Basic SIR Model

Jamie Prentice

2026-02-03

Load libraries and source files

This pulls in all the necessary libraries and source files

```
suppressPackageStartupMessages({
    source("libraries.R")
    source("source_files.R")
})
```

Generating a params list

The first step is to generate a params list that contains the basic information, with some messages to note exactly what it's using. All elements in `make_parameters()` have default values, so these are just to see what you can add.

The params list might not fully do everything we want, but as it is just a basic R list, we can easily modify it afterwards.

```
params <- make_parameters(
    model_type = "SIR",
    setup = "single",
    use_traits = "none",
    group_layout = "striped",
    sim_new_data = "r"
)
```

This provides a short summary of the params

```
summarise_params(params)
```

```
## Simulating new SIR data via R

## - Demography is:
##     10 sires, 20 dams, 500 progeny (530 total), 1 groups (500 per group)
##     R0 = 5
## - FEs are
##     Trial = , Donor = , TxD = , Weight =
## - Running MCMC with:
##     10,000 updates / 10,000 samples / 0.2 burnin / 16 chains
## - Data will be saved to 'datasets/testing/data/scen-1-1.bici'
```

```
## Sigma_G =

##     sus inf tol
## sus   0   0   0
## inf   0   0   0
## tol   0   0   0
```

Generate pedigree and popn

Next we take the `params` file, set the groups and traits, and apply any fixed effects. Note that you these are piped into each other, but you can save the intermediate result at any point to examine what's happening (e.g. for bug fixing).

```
popn <- make_pedigree(params) |>
    set_groups(params) |>
    set_traits(params) |>
    set_weights(params) |>
    apply_fixed_effects(params)
```

It's worth taking a look at the `popn` file to see what it includes. The columns are:

```
names(popn)
```

```
##  [1] "id"     "sire"   "dam"    "sdp"    "trial"  "group"  "weight" "donor"
##  [9] "GE"     "sus_g"  "inf_g"  "tol_g"  "sus_e"  "inf_e"  "tol_e"  "sus"
## [17] "inf"    "tol"
```

Where:

- `id`: the unique ID of the individual. Sires come first, then dams, then progeny.
- `sire` and `dam` of the individual (together with `id` this makes a pedigree).
- `sdp`: whether the individual is a sire, dam, or progeny.
- The `weight` in kg at start of experiment.
- Which `trial` the individual was assigned to.
- Which `group` the individual was assigned to.
- `donor`: if the individual was inoculated (1) or not (0).
- `GE`: a small tank dependent group effect.
- `sus_g`, `inf_g`, `tol_g`, `sus_e`, `inf_e`, `tol_e`: genetic and environmental values. These are normally distributed with mean 0.
- `sus`, `inf`, `tol`, `lat`, `det`: phenotypic values. These are log-normally distributed and incorporate any fixed effects.

A typical progeny looks like:

```
popn[sdp == "progeny"][1]
```

```
## Key: <id>
##       id  sire   dam     sdp trial group weight donor    GE sus_g inf_g tol_g
##    <int> <int> <int>  <fctr> <int> <int>  <num> <int> <num> <num> <num> <num>
## 1:    31     1    11 progeny     1     1   40.8     1     1     0     0     0
##    sus_e inf_e tol_e   sus   inf   tol
##    <num> <num> <num> <num> <num> <num>
## 1:     0     0     0     1     1     1
```

Simulate and plot epidemic

We now pass the population file `popn` and the parameters `params` to `simulate_epidemic()`, which will run the appropriate model and return a *new* `popn` file with event times, final status, which generation infective they were, and the individual responsible for their infection. You can generate multiple realisations of the epidemic from the same inputs if you save the `popn` output each time.

```
tic(); popn <- simulate_epidemic(popn, params); toc()
```

```
## 3.785 sec elapsed
```

We can also use the generation value to calculate R_0, which is the mean across all groups of the ratio of secondary to primary infectives. Also it's useful to see how long the epidemic took until it completed (since this information is necessary for BICI to use).

```
params$estimated_R0 <- get_R0(popn)
```

```
## R0 estimate: 5.8
```

```
params$tmax <- get_tmax(popn, params)

message("Tmax = [", str_flatten_comma(round(params$tmax, 1)), "]")
```

```
## Tmax = [72.4]
```

