Fitting stochastic COVID model using particle MCMC

Andrew Tredennick

3/28/2020

Using the lightly edited `pomp` model generated from `./code/make-pomp-object.R`, I used the `pomp::pmcmc` function to estimate parameters. This attempts to fit the model to the Georgia data.

First, read in the `pomp` object and define the prior density. The priors are based on the initial values chosen by eye and the variance around the means was chosen by looking at the resulting distributions. Note that `beta_d`, `beta_u`, and `beta_e` are all exponentiated within the process model itself, meaning that estimation occurs on the log scale. I did this because `pomp::pmcmc` *does not* adhere to the parameter transformations imposed on the pomp object (https://github.com/kingaa/pomp/issues/22#issuecomment-227276797).

```
# Load the pomp object ------------------------------------------------------

covid_ga_pomp <- readRDS("../output/covid-ga-pomp-object.RDS")


# Define the prior density --------------------------------------------------

prior_dens <- Csnippet(
  "
  lik = dnorm(beta_d,log(2e-7), 0.4, 1) +
    dnorm(beta_u, log(5e-8), 0.2, 1) +
    dnorm(beta_e, log(5e-8), 0.2, 1) +
    dunif(beta_red_factor, 0.01, 1, 1) +
    dlnorm(gamma_u, log(0.5), 0.2, 1) +
    dlnorm(gamma_u, log(0.5), 0.2, 1) +
    dunif(detect_frac_0, 0.01, 0.6, 1);
  if (!give_log) lik = exp(lik);
"
)
```

Now we can run the particle MCMC.

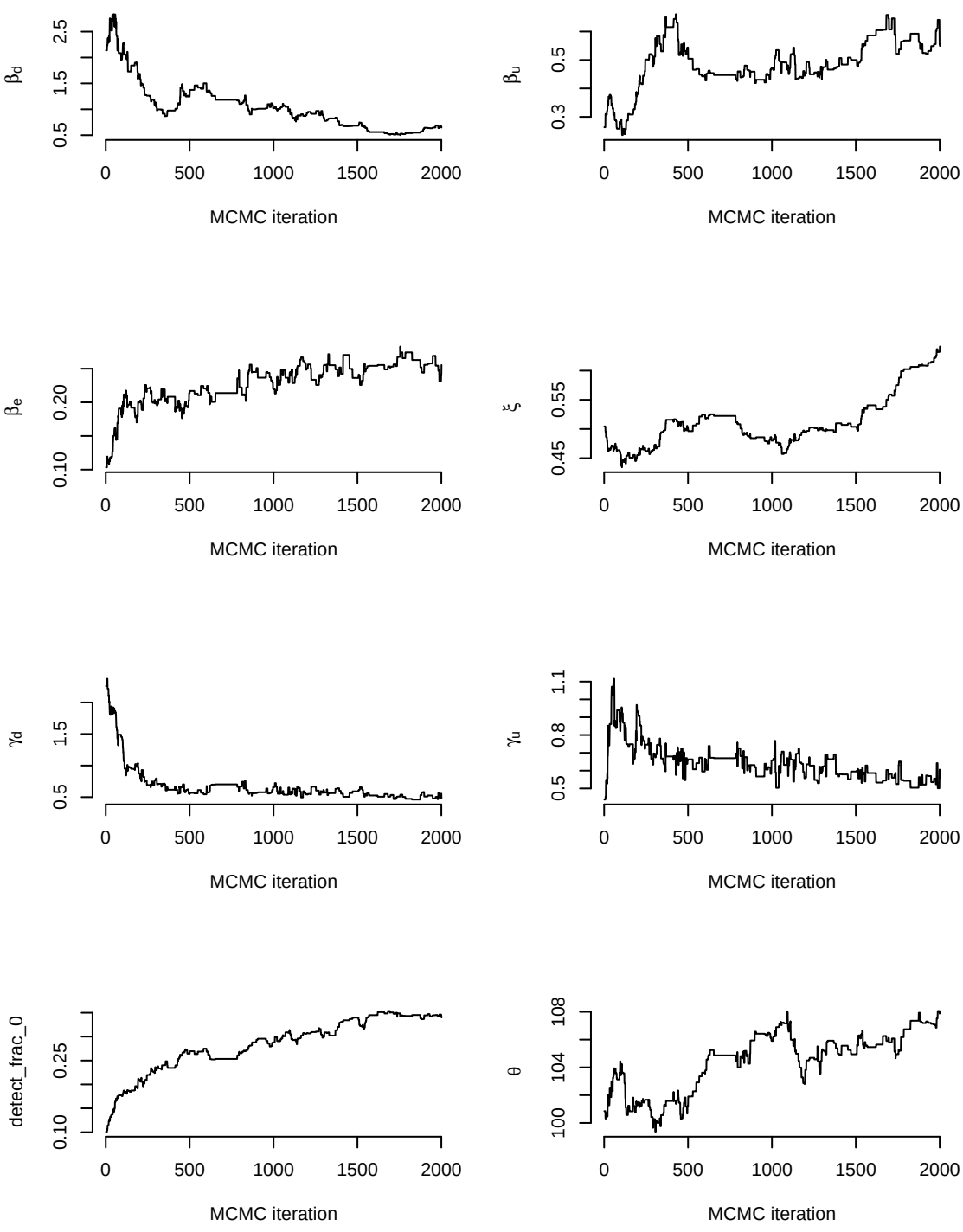
```
n <- 1   # number of mcmc chains

# Parameters to estimate
estpars <- c("beta_d", "beta_u",  "beta_e", "beta_red_factor",
             "gamma_u", "gamma_d", "detect_frac_0", "theta")

# Set noise level for parameter random walk for proposals
rw.sd <- c(beta_d = 0.05, beta_u = 0.05, beta_e = 0.05,
           beta_red_factor = 0.005, gamma_u = 0.1, gamma_d = 0.1,
           detect_frac_0 = 0.005, theta = 0.5)

out_mcmc <- pmcmc(
  pomp(
    covid_ga_pomp,
    dprior = prior_dens,
    paramnames = c("beta_d", "beta_u", "beta_e", "beta_red_factor",
                   "gamma_u", "gamma_d", "detect_frac_0", "theta")
  ),
```

```
  Nmcmc = 2000,
  Np = 2000,
  proposal = mvn.diag.rw(rw.sd),
  verbose = TRUE
)
```

