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

Andrew Tredennick & John Drake

March 25, 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 the "synthetic likelihood" approach described by Wood (2010) to estimate unknown parameters. The posterior distribution of parameters will be explored using MCMC. The analysis presented here relies on the `synlik` R package functions. 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 `synlik` functions. The `synlik` functions are essentially wrappers around the core model functions written by Drake and Rohani. See the R script `simulators.R` for simulation wrappers that coerce the stochastic model into a form usable by `synlik`.

First, we define a `synlik` object using the pre-written wrappers.

```r
# First source the necessary functions and load the synlik package
library(synlik)
```

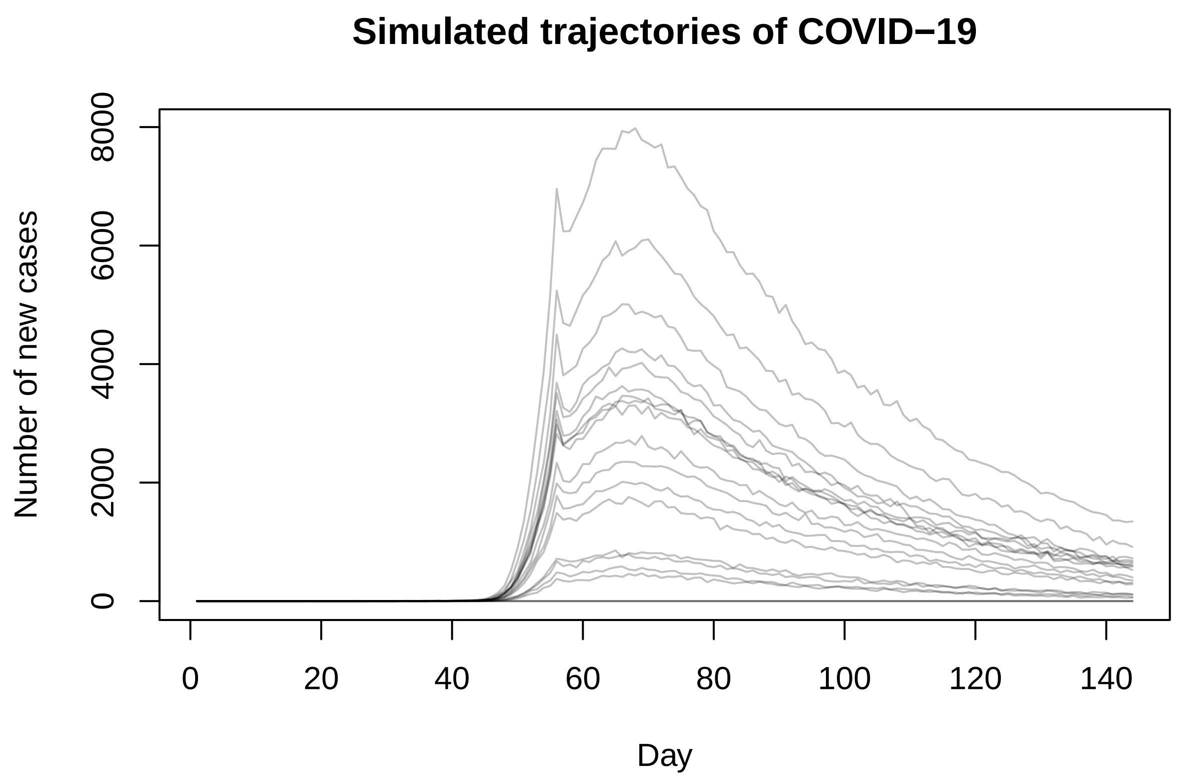
```
## This is "synlik"  0.1.2
```

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


# Create "synlik" object

# Parameters to estimate
params <- list(beta0 = 0.657, beta.factor = 1, sigma = 1/6.4)

# Initial conditions, specific to Hubei
init <- list(S=59002000, E1=0, E2=0, E3=0, E4=0, E5=0, E6=0,
              I1 = 1, I2= 0, I3=0, I4=0, Iu1=0, Iu2=0, Iu3=0, Iu4=0,
              H=0, Ru=0, C=0)

