Bayesian statistics and pomp

Qianying (Ruby) Lin   Spencer J. Fox

Lecture outline

1. Motivating Bayesian statistics
2. Short introduction to Bayesian statistics and theory
3. Introduction to MCMC
4. Introduction to PMCMC
5. Simple influenza case study

What we have covered so far

$$p(y|\theta)$$

- y can be thought of as your data or observations
- θ can be thought of as the model or parameter values
- Called the "Likelihood"

Issues with maximum likelihood estimation (MLE)

- Assumes results occur with some given “frequency” over period time or replicates/repeated experiments
  - If we had the same outbreak hundreds of time, what proportion of them would provide confidence intervals that contain the true value for the R_0
- Some difficulties in constraining parameter values based on outside data, information, or expert opinion
- Just not really intuitive...
  - We typically want to say something about the parameters based on the data, $p(\theta|y)$

Bayesian statistics

- Bayes theorem provides an intuitive framework to update parameter estimates based on both prior knowledge and experimental data
- End result is a posterior distribution, $p(\theta|y)$, directly describing the parameter and model of interest
- Easy to communicate results
  - "The reproduction number is estimated to be x, with a 95% credible interval from y to z"
- Issues
  - Computationally expensive
  - Without enough data, prior can bias posterior distribution, but this is what you want!

Bayes theorem 1

$$p(\theta|y) = \frac{p(y|\theta)p(\theta)}{p(y)}$$

- $p(\theta|y)$ is the posterior distribution
- $p(y|\theta)$ is the likelihood
- $p(\theta)$ is the prior distribution
- $p(y)$ is the marginal distribution (sometimes called a normalizing constant as it doesn't depend on the parameters)

Bayes theorem 2

$$p(\theta|y) = \frac{p(y|\theta)p(\theta)}{p(y)}$$

- $p(y) = \int p(y|\theta)p(\theta)d\theta$
- Probability of observing y marginal over all possible values of θ
- Typically is very difficult to calculate
- The good news is that $p(y)$ is a constant

Bayes theorem 3

$$p(\theta|y) \propto p(y|\theta)p(\theta)$$

- Since $p(y)$ is a constant, the posterior distribution is proportional to the likelihood times the prior
- If we can solve this we can get the posterior distribution because $\int p(\theta|y)d\theta = 1$
- Intuitively our parameter estimates are based on a combination of our observations $p(y|\theta)$ and our prior beliefs $p(\theta)$
- Only need to sample from the likelihood and prior distribution to get the posterior

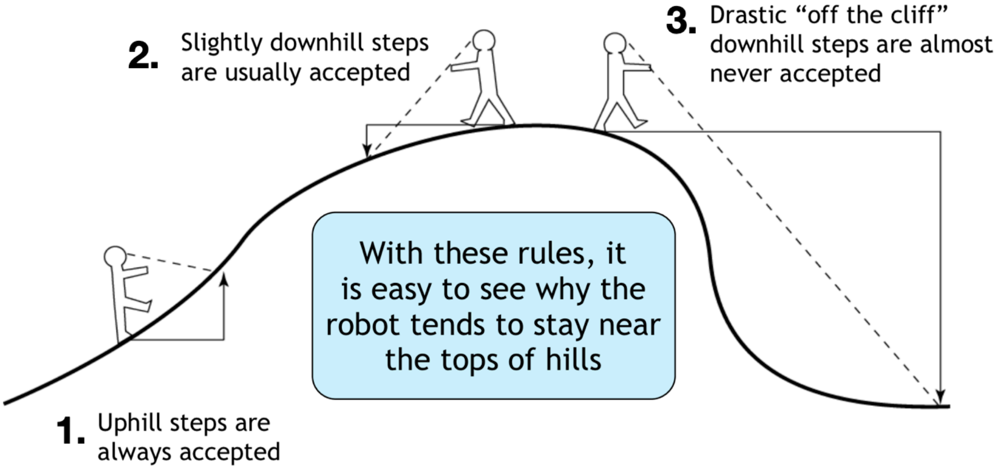
How do we do so?

