Estimation scheme for fitting the stochastic model for the transmission of 2019-nCov in Hubei

Andrew Tredennick & John Drake

March 28, 2020

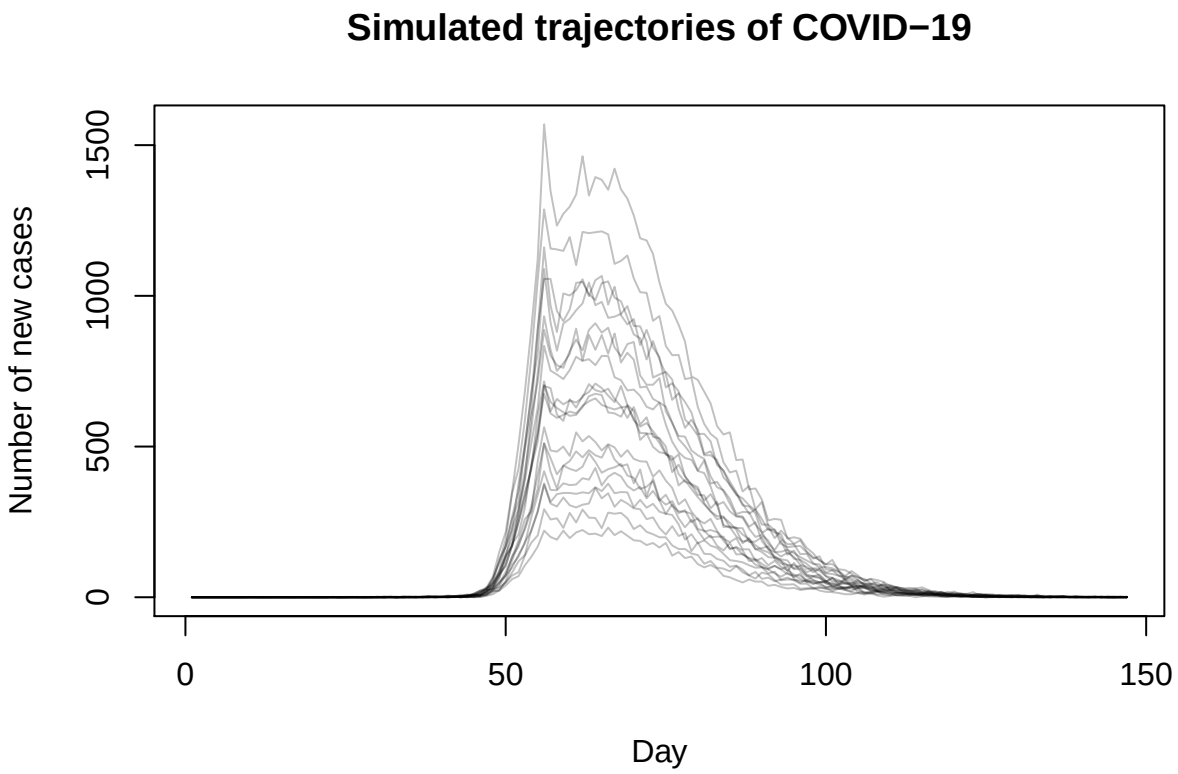
Introduction

This document describes a proposed estimation scheme for fitting the stochastic model (described elsewhere) of COVID-19 transmission to data. We plan to use Approximate Bayesian Computation to estimate unknown parameters. The posterior distribution of parameters will be explored using MCMC. While the end goal is estimate parameters given real data, we start by using the model itself to simulate a trajectory of disease transmission and tuning the model fitting process to ensure we can recover known parameters.

