Using ABC to Infer Parameters of a Simulated Zombie Epidemic

Henry Bourne, Emma Tarmey, Rachel Wood

June 2023

The REVISIT package

Need to create a package with appropriate folders and documentation - am happy to start on this

Documentation

Integration of R and C++

Simulations

Parallelisation

Generating SZR data

We simulate the zombie epidemic using the `generate.SIR.data()` function, which is given by the code below. For ease, this function has also been included in our `intractmodelinf` package.

We use this to simulate an outbreak for 3 different populations of different sizes:

- Students (**REVISIT** and staff?) at the University of Bristol
- Residents in the city of Bristol
- Residents in the UK

The data on these simulated outbreaks is included in our package

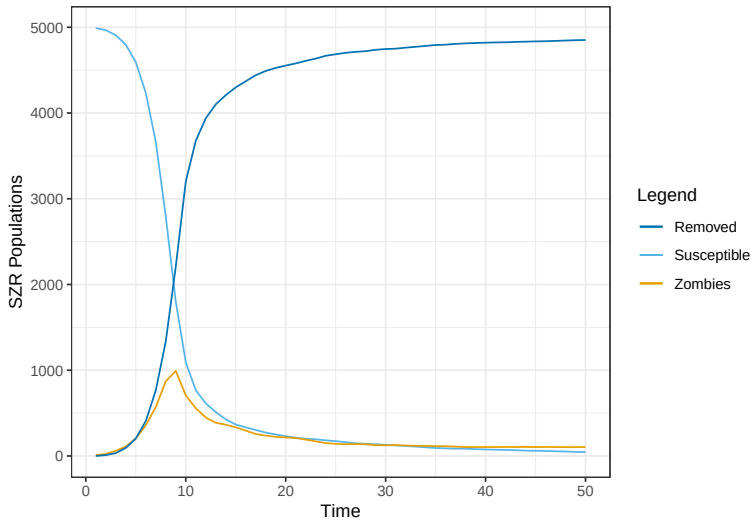
```
N     <-5000
initial <- generateStartCond(N, initial.inf = 10)

new.example <- generateSZRdata(initial ,total.T     = 50)
```

Implementing ABC

Checking Results

```
#bristol.uni.example <- read.csv("data/bristol_uni_example.csv", header = TRUE)
#bristol.example     <- read.csv("data/bristol_example.csv", header = TRUE)
#UK.example          <- read.csv("data/UK_example.csv", header = TRUE)

legend.colors <- c("Susceptible" = "#56B4E9", "Zombies" = "#E69F00", "Removed" =
↪  "#0072B2")
source(file = "intractmodelinf/R/plots.R")
```

```
plot.SZR(new.example$results) + theme_linedraw() +theme_bw()
```




```
simulated.mat <- new.example$results
simulated.mat  <- as.matrix(simulated.mat[,2:4])
head(simulated.mat )
```

```
##       S.t Z.t R.t
## [1,] 4990  10   0
## [2,] 4966  24  10
## [3,] 4906  60  34
## [4,] 4796 110  94
## [5,] 4592 204 204
## [6,] 4234 358 408
```

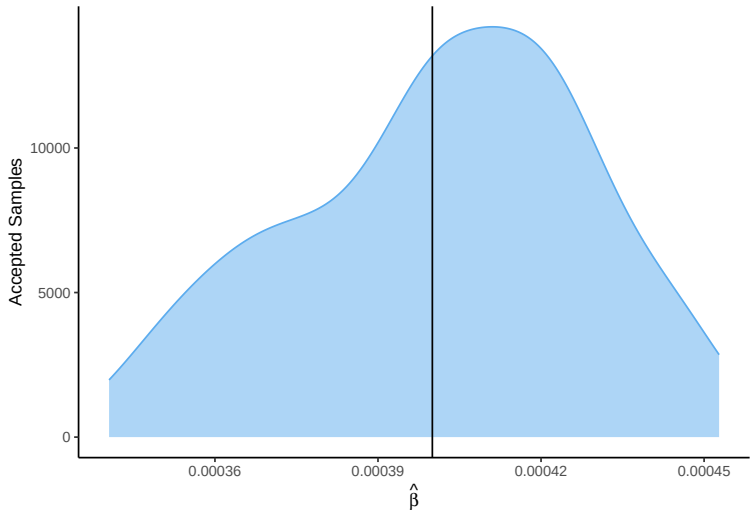
```
priorMin <- c(0.000004,0.000004,  0.000004)
priorMax <- c(0.0015 ,0.001, 0.001)
res <- abcRej(simulated.mat , 10000, 400, priorMin, priorMax)
head(res)
```

```
##              [,1]         [,2]         [,3]
## [1,] 0.0004250723 0.0008617067 0.0006589291
## [2,] 0.0004237616 0.0009770686 0.0007613837
## [3,] 0.0003875616 0.0004843493 0.0006826764
## [4,] 0.0003708313 0.0006252518 0.0007569413
## [5,] 0.0004010205 0.0007269969 0.0005887166
## [6,] 0.0004492728 0.0009563075 0.0002842800
```