- Many ways to do so (and many software packages), but we're only going to talk about one...
- Markov chain Monte Carlo (MCMC) is a class of algorithms used to draw samples from a probability distribution
- Will not cover the theoretical details, but will attempt to motivate

Assume you have an unknown probability distribution (hill) to explore... How would you do so?

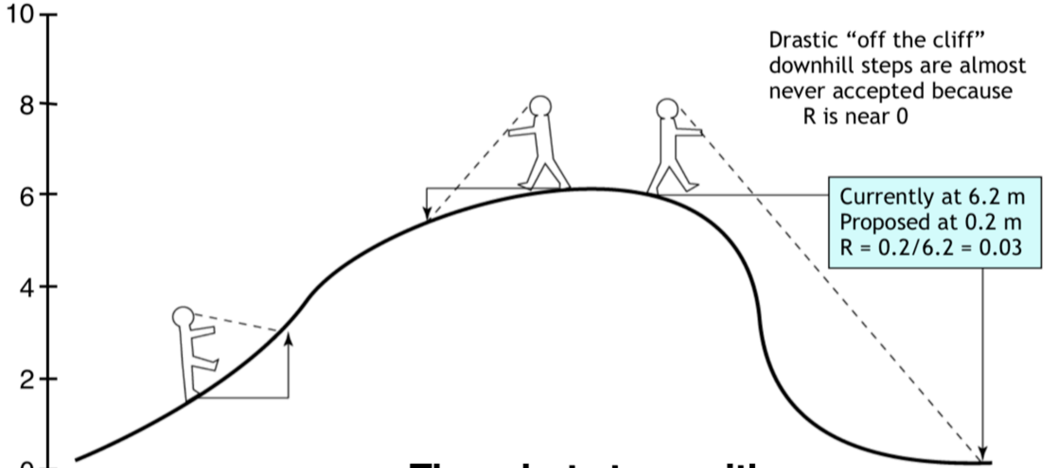
- You don't know where it is in parameter space
- You don't know it's shape

MCMC Robot Rules


2. Slightly downhill steps are usually accepted
3. Drastic "off the cliff" downhill steps are almost never accepted
With these rules, it is easy to see why the robot tends to stay near the tops of hills
1. Uphill steps are always accepted

Paul O. Lewis (2014 Woods Hole Molecular Evolution Workshop)

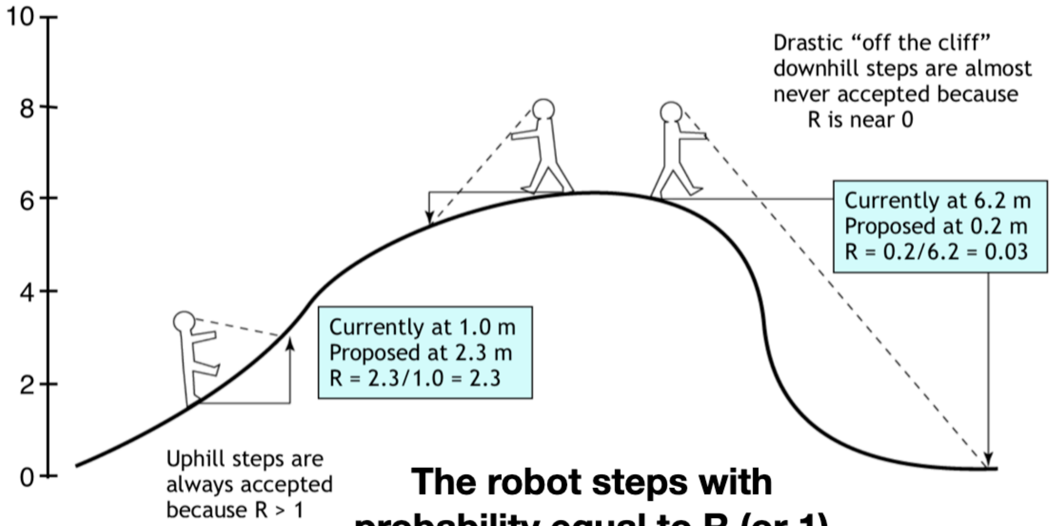
MCMC Robot Rules (actual)