Simulated data

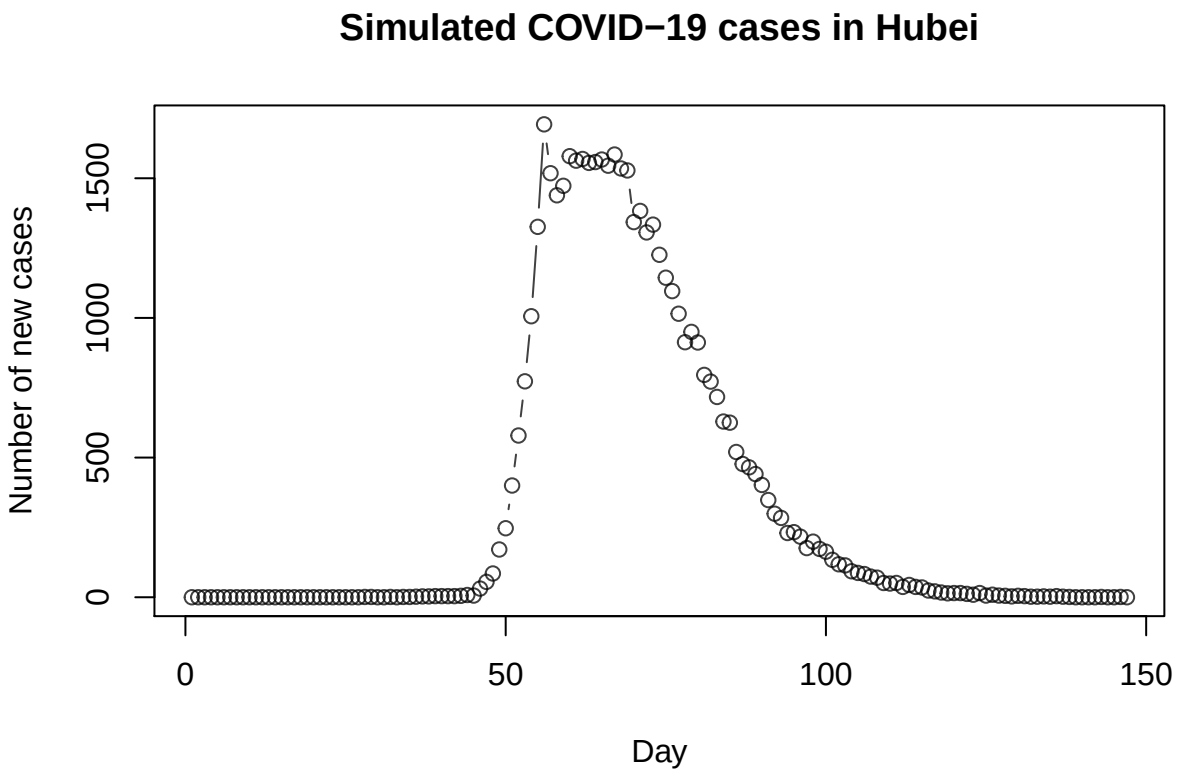
Here we simulate data from the stochastic model using the `simulators.R` functions. The `simulators.R` functions are wrappers around the core model functions written by Drake and Rohani.

```
# First source the necessary functions
source("stochastic-model.R")  # the stochastic disease transmission model
source("simulators.R")  # simulator wrappers
source("style-definitions.R")  # colors

# Time spans for simulation
start <- as.Date("2019-12-01")
today <- Sys.Date()

# Unknown parameters
theta <- list(beta0 = 0.657, beta.factor = 1, sigma = 1/6.4, I0 = 1)

# Known parameters
extraArgs <- list(nstep = NULL, start = start,
                  today = today, dt = 0.05, w = 40, z = 45, c = 1,
                  presymptomatic = 0, timesToObs = FALSE,
                  b = 0.143, a0 = 0.0446, paramNames = names(theta),
                  nObs = NULL, obsDataDate = NULL)

# Initial conditions
init <- list(S=59002000, E1=0, E2=0, E3=0, E4=0, E5=0, E6=0,
             I1 = NA, I2= 0, I3=0, I4=0, Iu1=0, Iu2=0, Iu3=0, Iu4=0,
             H=0, Ru=0, C=0)
extraArgs$init <- init

# Simulate from the model
res <- mod_wrap(param = log(unlist(theta)), nsim = 20, extraArgs = extraArgs)

matplot(t(res), type = "l", col = col.cases.ci, lty = 1,
        main = "Simulated trajectories of COVID-19",
        xlab = "Day", ylab = "Number of new cases")
```



We can save one simulated trajectory as "data" for testing of the estimation scheme.

```
obs_data <- mod_wrap(param = log(unlist(theta)), nsim = 1, extraArgs = extraArgs)
extraArgs$obsData <- obs_data
extraArgs$nObs <- length(obs_data)
plot(obs_data, type='b',  xlab='Day', col = col.cases,
     ylab='Number of new cases', main='Simulated COVID-19 cases in Hubei')
```