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

# Extra arguments needed for the stochastic model, specific to Hubei
extraArgs <- list("init" = init, "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(params), "nObs" = NULL)
```

```r
covid_sl <- new("synlik",
                simulator = mod_wrap,
                param = log(unlist(params)),
                extraArgs = extraArgs)
```

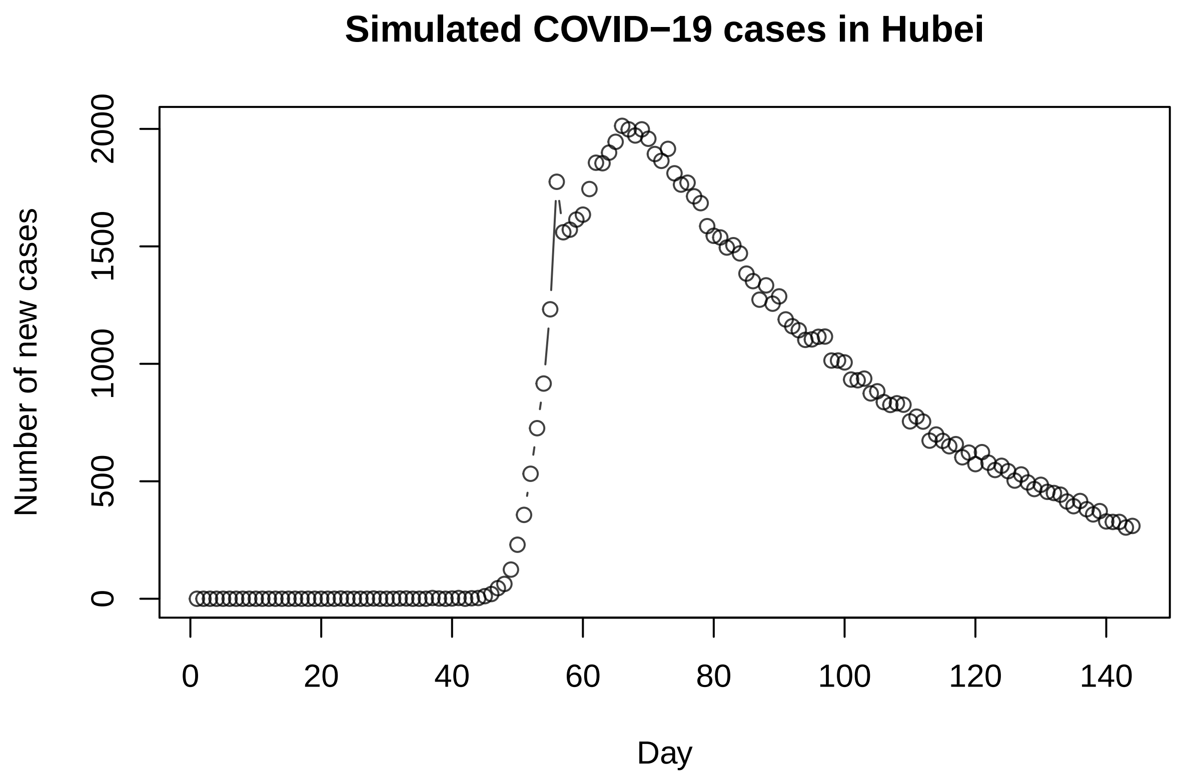
Then we simulate from the model to make sure it is working.

