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

*Andrew Tredennick & John Drake*

*March 26, 2020*

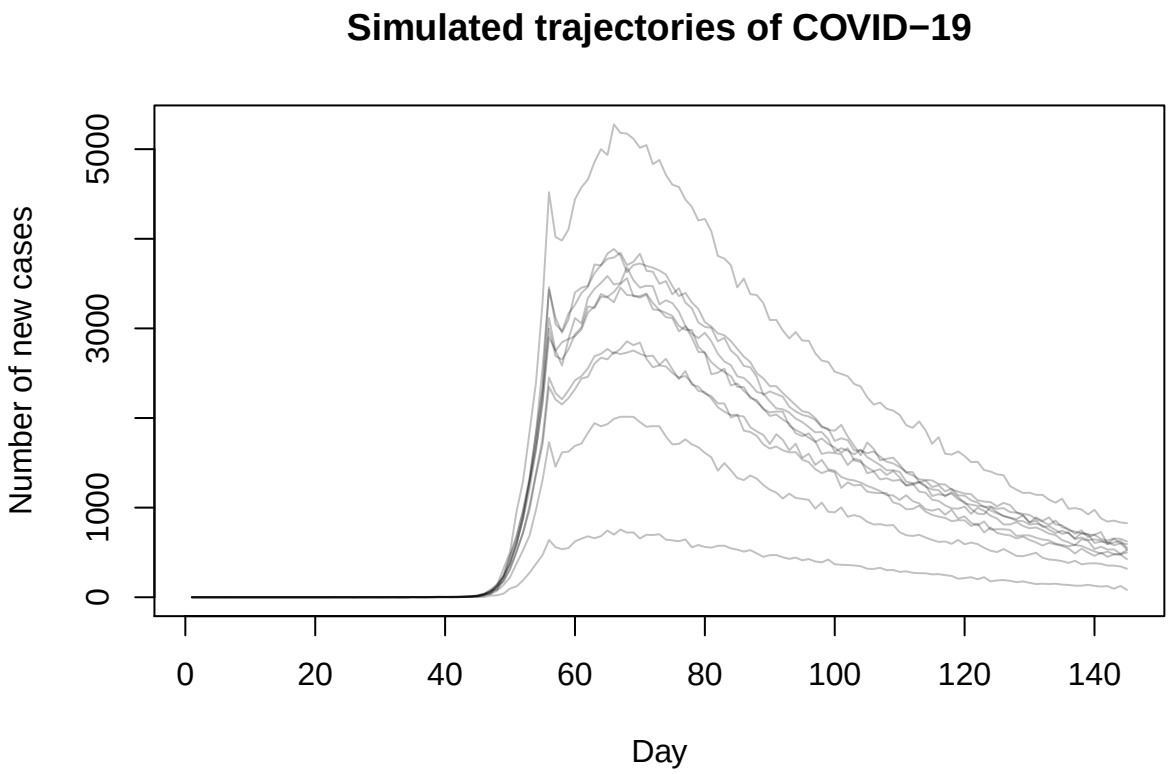
## Introduction

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

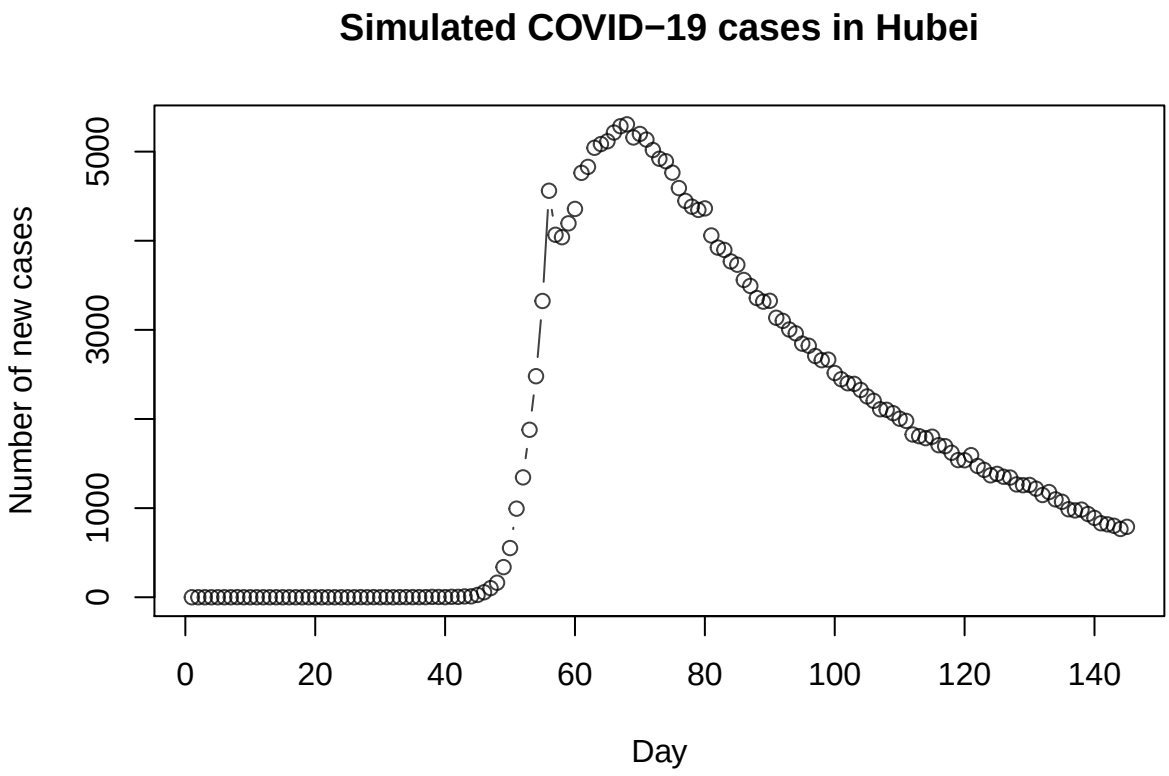
Here we simulate data from the stochastic model using the `simulators.R` functions. The `simulators.R` functions are wrappers around the core model functions written by Drake and Rohani.

```
# 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
set.seed(18276)
res <- mod_wrap(param = log(unlist(theta)), nsim = 10, 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')
```