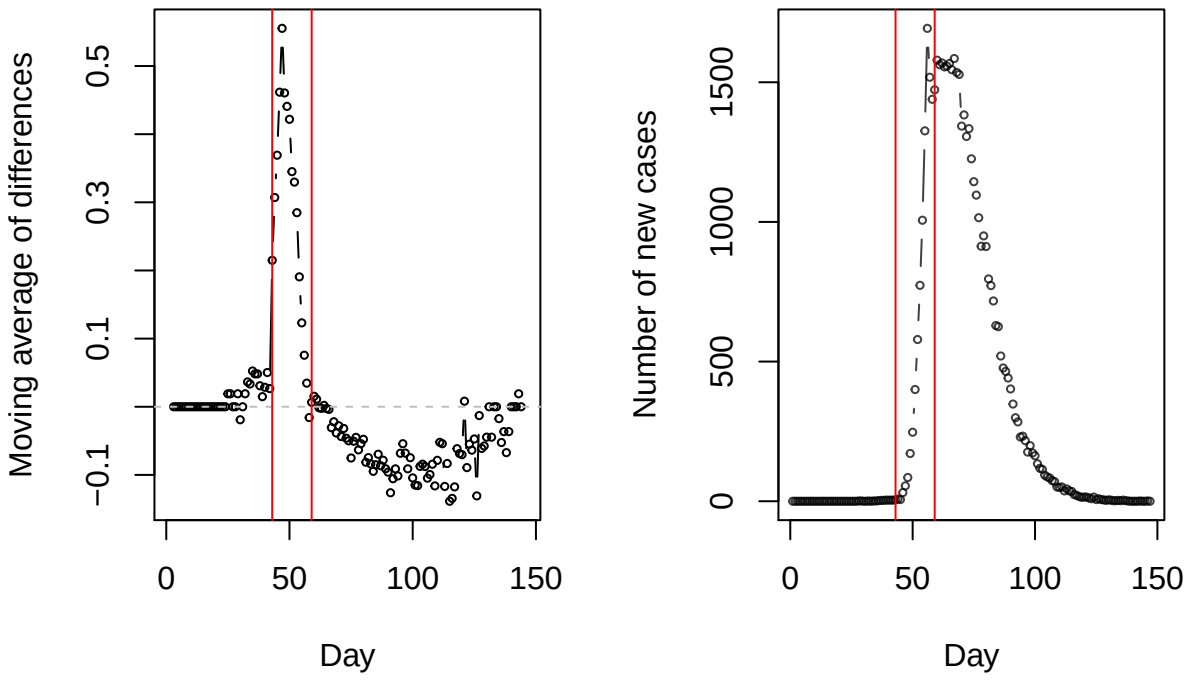


Summary statistics for the ABC

Fitting models using ABC requires calculating a vector of summary statistics (S) that summarize the observed time series and simulated trajectories. The summary statistics (S_{sim}) from simulated time series are compared to the summary statistics of the observed time series (S_{obs}) using a distance function d. To begin with, I used a simple d function that calculates the mean absolute diference across all N summary statistics: $\sum_{i=1}^{N} |S_{i,\text{obs}} - S_{i,\text{sim}}|/N$.

For our particular problem, I used the following summary statistics based on phases of the epidemic that are defined by case numbers. Specifically, we used a five day moving average of the differences of log observed new cases (plus 10) to define the following phases: day at which the the moving average of log differences exceeds 0.1 (d_0) and the day at which the moving average of log differences drops below -0.01 (d_1). The plot below shows the time series of the five-day moving average of the differences of new cases, along with the calculated break points defined above

```
ma <- function(x, n = 5){stats::filter(x, rep(1 / n, n), sides = 2)}
stat_times <- ma(diff(log(obs_data+10)), n = 5)
d0 <- min(which(stat_times > 0.1))
d1 <- d0 + min(which(stat_times[d0:length(stat_times)] < -0.01))
par(mfrow = c(1,2))
plot(stat_times, xlab = "Day", ylab = "Moving average of differences",
     type = "b", cex = 0.5)
abline(h = 0, col = "grey", lty = 2)
abline(v = c(d0, d1), col = "red")
plot(obs_data, type='b',  xlab='Day', col = col.cases,
     ylab='Number of new cases', cex = 0.5)
abline(v = c(d0, d1), col = "red")
```



The summary statistics are as follows:

1. Maximum incidence before day d_0 (stuttering chain of tranmission)
2. Maximum incidence between day d_0 and day d_1 (exponential phase)
3. Maximum incidence after day d_1 (decling phase)
4. Cumulative incidence before day d_0
5. Cumulative incidence between day d_0 and day d_1
6. Cumulative incidence after day d_1
7. Maximum incidence along the whole trajectory
8. Day of epidemic peak (timing of max incidence)
9. Final size of epidemic (total incidence)

Now we code up calculations for these summary statistics for `synlik`.

```
covid_stats <- function(x, extraArgs, ...) {
  ## obsData is a vector of observed path
  ## x is a M by n.t matrix of paths, each row of 'x' is a replicate

  ma <- function(x, n = 5){stats::filter(x, rep(1 / n, n), sides = 2)}
  stat_times <- ma(diff(log(extraArgs$obsData+10)), n = 5)
  d0 <- min(which(stat_times > 0.1))
  d1 <- d0 + min(which(stat_times[d0:length(stat_times)] < -0.01))

  obsData <- extraArgs$obsData

  stopifnot(is.vector(obsData), length(obsData) != 0)
  if (!is.matrix(x)) x <- matrix(x, 1, length(x))
```

```
  x <- log(x + 10)  # log transform

  # Max incidence
  X0 <- (apply(x, 1, function(x) max(x[1:(d0-1)])))
  X0 <- cbind(X0, (apply(x, 1, function(x) max(x[d0:d1]))))
  X0 <- cbind(X0, (apply(x, 1, function(x) max(x[(d1+1):length(x)]))))

  # Cumulative incidence
  X0 <- cbind(X0, (apply(x, 1, function(x) sum(x[1:(d0-1)]))))
  X0 <- cbind(X0, (apply(x, 1, function(x) sum(x[d0:d1]))))
  X0 <- cbind(X0, (apply(x, 1, function(x) sum(x[(d1+1):length(x)]))))

  # Max along whole trajectory
  X0 <- cbind(X0, apply(x, 1, max))

  # Day of max
  X0 <- cbind(X0, apply(x, 1, function(x) which.max(x)))

  # Exponential increase and decline
  X0 <- cbind(X0, apply(x, 1, max) / (apply(x, 1, function(x) which.max(x)) - d0))

  # Linear regression coefs
  X0 <- cbind(X0, apply(x, 1, function(x) as.numeric(coef(lm(x[1:(d0-1)] ~ seq_along(x[1:(d0-1)])))[2]))
  X0 <- cbind(X0, apply(x, 1, function(x) as.numeric(coef(lm(x[d0:d1] ~ seq_along(x[d0:d1])))[2])))
  X0 <- cbind(X0, apply(x, 1, function(x) as.numeric(coef(lm(x[(d1+1):length(x)] ~ seq_along(x[(d1+1):le

  # Final epidemic size
  X0 <- cbind(X0, apply(x, 1, sum))
  return(X0)
}
```