10
8
6
4
2
0
Drastic "off the cliff" downhill steps are almost never accepted because R is near 0
Currently at 6.2 m
Proposed at 0.2 m
R = 0.2/6.2 = 0.03

The robot steps with probability equal to R (or 1)

Paul O. Lewis (2014 Woods Hole Molecular Evolution Workshop)

MCMC Robot Rules (actual)

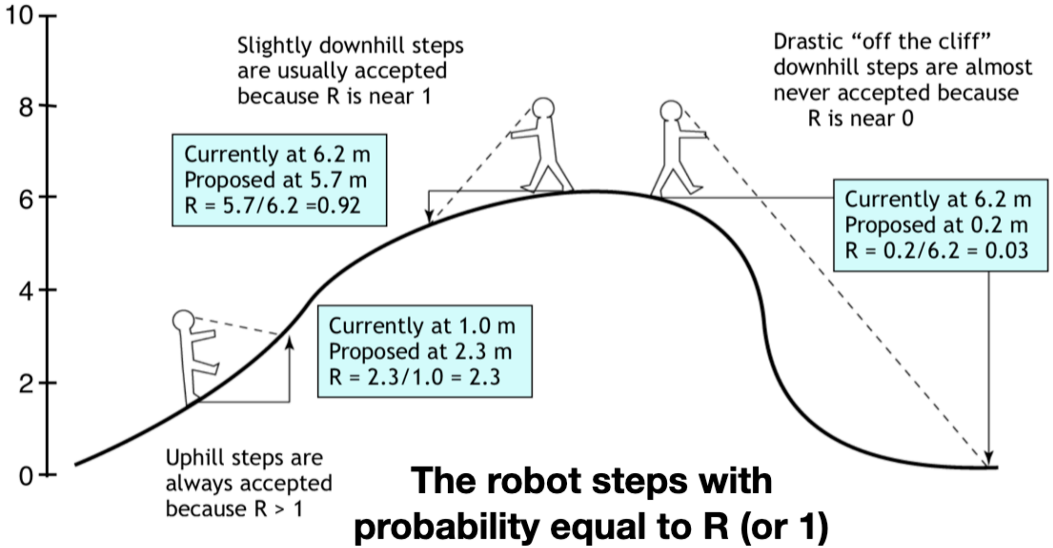


10
8
6
4
2
0
Drastic “off the cliff” downhill steps are almost never accepted because R is near 0
Currently at 6.2 m
Proposed at 0.2 m
R = 0.2/6.2 = 0.03
Currently at 1.0 m
Proposed at 2.3 m
R = 2.3/1.0 = 2.3
Uphill steps are always accepted because R > 1

The robot steps with probability equal to R (or 1)

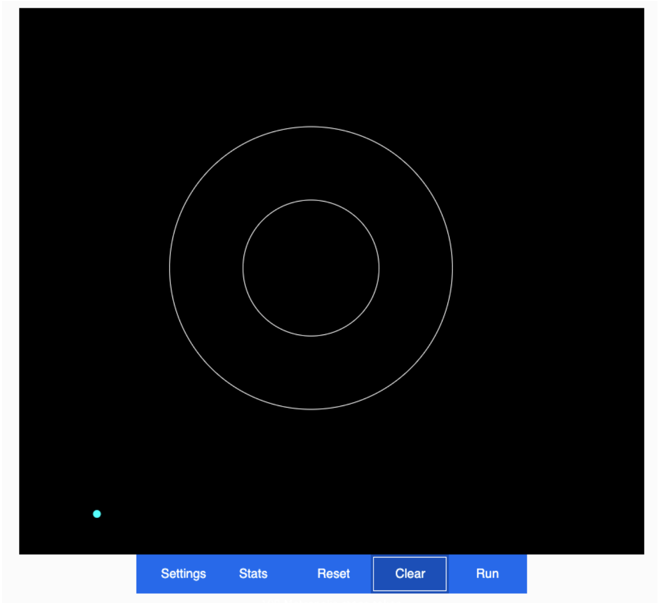
Paul O. Lewis (2014 Woods Hole Molecular Evolution Workshop)