```r
res <- simulate(covid_sl, nsim = 20, seed = 1234)
matplot(t(res), type = "l", col = col.cases.ci, lty = 1,
        main = "Simulated trajectories of COVID-19",
        xlab = "Day", ylab = "Number of new cases")
```



Lastly, we can save one simulated trajectory as “data” for testing of the estimation scheme.

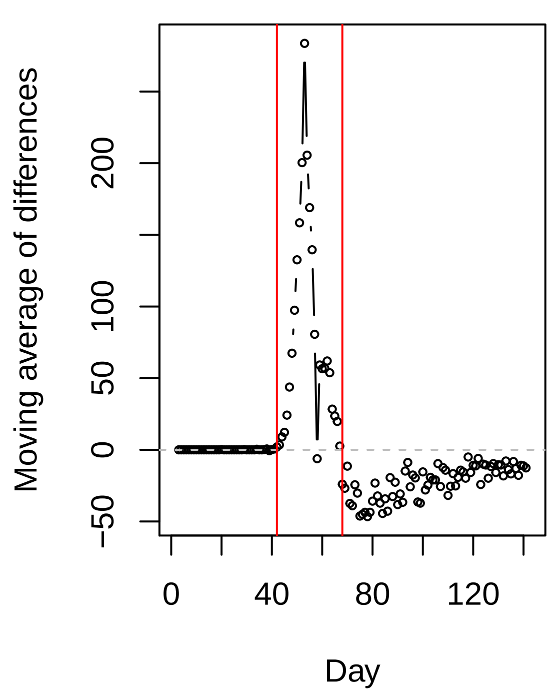
```r
covid_sl@data <- res[1, ]  # one replicate of the model simulations
covid_sl@extraArgs$obsData <- res[1, ]
covid_sl@extraArgs$nObs <- length(res[1, ])
plot(covid_sl@data, type='b',  xlab='Day', col = col.cases,
     ylab='Number of new cases', main='Simulated COVID-19 cases in Hubei')
```

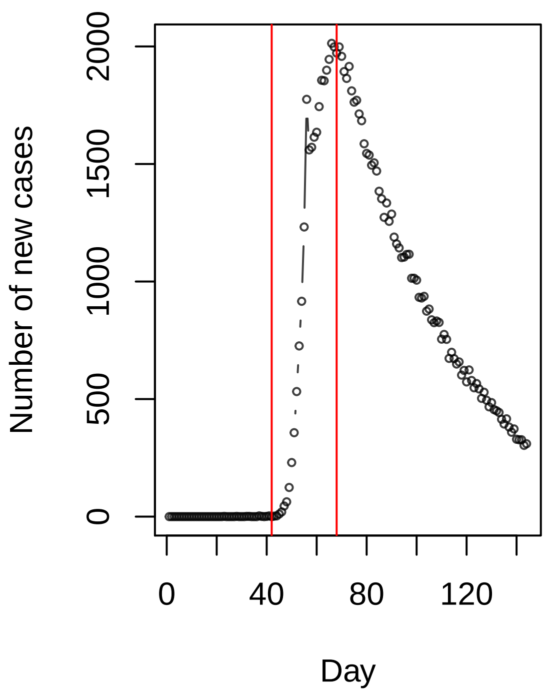


Summary statistics for the synthetic likelihood

Fitting models using the synthetic likelihood (Wood 2010) 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}) to evaluate the (synthetic) likelihood. 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 observed new cases to define the following phases: day at which the the moving average of differences exceeds 1 (d_0) and the day at which the moving average of differences drops below -10 (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

```r
ma <- function(x, n = 5){stats::filter(x, rep(1 / n, n), sides = 2)}
stat_times <- ma(diff(covid_sl@data), n = 5)
d0 <- min(which(stat_times > 1))
d1 <- min(which(stat_times < -10))
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(covid_sl@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`.

```r
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(extraArgs$obsData), n = 5)
  d0 <- min(which(stat_times > 1))
  d1 <- min(which(stat_times < -10))

  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):l

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

And add them to the `synlik` object.

```
covid_sl@summaries <- covid_stats
```

Here are example statistics from a few simulations.