Here are example statistics from a few simulations.

```
summaries <- covid_stats(x = res, extraArgs = extraArgs)
print(summaries)
```

```
##             X0
##  [1,] 2.564949 7.002156 6.960348 98.17811 87.04742 395.5828 7.002156 56
##  [2,] 2.772589 7.364547 7.295056 98.62321 92.14244 410.3452 7.364547 56
##  [3,] 2.639057 6.971669 6.964136 98.33789 87.72826 393.4738 6.971669 56
##  [4,] 2.564949 6.566672 6.577861 97.53450 82.19288 369.4539 6.577861 64
##  [5,] 2.564949 6.799056 6.823286 98.08280 85.07317 386.8299 6.823286 64
##  [6,] 2.484907 6.848005 6.803505 98.46967 85.51893 381.6392 6.848005 56
##  [7,] 2.484907 6.587550 6.575076 97.36745 82.73417 368.8336 6.587550 56
##  [8,] 2.484907 6.255750 6.190315 97.54977 76.95845 348.7695 6.255750 56
##  [9,] 2.397895 5.948035 5.921578 97.18512 73.18544 334.2068 5.948035 56
## [10,] 2.484907 5.710427 5.707110 97.82740 70.39128 325.1517 5.710427 56
## [11,] 2.484907 6.352629 6.322565 97.46276 78.94450 356.9220 6.352629 56
## [12,] 2.484907 6.253829 6.242223 97.46276 78.28662 346.6683 6.253829 56
## [13,] 2.484907 6.570883 6.529419 97.08152 82.26776 368.1793 6.570883 56
## [14,] 2.484907 6.059123 6.084499 97.35915 74.81475 351.6034 6.084499 64
## [15,] 2.564949 6.738152 6.698268 98.17811 83.09824 383.4016 6.738152 56
## [16,] 2.484907 5.442418 5.484797 97.27214 67.65934 309.2514 5.484797 66
## [17,] 2.639057 7.065613 6.981006 98.31566 88.27509 394.3341 7.065613 56
```

```
## [18,] 2.484907 6.529419 6.507278 97.74039 81.60760 366.5757 6.529419 56
## [19,] 2.639057 7.167809 7.109879 97.97458 89.90331 404.4255 7.167809 56
## [20,] 2.397895 5.961005 5.958425 97.08981 73.99956 338.2358 5.961005 56
##
##  [1,] 0.5386274 0.0034714974 0.3150393 -0.06541992 580.8083
##  [2,] 0.5665036 0.0050686363 0.3265908 -0.06973101 601.1108
##  [3,] 0.5362822 0.0041726915 0.3198208 -0.06649070 579.5400
##  [4,] 0.3132315 0.0021942219 0.2889448 -0.06051558 549.1813
##  [5,] 0.3249184 0.0027895255 0.3061248 -0.06346806 569.9859
##  [6,] 0.5267696 0.0037616147 0.3054462 -0.06343206 565.6278
##  [7,] 0.5067346 0.0015442434 0.2991502 -0.06108401 548.9352
##  [8,] 0.4812115 0.0021667539 0.2765895 -0.05493875 523.2777
##  [9,] 0.4575412 0.0012434113 0.2604007 -0.04896663 504.5773
## [10,] 0.4392636 0.0026411815 0.2489646 -0.04627693 493.3703
## [11,] 0.4886638 0.0015633779 0.2807026 -0.05633656 533.3293
## [12,] 0.4810638 0.0019340843 0.2702569 -0.05595747 522.4177
## [13,] 0.5054525 0.0009254008 0.2946818 -0.05926474 547.5286
## [14,] 0.2897381 0.0015603212 0.2750537 -0.05220592 523.7773
## [15,] 0.5183194 0.0030774909 0.3168150 -0.06109578 564.6779
## [16,] 0.2384694 0.0015788240 0.2382254 -0.04426751 474.1829
## [17,] 0.5435087 0.0037037330 0.3117819 -0.06558401 580.9249
## [18,] 0.5022630 0.0020929061 0.3009729 -0.06019695 545.9237
## [19,] 0.5513699 0.0030082957 0.3316918 -0.06732836 592.3034
## [20,] 0.4585389 0.0009576584 0.2613926 -0.05161737 509.3252
```

And the statistics from the "data".

```
obs_summaries <- covid_stats(x = obs_data, extraArgs = extraArgs)
print(obs_summaries)
```

```
##            X0
## [1,] 2.639057 7.440147 7.374629 99.23806 94.09151 420.9693 7.440147 56
##
## [1,] 0.572319 0.006713293 0.3343521 -0.07195647 614.2988
```