MCMC Robot Rules (actual)

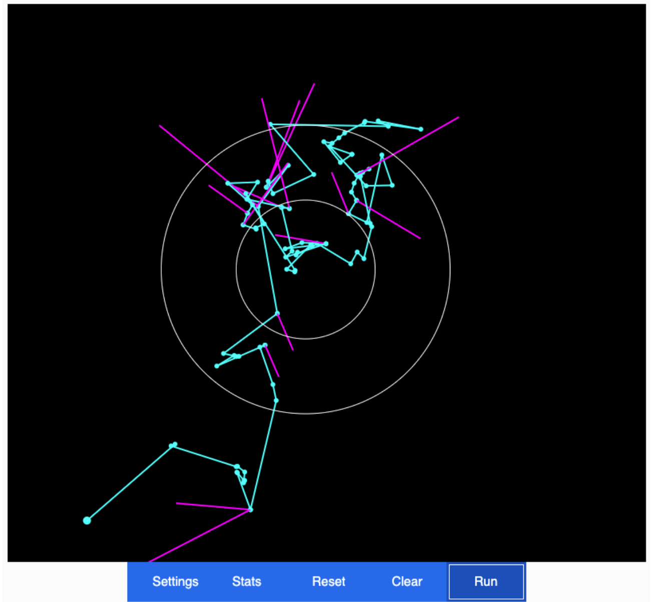

10
8
6
4
2
0
Slightly downhill steps are usually accepted because R is near 1
Currently at 6.2 m
Proposed at 5.7 m
R = 5.7/6.2 =0.92
Drastic "off the cliff" downhill steps are almost never accepted because R is near 0
Currently at 6.2 m
Proposed at 0.2 m
R = 0.2/6.2 = 0.03
Currently at 1.0 m
Proposed at 2.3 m
R = 2.3/1.0 = 2.3
Uphill steps are always accepted because R > 1
The robot steps with probability equal to R (or 1)

Paul O. Lewis (2014 Woods Hole Molecular Evolution Workshop)

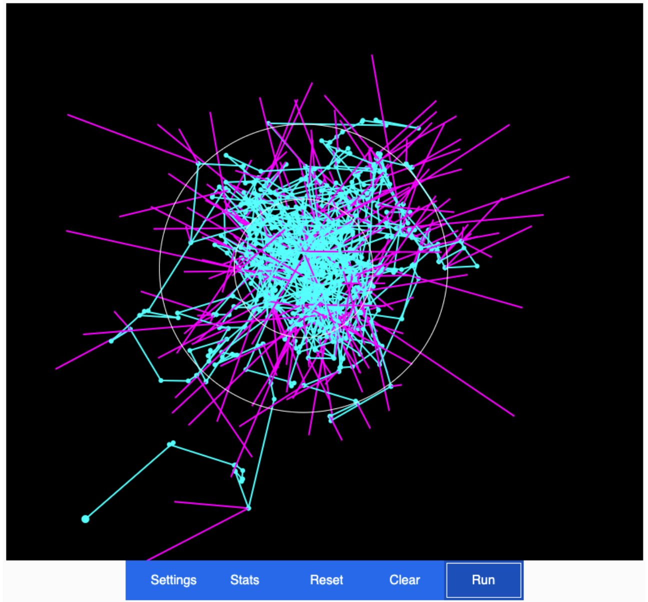
MCMC Demonstration (https://plewis.github.io/applets/mcmc-robot/)



