------	--

Raw transmissibility of a virus should be a function of its genetic sequence. Nonetheless, transmissibility can be reduced as a result of having one or more tools to fight the virus. Because of this, transmissibility should be a function of the agent.

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

- include/epiworld/agent-bones.hpp
- include/epiworld/virus-bones.hpp
- include/epiworld/virus-meat.hpp

## 3.18 Viruses< TSeq > Class Template Reference

Set of viruses (useful for building iterators)

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

### Public Member Functions

- **Viruses** ([Agent](#)< TSeq > &p)
- std::vector< VIRUSPTR >::iterator **begin** ()
- std::vector< VIRUSPTR >::iterator **end** ()
- VIRUSPTR & **operator**() (size\_t i)
- VIRUSPTR & **operator**[] (size\_t i)
- size\_t **size** () const noexcept

## Friends

- class **Virus**< TSeq >
- class **Agent**< TSeq >

### 3.18.1 Detailed Description

```
template<typename TSeq>
class Viruses< TSeq >
```

Set of viruses (useful for building iterators)

#### Template Parameters

<i>TSeq</i>	
-------------	--

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

- include/epiworld/agent-bones.hpp
- include/epiworld/viruses-bones.hpp

## 3.19 Viruses\_const< TSeq > Class Template Reference

Set of [Viruses](#) (const) (useful for iterators)

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

## Public Member Functions

- **Viruses\_const** (const [Agent](#)< TSeq > &p)
- std::vector< VIRUSPTR >::const\_iterator **begin** ()
- std::vector< VIRUSPTR >::const\_iterator **end** ()
- const VIRUSPTR & **operator()** (size\_t i)
- const VIRUSPTR & **operator[]** (size\_t i)
- size\_t **size** () const noexcept

## Friends

- class **Virus**< TSeq >
- class **Agent**< TSeq >

### 3.19.1 Detailed Description

```
template<typename TSeq>
class Viruses_const< TSeq >
```

Set of [Viruses](#) (const) (useful for iterators)

## Template Parameters

<i>TSeq</i>	
-------------	--

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

- include/epiworld/agent-bones.hpp
- include/epiworld/viruses-bones.hpp



# Index

## Action

Action< TSeq >, [6](#)

Action< TSeq >, [5](#)

Action, [6](#)

add\_global\_action

Model< TSeq >, [19](#)

AdjList, [7](#)

AdjList, [7](#)

Agent< TSeq >, [8](#)

DataBase< TSeq >, [10](#)

record\_variant, [11](#)

LFMCMC< TData >, [11](#)

Location< TSeq >, [13](#)

Model< TSeq >, [13](#)

add\_global\_action, [19](#)

reset, [19](#)

run\_multiple, [20](#)

write\_data, [20](#)

PersonTools< TSeq >, [21](#)

Progress, [21](#)

Queue< TSeq >, [21](#)

RandGraph, [22](#)

record\_variant

DataBase< TSeq >, [11](#)

reset

Model< TSeq >, [19](#)

run\_multiple

Model< TSeq >, [20](#)

Tool< TSeq >, [22](#)

Tools< TSeq >, [24](#)

Tools\_const< TSeq >, [24](#)

## UserData

UserData< TSeq >, [26](#)

UserData< TSeq >, [25](#)

UserData, [26](#)

vecHasher< T >, [27](#)

Virus< TSeq >, [27](#)

Viruses< TSeq >, [29](#)

Viruses\_const< TSeq >, [30](#)

write\_data

Model< TSeq >, [20](#)