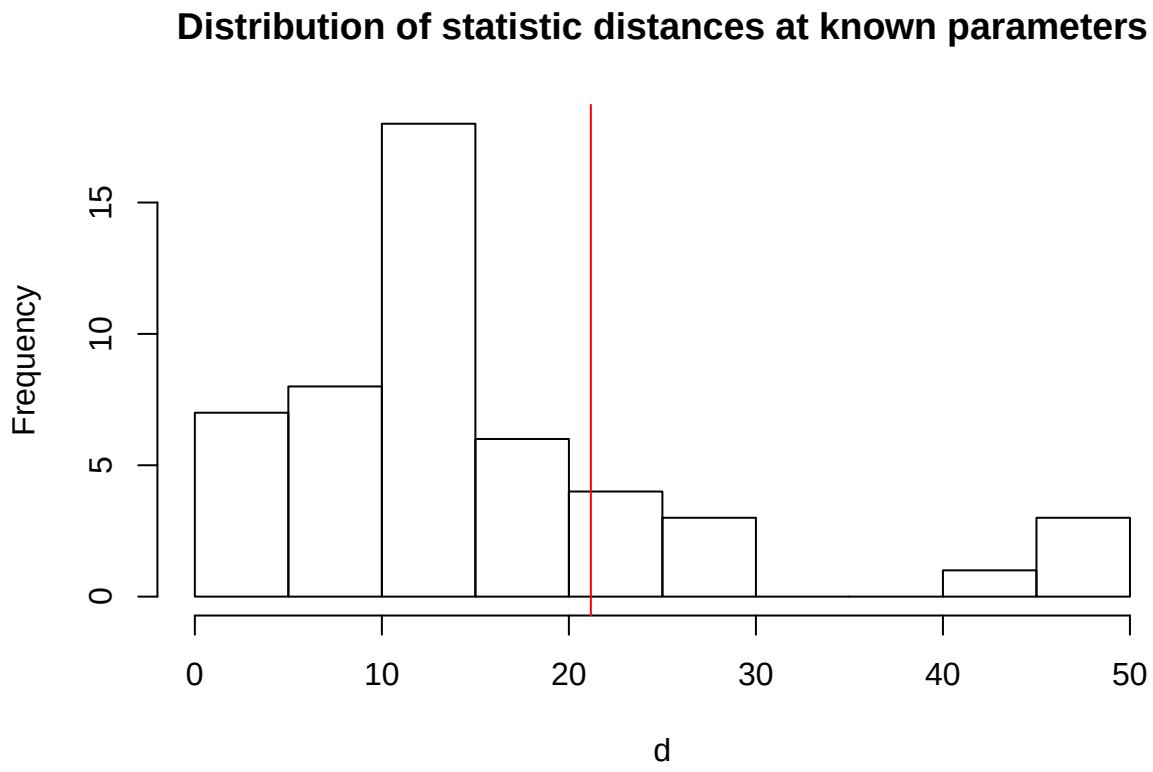
Last, we define a distance function d. It is important to know how d might vary due to model stochasticity alone because we have to set a threshold value (ε) that d must fall below for acceptance of a parameter vector in the ABC-MCMC algorithm. So, I simulated 200 trajectories from the same parameter values and calculated the distances of the summary statistics. The resulting histogram indicates an initial $\varepsilon \approx 20$.

```
d <- function(xsim, xobs) {
  mean(abs(xobs - xsim))
}

d(summaries[2,], obs_summaries)
```

```
## [1] 2.05829
```

```
dsims <- 50
dist <- numeric(dsims)
for(i in 1:dsims) {
  res <- mod_wrap(param = log(unlist(theta)), nsim = 1, extraArgs = extraArgs)
  summaries <- covid_stats(x = res, extraArgs = extraArgs)
  dist[i] <- d(summaries, obs_summaries)
}
hist(dist, main = "Distribution of statistic distances at known parameters",
     xlab = "d")
```

```r
abline(v = quantile(dist, 0.8), col = "red")
```




```r
epsilon <- quantile(dist, 0.8)
```

ABC Algorithm

Here I use ABC-MCMC to estimate β_0 assuming all other parameters are known.

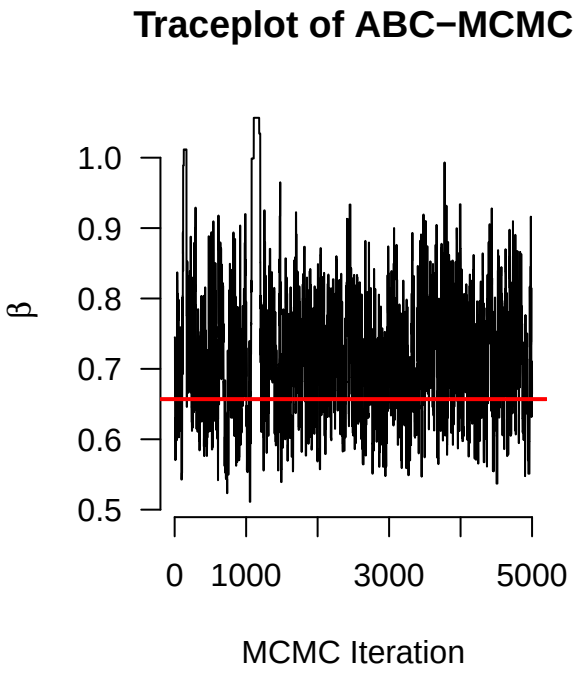
```r
get_prior_d <- function(theta, theta_prime) {
  d_theta <- dnorm(theta[1], log(0.657), 0.5) +
             dnorm(theta[2], log(1), 0.5) +
             dnorm(theta[3], log(1/6,4), 0.5) +
             dunif(theta[4], -5, 5)

  d_theta_prime <- dnorm(theta_prime[1], log(0.657), 0.5) +
                   dnorm(theta_prime[2], log(1), 0.5) +
                   dnorm(theta_prime[3], log(1/6,4), 0.5) +
                   dunif(theta_prime[4], -5, 5)
  return(min(1, exp(d_theta_prime - d_theta)))
}


run_abc <- function(nburn, nsamp, epsilon, f_stats, f_distance, f_model,
                    init_theta, obs_data, ...) {
  results <- matrix(nrow = nsamp, ncol = length(init_theta))
  i <- 0
```