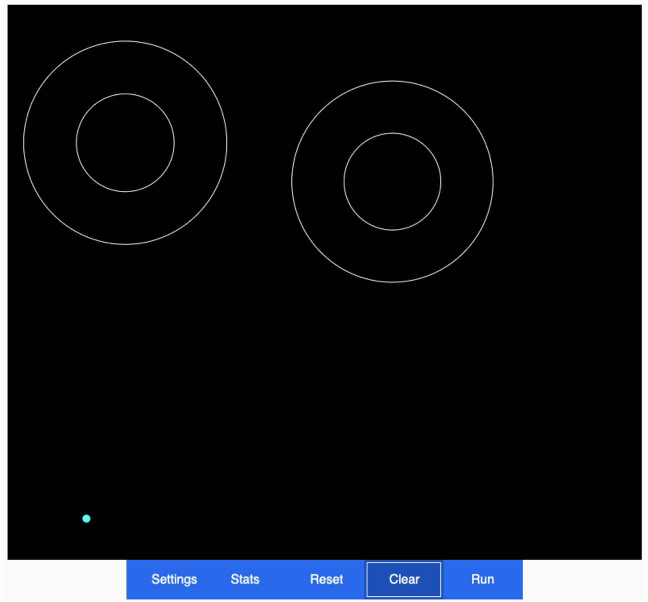
MCMC Demonstration (https://plewis.github.io/applets/mcmc-robot/)



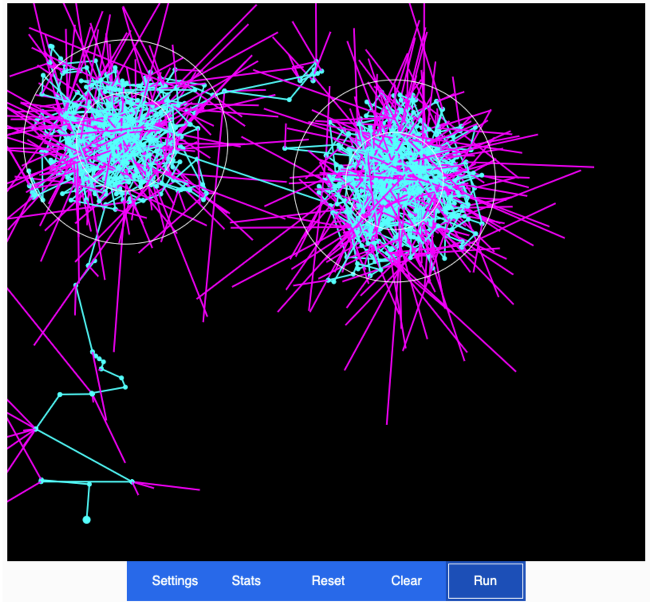
MCMC Demonstration (https://plewis.github.io/applets/mcmc-robot/)



MCMC Demonstration (https://plewis.github.io/applets/mcmc-robot/)



MCMC Demonstration (https://plewis.github.io/applets/mcmc-robot/)



Steps for Metropolis-Hastings MCMC

1. Choose a reasonable starting place for parameters (θ)
2. Propose new parameter values (θ')
3. Calculate the proposal probability α
4. Accept (change parameters to θ') or reject (keep parameters at θ) with probability α
5. Repeat steps 2-5 for as many MCMC iterations as desired

Steps for Metropolis-Hastings MCMC

1. Choose a reasonable starting place for parameters (θ)
2. Propose new parameter values (θ')
3. Calculate the proposal probability α
4. Accept (change parameters to θ') or reject (keep parameters at θ)
5. Repeat steps 2-5 for as many MCMC iterations as desired

Where

$$\alpha = \text{min}(1, \rho)$$

and

$$\rho = \frac{p(\theta'|y)}{p(\theta|y)} \cdot \frac{G(\theta|\theta')}{G(\theta'|\theta)}$$

ρ equals the ratio of posterior distributions multiplied by the proposal probability ratio

$$\rho = \frac{p(\theta^{'}|y)}{p(\theta|y)} \cdot \frac{G(\theta|\theta^{'})}{G(\theta^{'}|\theta)}$$

- For symmetric proposal distributions (random walks) $\frac{G(\theta|\theta')}{G(\theta'|\theta)} = 1$
- Notice that the posterior denominators from before ($p(y)$) cancel each other out in this ratio