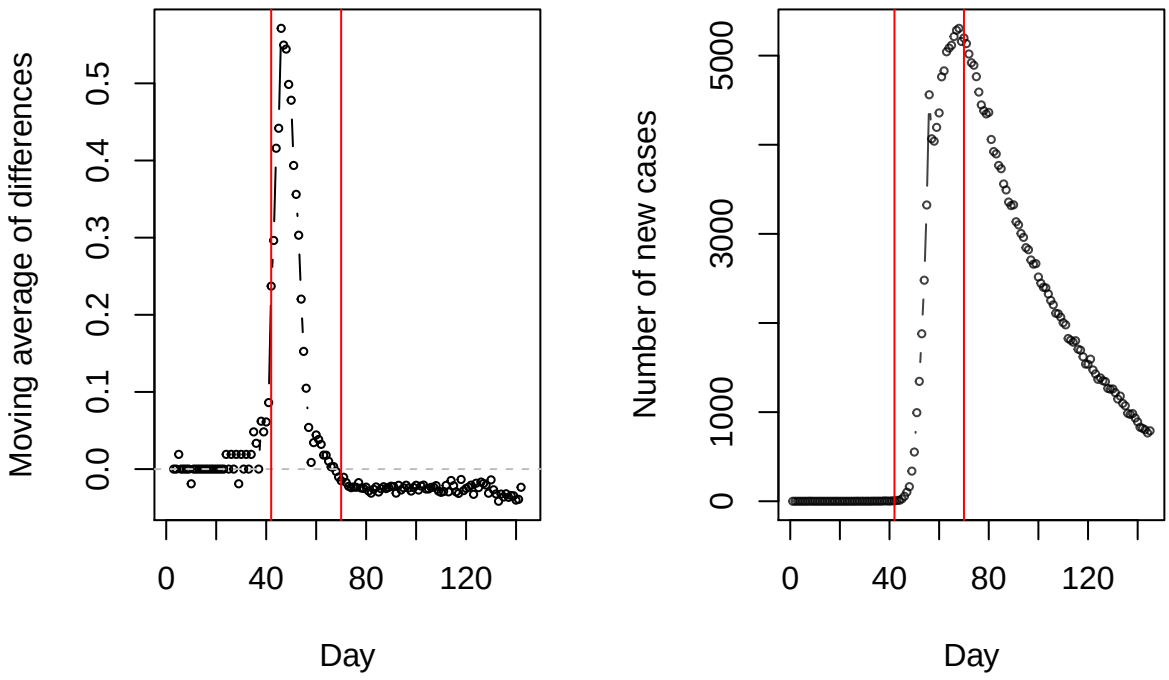



### Summary statistics for the ABC

Fitting models using ABC requires calculating a vector of summary statistics ($S$) that summarize the observed time series and simulated trajectories. The summary statistics ($S_{\text{sim}}$) from simulated time series are compared to the summary statistics of the observed time series ($S_{\text{obs}}$) 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,] 2.484907 7.924072 7.903227 95.52481 186.1482 529.6638 7.924072 69
##  [2,] 2.833213 8.256867 8.203304 96.19196 194.4073 549.3925 8.256867 67
##  [3,] 2.708050 8.573006 8.528133 96.07377 201.8044 572.7090 8.573006 66
##  [4,] 2.564949 8.180601 8.131237 95.57299 191.8816 543.2914 8.180601 68
##  [5,] 2.708050 8.213924 8.156510 95.55734 193.0161 545.0115 8.213924 68
##  [6,] 2.639057 8.267706 8.218248 96.19539 194.9021 550.9975 8.267706 66
##  [7,] 2.772589 7.960673 7.890208 95.86304 186.5693 526.2483 7.960673 68
##  [8,] 2.484907 6.641182 6.569481 94.87424 153.7008 428.6390 6.641182 67
##  [9,] 2.397895 7.612831 7.559559 95.07316 178.4311 501.6695 7.612831 67
## [10,] 2.639057 8.150756 8.127405 96.27780 191.7108 539.5819 8.150756 67
##
##  [1,] 0.2934842 0.002464616 0.2024469 -0.02396760 811.3368
##  [2,] 0.3302747 0.004890071 0.2095921 -0.02436040 839.9918
##  [3,] 0.3572086 0.004602215 0.2156467 -0.02462688 870.5872
##  [4,] 0.3146385 0.003507723 0.2061949 -0.02364319 830.7460
##  [5,] 0.3159201 0.003170736 0.2063447 -0.02448781 833.5849
##  [6,] 0.3444877 0.004944146 0.2065177 -0.02455098 842.0949
```