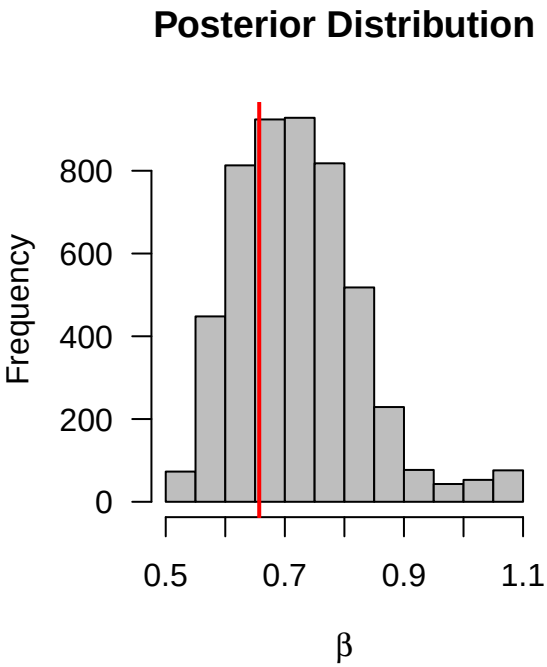
```
  theta <- init_theta
  while(i < (nburn+nsamp)) {
    theta_prime <- rnorm(length(theta), theta, sd = c(0.1, 0, 0, 0))
    p <- f_prior(as.numeric(theta), theta_prime)
    if(rbinom(1, 1, p) == 1) {
      m <- f_model(param = theta_prime, nsim = 1, extraArgs = extraArgs)
      mS <- f_stats(x = m, extraArgs = extraArgs)
      m0 <- f_stats(x = obs_data, extraArgs = extraArgs)
      dtest <- f_distance(mS, m0)
      if(dtest < epsilon) {
        theta <- theta_prime
      }
    }
#     propsal = c
#     accept with p exp(prior(propsal) - prior(current))
#     if accept do ABC step
    i <- i + 1
    if(i > nburn) results[(i - nburn), ] <- theta
  }
  return(results)
}
```

```
init_theta <- log(c(beta0 = 0.5, beta.factor = 1, sigma = 1/6.4, I0 = 1))
out_mcmc <- run_abc(nburn = 2000, nsamp = 5000, epsilon = 10, f_stats = covid_stats,
                    f_distance = d, f_model = mod_wrap, f_prior = get_prior_d,
                    init_theta = init_theta, obs_data = obs_data)
```

We are able to retrieve the true parameter estimate, as $\beta_0 = 0.657$ falls well within the posterior distribution estimated by the model fitting algorithm. Note that the initial value of $\beta_0 = 0.5$ falls outside the posterior distribution, indicating the algorithm does move away from unlikely values effectively.

```
# Plot the chain and posterior distribution of beta0
out_mcmc <- readRDS("mcmc-chains-abc.RDS")
par(mfrow = c(1,2))
plot(exp(out_mcmc[,1]), type = "l", xlab = "MCMC Iteration",
          ylab = expression(beta), las = 1, bty = "n", main = "Traceplot of ABC-MCMC")
abline(h = as.numeric(theta[1]), col = "red", lwd = 2)
hist(exp(out_mcmc[,1]), breaks = 15, col = "grey", xlab = expression(beta),
     las = 1, main = "Posterior Distribution")
abline(v = as.numeric(theta[1]), col = "red", lwd = 2)
```


Traceplot of ABC−MCMC
β
1.0
0.9
0.8
0.7
0.6
0.5
0
1000
3000
5000
MCMC Iteration


Posterior Distribution
Frequency
800
600
400
200
0
0.5
0.7
0.9
1.1
β