We can simplify the acceptance probability significantly

$$\rho = \frac{p(y|\theta^{'})p(\theta^{'})}{p(y|\theta)p(\theta)}$$

- Only depends on the likelihood and prior probabilities

Prior parameter distributions

- Prior distributions assign probabilities to specific parameter values and are created separately for every parameter
- Other than strict constraints, we expect the data and likelihood to drive the posterior distribution, but always good idea to check the impact the prior distributions have in a sensitivity analysis

Three main types of prior distributions

1. Informative priors craft a distribution based on previous scientific studies
   - Typically only used if a specific quantity is well known from outside data (e.g. the infection fatality rate or infectious period)
2. Weakly informative priors craft a distribution with reasonable constraints
   - Used for regularization (e.g. to "suggest" that a parameter is most likely to be positive)
3. Uninformative (flat priors) assign equal probabilities across a range of plausible values
   - Can constrain parameters to be strictly positive or in biologically/epidemiologically relevant range
   - Not always "uninformative"

Questions about MCMC?

PMCMC

$$p(y|\theta)p(\theta)$$

- Standard MCMC algorithm assumes a deterministic likelihood
- When process stochasticisty is included we use the particle filter to obtain likelihood
- Particle filter (Sequential Monte Carlo procedure) approximates $p(y|\theta)$, but with variability
    - Can complicate things a bit, but usually not an issue

Further PMCMC resources

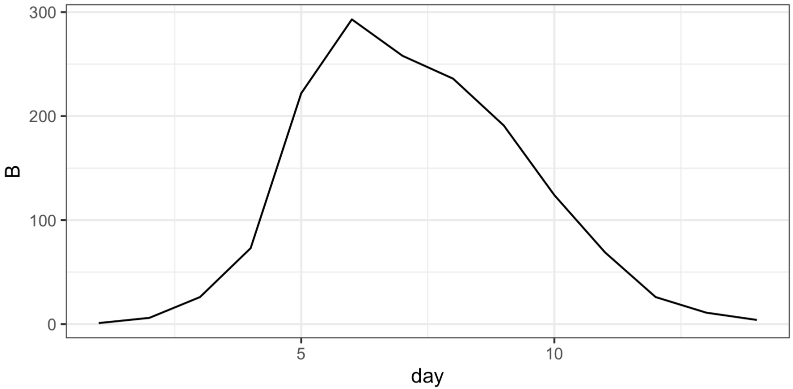
- For an awesome field-specific explanation
  - Introduction to particle Markov-chain Monte Carlo for disease dynamics modellers
- For some considerations with modeling and inference
  - Choices and trade-offs in inference with infectious disease models
- For an example of how it has been used in recent epidemiological literature
  - Estimating SARS-CoV-2 transmission parameters between coinciding outbreaks in a university population and the surrounding community
- For working with pmcmc in the `pomp` package
  - Getting started with pomp

Considerations for moving from `mif2` to `pmcmc` in `pomp`

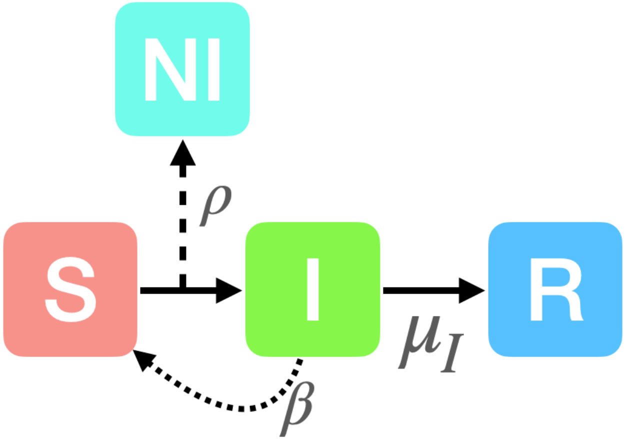
1. Specify the prior distributions for all parameters
2. Use the `pmcmc()` function
3. Use MCMC diagnostics to check convergence, etc.