Note that the above results in no errors, no warnings, and no particle failures. The acceptance rates hovers around 0.2 (starting high around 0.5, and ending low around 0.1, by the final MCMC iterations).

Let's take a look at the chains.

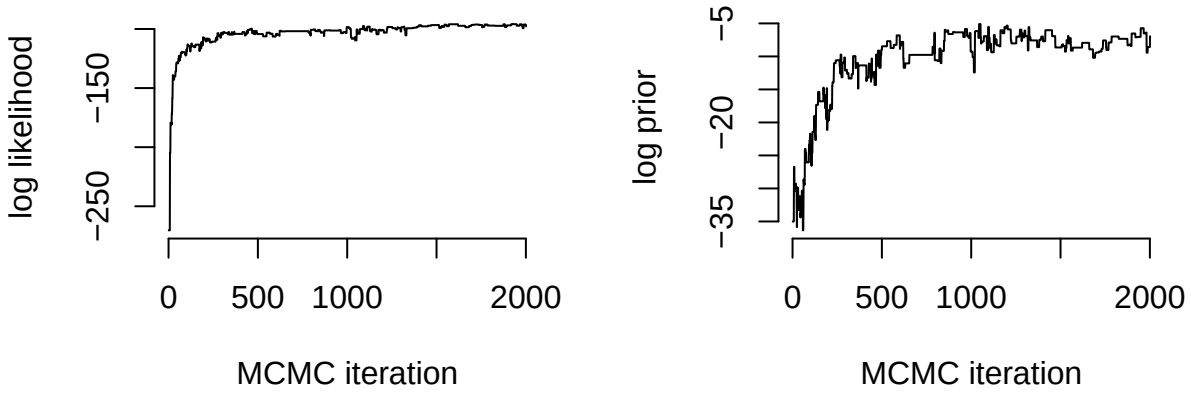
```
chain <- as.data.frame(out_mcmc@traces)

par(mfrow = c(4,2))
plot(exp(chain$beta_d)*10600000, type = "l", bty = "n",
     xlab = "MCMC iteration", ylab = expression(beta[d]))
plot(exp(chain$beta_u)*10600000, type = "l", bty = "n",
     xlab = "MCMC iteration", ylab = expression(beta[u]))
plot(exp(chain$beta_e)*10600000, type = "l", bty = "n",
     xlab = "MCMC iteration", ylab = expression(beta[e]))
plot(chain$beta_red_factor, type = "l", bty = "n",
     xlab = "MCMC iteration", ylab = expression(xi))
plot(chain$gamma_d, type = "l", bty = "n",
     xlab = "MCMC iteration", ylab = expression(gamma[d]))
plot(chain$gamma_u, type = "l", bty = "n",
     xlab = "MCMC iteration", ylab = expression(gamma[u]))
plot(chain$detect_frac_0, type = "l", bty = "n",
     xlab = "MCMC iteration", ylab = "detect_frac_0")
plot(chain$theta, type = "l", bty = "n",
     xlab = "MCMC iteration", ylab = expression(theta))
```



And we can look at the log likelihood and the log prior density over the MCMC iterations.

```
par(mfrow = c(1,2))
plot(chain$loglik, type = "l", bty = "n",
     xlab = "MCMC iteration", ylab = "log likelihood")
```

```
plot(chain$log.prior, type = "l", bty = "n",
     xlab = "MCMC iteration", ylab = "log prior")
```



Next steps

It is clear that the MCMC chains have not reached stationarity. Not sure about convergence yet because I only ran one chain. But, given the identifiability issues, here are what I think are the next steps:

1. Fit with MIF to identify starting parameters to give the pMCMC a "helping hand" (Ionides and King refer to this in their Cholera PNAS paper).
2. Really nail down the priors.
    - In some cases, I think we will need very, very informative priors that almost result in fixed parameters unless the data really overwhelms the prior.
3. Do above with simulated data first, then real data.