The progeny we had before now looks like this, with the additional columns:

- `Tinf`: the infection time ($t : S \to E$). In actual data this is 0 for inoculated individuals, and missing for all others.
- `Tinc`: the incubation time ($t : E \to I$), missing in the data.
- `Tsym`: the time of symptoms ($t : I \to D$), present in data.
- `Tdeath`: the time of death ($t : D \to R$), present in data.
- `status`: which compartment the individual is in at the end of the epidemic (should only be `S` or `R`)
- `generation`: if an individual is a primary, secondary, tertiary etc. infective.
- `infected_by`: the id of the individual responsible for this infection (0 for inoculated individuals).
- `parasites` `TRUE` if an individual was infected. In actual data the sensitivity is not 100%.

```
popn[sdp == "progeny"][1]
```

```
##       id  sire   dam     sdp trial group weight donor    GE sus_g inf_g tol_g
##    <int> <int> <int>  <fctr> <int> <int>  <num> <int> <num> <num> <num> <num>
## 1:    31     1    11 progeny     1     1   40.8     1     1     0     0     0
##    sus_e inf_e tol_e   sus   inf   tol  Tinf   Tdeath status generation
##    <num> <num> <num> <num> <num> <num> <num>    <num> <fctr>      <int>
## 1:     0     0     0     1     1     1     0 6.392557      R          1
##    infected_by parasites
##          <int>    <lgcl>
## 1:           0      TRUE
```

Plot the epidemic

We can take a look at the time trajectory

```
plt <- plot_model(popn, params)
```