Influenza boarding school example

- 1978 influenza epidemic in boarding school that infected much of the school
- Data are actually kids in beds, but we are assuming it's new infections



Modeling as an SIR model with reporting of new infections



Modeling as an SIR model with reporting of new infections

```
rproc <- Csnippet("
  double N = 2000;
  double t1 = rbinom(S,1-exp(-Beta*I/N*dt));
  double t2 = rbinom(I,1-exp(-mu_I*dt));
  S  -= t1;
  I  += t1 - t2;
  NI += t1;
  R += t2;
")

rmeas <- Csnippet("
  B = rpois(rho*NI+1e-6);
")
```

Specifying the prior distribution

```r
priorDens <- Csnippet("
  lik = dunif(Beta, 1, 4, 1) +
        dunif(mu_I, 0.5, 3, 1) +
        dunif(rho, 0.5, 1, 1);
  if (!give_log) lik = exp(lik);
")
```

- Add the densities (probabilities) for each parameter value independently
- Include the same log functionality used before
- We are specifying a plausible range of values for each parameter here

Running an MCMC chain

```
flu |> ## Standard pomp object that has already been created
  pomp(dprior = priorDens, ## Prior specified from previous slide
       params = sim_params, ## Parameter starting point
       paramnames=c("Beta","mu_I","rho")) |> ## Parameter names
  pmcmc(Nmcmc = 10000, ## Number of MCMC iterations
        Np = 200, ## Number of particles to use
        proposal = mvn_diag_rw(rw.sd = c(Beta=0.3, mu_I=0.3, rho=0.1))
  ) -> test_mcmc
```

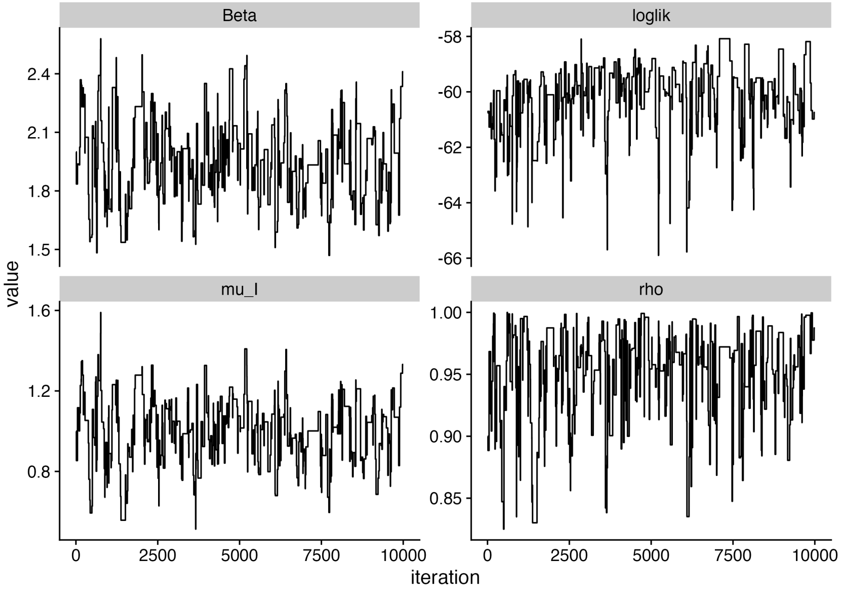
Proposal distributions in `pomp`

- `mvn_diag_rw(rw.sd)` - you provide the standard deviations for the proposals for each parameter
- `mvn_rw(rw.var)` - you provide the variance/covariance matrix for proposals
- `mvn_rw_adaptive()` - you provide either of the above and it attempts to automatically "tune" parameters to achieve good MCMC mixing

Typically you need to test different values initially

```r
flu |> ## Standard pomp object that has already been created
  pomp(dprior = priorDens, ## Prior specified from previous slide
       params = sim_params, ## Parameter starting point
       paramnames=c("Beta","mu_I","rho")) |> ## Parameter names
  pmcmc(Nmcmc = 10000, ## Number of MCMC iterations
        Np = 200, ## Number of particles to use
        proposal = mvn_diag_rw(rw.sd = c(Beta=0.3, mu_I=0.3, rho=0.1))
  ) -> test_mcmc
```