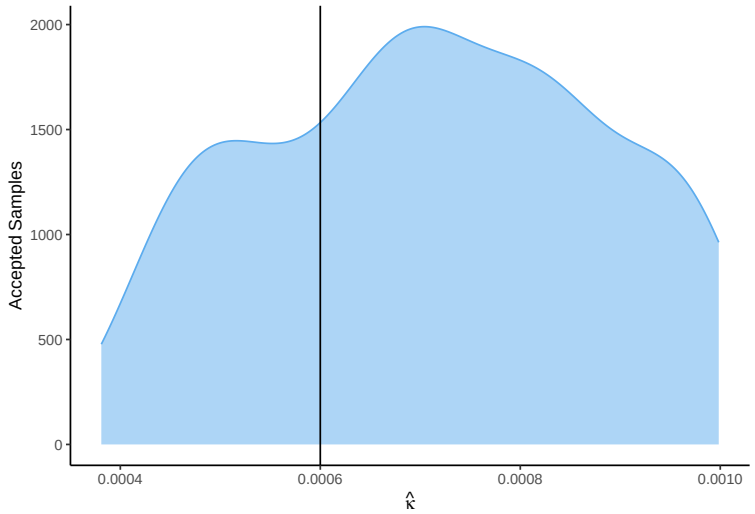
```
library(latex2exp)
require(ggplot2)
require(gridExtra)
plot.posterior.samples <- function(sample, mean){
  beta_plot <- ggplot(data = sample, aes(x = beta)) +
    geom_density(colour = "steelblue2", fill = "steelblue2", alpha = 0.5) +
    geom_vline(aes(xintercept = mean[1]))+
    labs(x = TeX("$\\hat{\\beta}$"), y = "Accepted Samples") +
    theme_classic()
  kappa_plot <- ggplot(data = sample, aes(x = kappa)) +
```

```
    geom_density(fill = "steelblue2", alpha = 0.5,colour = "steelblue2") +
    geom_vline(aes(xintercept = mean[2])) +
    labs(x = TeX("$\\hat{\\kappa}$"), y = "Accepted Samples") +
    theme_classic()
  rho_plot <- ggplot(data = sample, aes(x = rho)) +
    geom_density(colour = "steelblue2", fill = "steelblue2", alpha = 0.5) +
    geom_vline(aes(xintercept = mean[3]))+
    labs(x = TeX("$\\hat{\\rho}$"), y = "Accepted Samples") +
    theme_classic()
  return(list(beta_plot, kappa_plot, rho_plot))
}
```

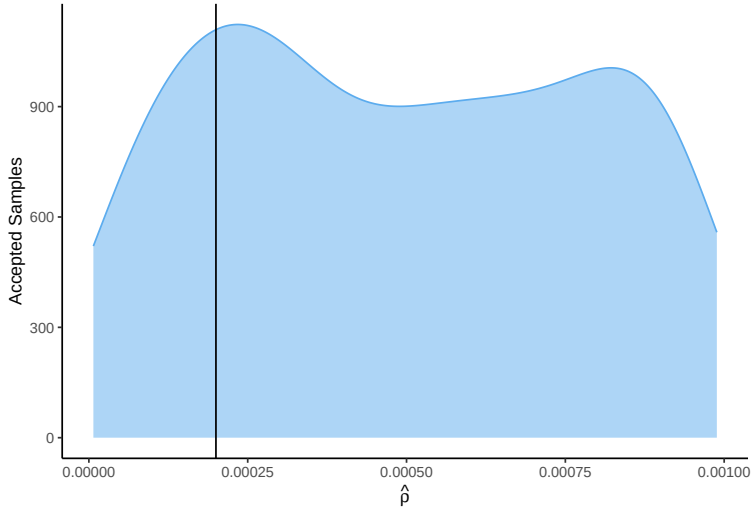
```
library(ggplot2)
library(dplyr)
library(ggthemes)
means <- c(new.example$beta, new.example$kappa, new.example$rho)
samples <- res %>%
  as_tibble()
colnames(samples) <- c("beta", "kappa", "rho")

plots <- plot.posterior.samples(samples, means)
plots[[1]]
```




```
plots[[2]]
```


2000
1500
1000
500
0
Accepted Samples
0.0004
0.0006
0.0008
0.0010
κ̂

```
plots[[3]]
```


900
600
300
0
Accepted Samples
0.00000
0.00025
0.00050
0.00075
0.00100
ρ̂