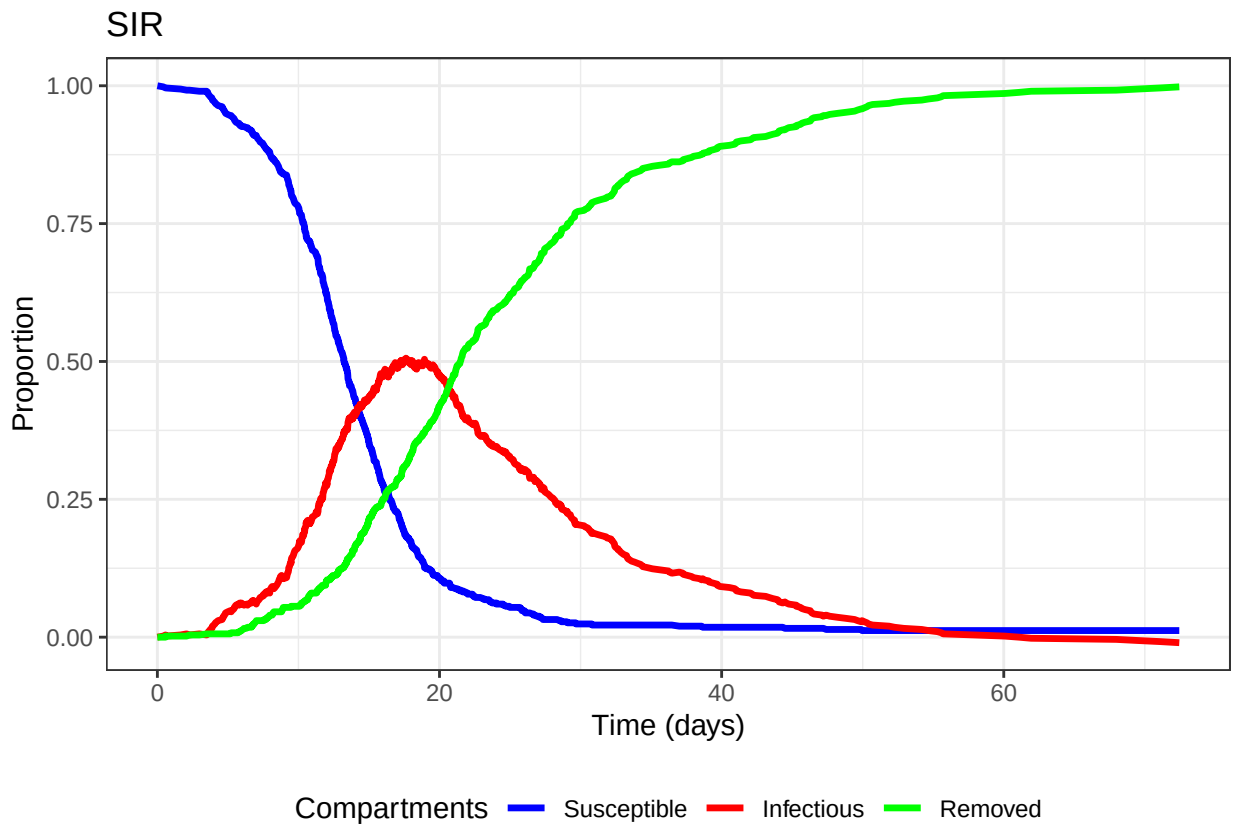

Figure 1: Time trajectory of SEIDR model

It's also important to look at the Kaplan-Meier survival plot.

```
plt2 <- basic_km(popn, params)
```

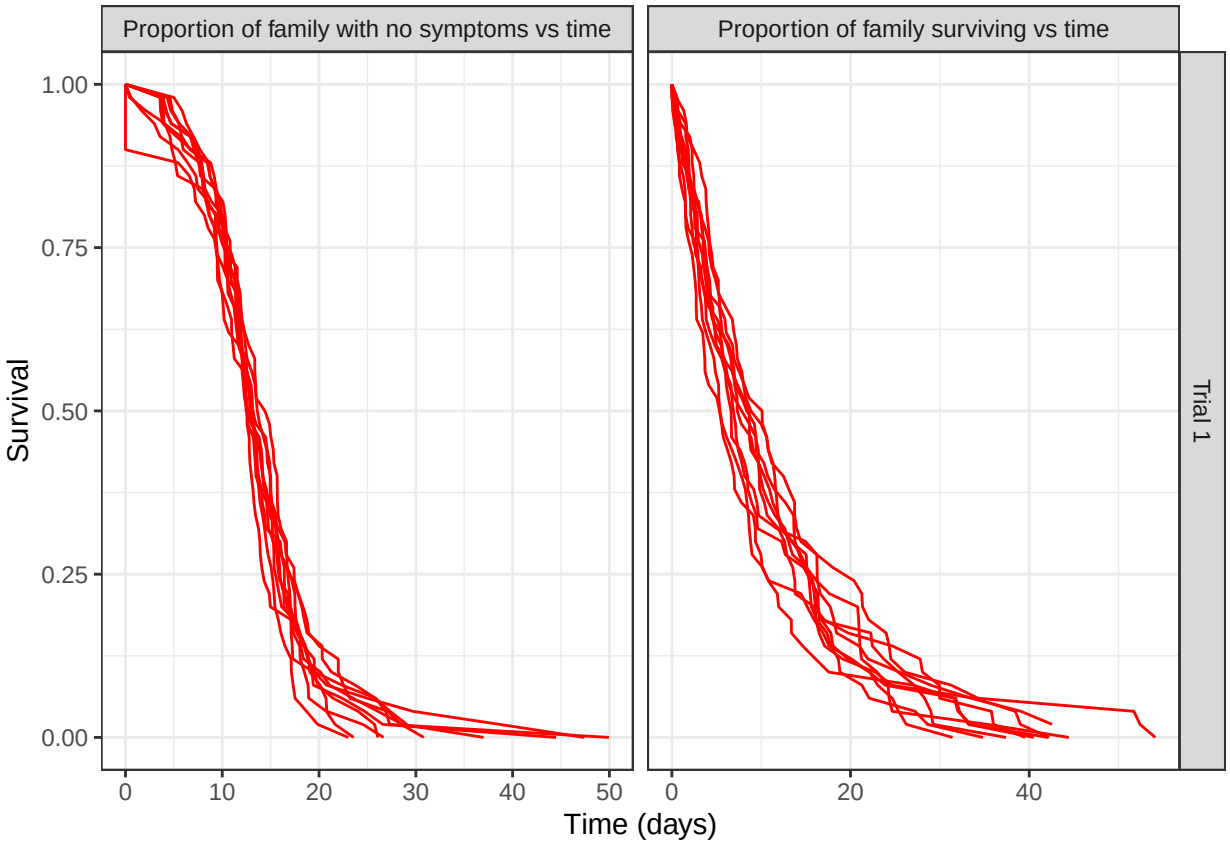

Figure 2: KM plot of SEIDR model, showing the proportion of each family surviving over time.