```
test <- simulate(covid_sl, nsim = 5, stats = TRUE)
print(test)
```

```
##            X0
## [1,] 2.772589 8.776630 8.771835 96.87009 191.2623 598.6114 8.776630 67
## [2,] 2.397895 6.056784 5.988961 94.59661 126.9921 390.6936 6.056784 67
## [3,] 2.639057 6.904751 6.921658 94.92478 145.9837 455.1548 6.921658 69
## [4,] 2.397895 6.643790 6.610696 94.78723 140.3252 434.2356 6.643790 65
## [5,] 2.639057 7.896553 7.878534 95.48834 169.6227 530.4543 7.896553 67
##
## [1,] 0.3510652  7.159412e-03 0.2377974 -0.02524736 886.7438
## [2,] 0.2422714 -3.320912e-05 0.1669202 -0.02362363 612.2823
## [3,] 0.2563577  1.712354e-03 0.1968332 -0.02316932 696.0633
## [4,] 0.2888604  2.158593e-04 0.1855949 -0.02364982 669.3480
## [5,] 0.3158621  2.630778e-03 0.2211215 -0.02524936 795.5654
```

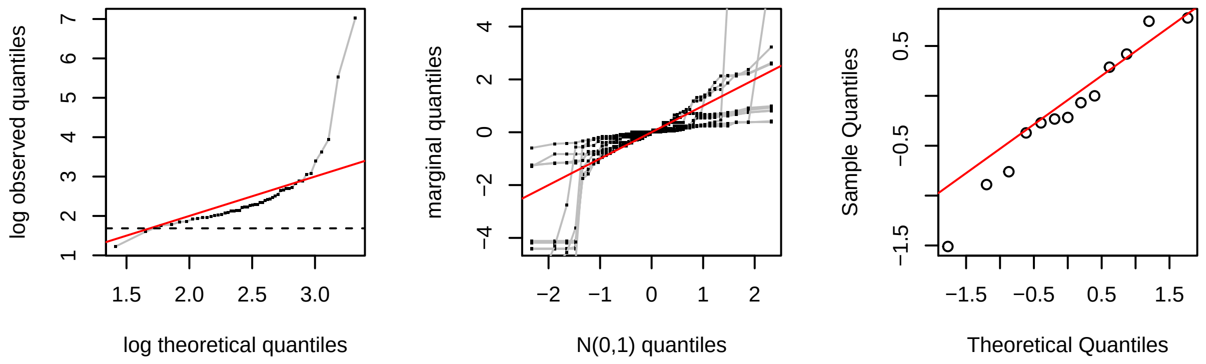
And the statistics from the “data”.

```r
covid_sl@summaries(x = covid_sl@data, extraArgs = covid_sl@extraArgs)
```

```
##            X0
## [1,] 2.564949 7.612337 7.604894 95.50258 162.7149 511.513 7.612337 66
##
## [1,] 0.3171807 0.0027746 0.2171529 -0.0243899 769.7305
```

We also want to make sure the summary statistics meet the assumption of normality.

```r
test_params <- c(beta0 = 0.657, beta.factor = 1, sigma = 1/6.4)
checkNorm(covid_sl, param = log(test_params), nsim = 50)
```



Model fitting

We are now ready to use MCMC to estimate unknown parameters. For the preliminary tests, we will try to fit two parameters: β_0 and ξ_β, which is the factor by which β_0 is reduced after policies designed to limit transmission were imposed on day x.

It is useful to make sure the likelihood can be evaluated before running MCMC.

```r
test_params <- c(beta0 = 0.67, beta.factor = 1, sigma = 1/6.4)
slik(covid_sl, param  = unname(log(test_params)), nsim = 40)
```

```
## [1] 26.59568
```

We want to work with informative priors, so we define those for `synlik`. The prior for β_0 is $N(-0.4, 1)$; the prior for ξ_β is $N(-1.6, 0.2)$. Note that parameters are specified on the log scale for estimation and transformed to the arithmetic scale for simulation.

```r
# This follows standard synlik notation
covid_priors_sl <- function(input, ...) { sum( input ) +
    dnorm (input[1], log(0.67), 1, log = TRUE) +  # beta0 prior
    dnorm (input[2], log(0.2), 0.2, log = TRUE) +
    dnorm (input[2], log(1/6.4), 0.25, log = TRUE)}  # beta reduction prior
```

Finally, we can run the `synlik::smcmc` routine.