```
##  [7,] 0.3061797 0.004413960 0.2039459 -0.02470150 808.6806
##  [8,] 0.2656473 0.001059055 0.1673614 -0.02493819 677.2140
##  [9,] 0.3045132 0.001344969 0.1934893 -0.02390334 775.1738
## [10,] 0.3260303 0.004952310 0.2057519 -0.02518504 827.5705
```

And the statistics from the "data".

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

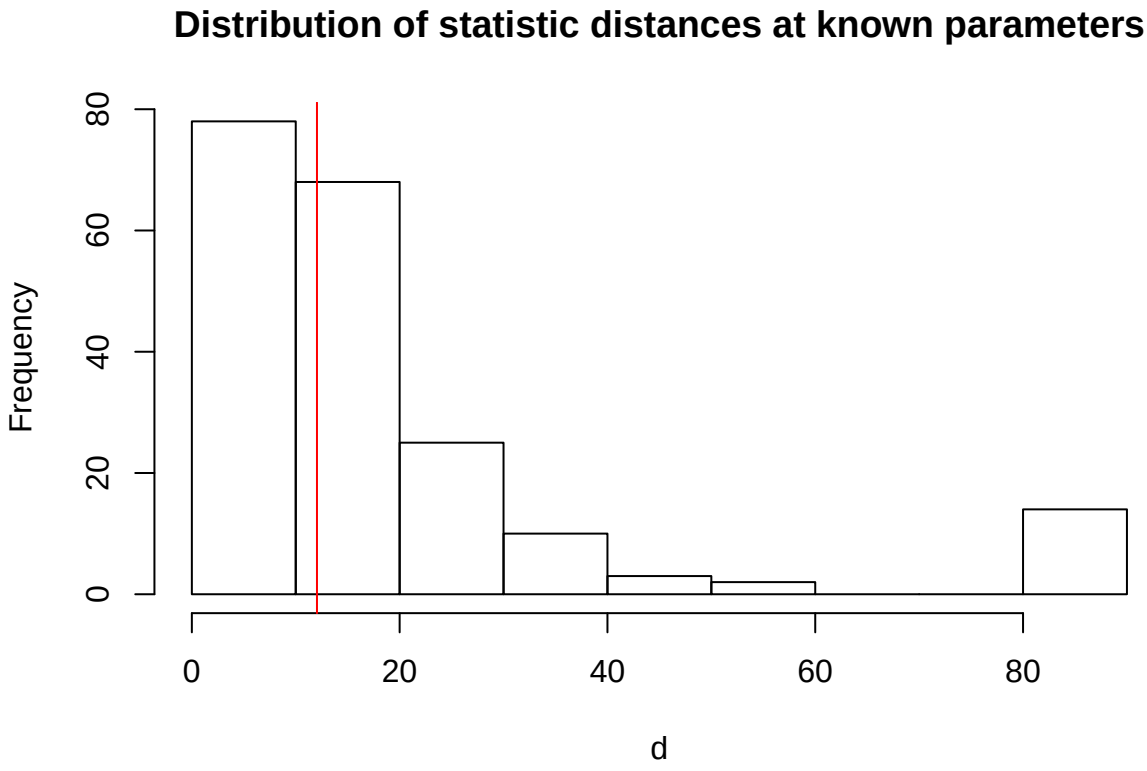
```
##           X0
## [1,] 2.70805 8.578476 8.545781 96.36339 203.1631 573.1521 8.578476 68
##
## [1,] 0.3299414 0.004892492 0.2120789 -0.02512898 872.6786
```

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 ($\varepsilon$) 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 $\varepsilon \approx 10$.

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

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

```
## [1] 5.191392
```

```
dsims <- 200
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")
abline(v = median(dist), col = "red")
```




## ABC Algorithm

```
run_abc <- function(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 < nsamp) {
    theta_prime <- rnorm(length(theta), theta, sd = c(0.05, 0.05, 0.05, 0.05))
#     propsal = c
#     accept with p exp(prior(propsal) - prior(current))
#     if accept do ABC step
    m <- f_model(param = theta_prime, nsim = 1, extraArgs = extraArgs)
    mS <- f_stats(x = m, extraArgs = extraArgs)
    mO <- f_stats(x = obs_data, extraArgs = extraArgs)
    dtest <- f_distance(mS, mO)
    if(dtest < epsilon) {
      theta <- theta_prime
    }
    i <- i + 1
    results[i, ] <- theta
  }
  return(results)
}
```

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

# Plot the chains
par(mfrow = c(2, 2))
plot(out_mcmc[,1], type = "l", xlab = "Iteration",
     ylab = expression(log(beta)), las = 1)
abline(h = log(as.numeric(theta[1])), col = "red")
plot(out_mcmc[,2], type = "l", xlab = "Iteration",
     ylab = expression(log(xi)), las = 1)
abline(h = log(as.numeric(theta[2])), col = "red")
plot(out_mcmc[,3], type = "l", xlab = "Iteration",
     ylab = expression(log(sigma)), las = 1)
abline(h = log(as.numeric(theta[3])), col = "red")
plot(out_mcmc[,4], type = "l", xlab = "Iteration",
     ylab = expression(log(I[0])), las = 1)
abline(h = log(as.numeric(theta[4])), col = "red")
```