```r
# Define perturbation covariance matrix for MCMC proposals
covMat <- diag(rep(0.1, length(params)))

# Set the initial parameters, arithmetic scale
init_params <- c(beta0 = 0.2, beta.factor = 1.5, sigma = 0.25)
```

```r
# Set up a cluster for parallel simulations within MCMC steps
num_cores <- parallel::detectCores() - 1
cl <- parallel::makeCluster(num_cores)
parallel::clusterExport(cl, ls())
parallel::clusterEvalQ(cl, {
  suppressMessages(suppressWarnings(library(synlik)))
})
```

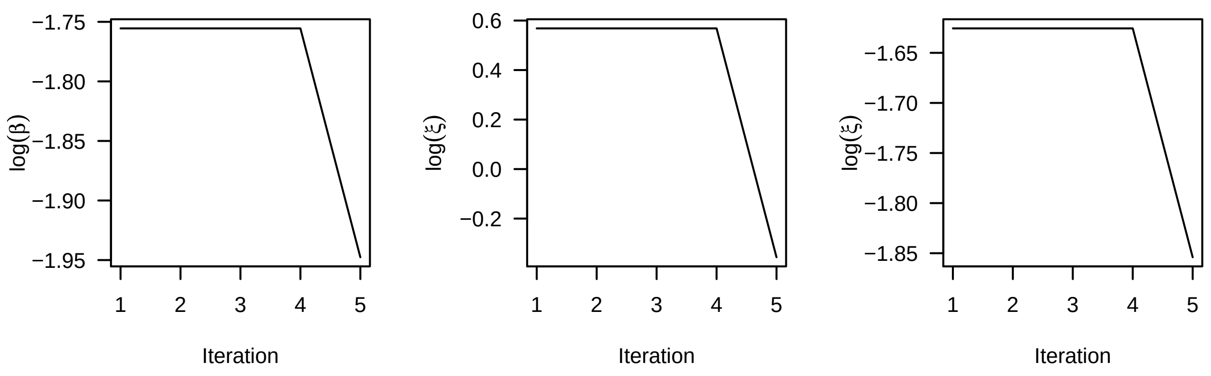
```
## [[1]]
## [1] "synlik"    "Rcpp"      "stats"     "graphics"  "grDevices" "utils"
## [7] "datasets"  "methods"   "base"
##
## [[2]]
## [1] "synlik"    "Rcpp"      "stats"     "graphics"  "grDevices" "utils"
## [7] "datasets"  "methods"   "base"
##
## [[3]]
## [1] "synlik"    "Rcpp"      "stats"     "graphics"  "grDevices" "utils"
## [7] "datasets"  "methods"   "base"
##
## [[4]]
## [1] "synlik"    "Rcpp"      "stats"     "graphics"  "grDevices" "utils"
## [7] "datasets"  "methods"   "base"
##
## [[5]]
## [1] "synlik"    "Rcpp"      "stats"     "graphics"  "grDevices" "utils"
## [7] "datasets"  "methods"   "base"
##
## [[6]]
## [1] "synlik"    "Rcpp"      "stats"     "graphics"  "grDevices" "utils"
## [7] "datasets"  "methods"   "base"
##
## [[7]]
## [1] "synlik"    "Rcpp"      "stats"     "graphics"  "grDevices" "utils"
## [7] "datasets"  "methods"   "base"
```

```r
# Run the MCMC
covid_mcmc <- smcmc(covid_sl,
                  initPar = log(init_params),
                  nsim = 30,
                  niter = 5,
                  burn = 0,
                  priorFun = covid_priors_sl,
                  propCov = covMat,
                  multicore = TRUE,
                  ncores = num_cores,
                  cluster = cl)
```

```
## Error in qr.default(t(sY1)) : NA/NaN/Inf in foreign function call (arg 1)
## Error in qr.default(t(sY1)) : NA/NaN/Inf in foreign function call (arg 1)
```

```r
# Stop the cluster
parallel::stopCluster(cl)

# Plot the chains
```

```r
par(mfrow = c(1, 3))
plot(covid_mcmc@chains[,1], type = "l", xlab = "Iteration",
     ylab = expression(log(beta)), las = 1)
plot(covid_mcmc@chains[,2], type = "l", xlab = "Iteration",
     ylab = expression(log(xi)), las = 1)
plot(covid_mcmc@chains[,3], type = "l", xlab = "Iteration",
     ylab = expression(log(xi)), las = 1)
```


log(β)
Iteration
log(ξ)
Iteration
log(ξ)
Iteration