Diagnosing chain mixing



Diagnosing chain mixing - autocorrelation

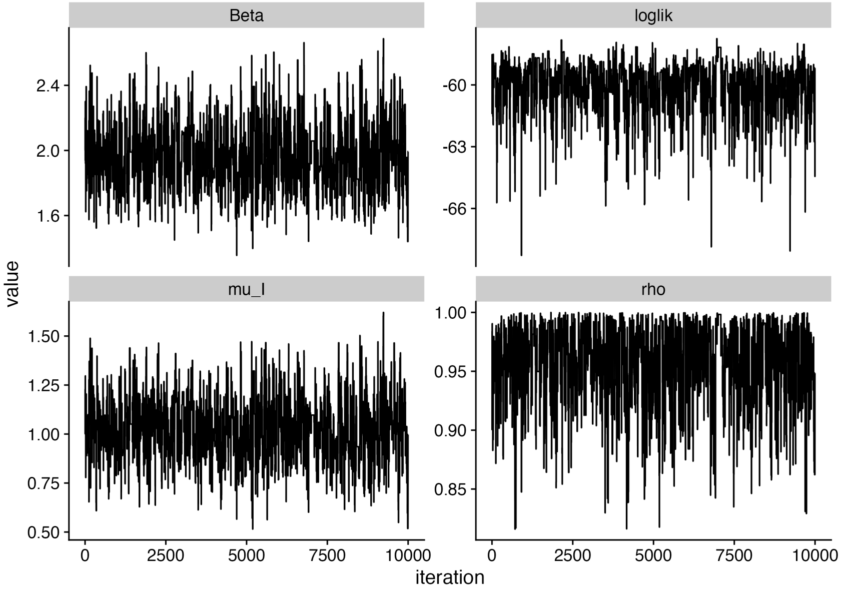
```r
library(coda)

test_mcmc |>
  traces() |>
  autocorr.diag(lags=c(10, 50, 100))
```

```
            loglik log.prior      Beta      mu_I        rho
Lag 10   0.7353053 0.9990001 0.8494006 0.8366332 0.73641949
Lag 50   0.2745494 0.9950005 0.5098440 0.4723025 0.25603762
Lag 100 0.1485070 0.9900010 0.2544408 0.2222884 0.08499315
             mu_R1
Lag 10   0.9990001
Lag 50   0.9950005
Lag 100 0.9900010
```

Second step often uses the empirical covariance matrix for proposals to improve mixing

```r
flu |>
  pomp(dprior = priorDens,
       params = sim_params, ##Using the sim params as a starting spot
       paramnames=c("Beta","mu_I","rho")) |>
  pmcmc(Nmcmc = 10000,
        Np = 200,
        proposal = mvn_rw(covmat(test_mcmc, thin = 50))
  ) -> test_mcmc2
```

Improved trace plots!



Improved autocorrelation!

```r
test_mcmc2 |>
  traces() |>
  autocorr.diag(lags=c(10, 50, 100))
```

```
              loglik log.prior          Beta       mu_I
Lag 10    0.32899769 0.9990001  0.3728946153 0.38301448
Lag 50    0.01244695 0.9950005 -0.0005220889 0.01322092
Lag 100  -0.02213489 0.9900010  0.0250372644 0.03490559
                 rho     mu_R1
Lag 10    0.40141937 0.9990001
Lag 50    0.02678762 0.9950005
Lag 100  -0.03388893 0.9900010
```

Estimating posterior distributions

1. Once you are happy with the mixing of your `pmcmc` you are ready to do a run for estimating posteriors
2. First randomly choose 3-5 starting parameter values (example uses Latin-hypercube sampling)
3. Run (in parallel or not depending on computational time) a chain intialized with each
4. Diagnose chain mixing with trace plots and Gelman-Rubin convergence diagnostic
5. Remove the burn-in period and thin based on diagnostics
6. Summarize parameter posterior distributions and credible intervals

Summary process for PMCMC in `pomp`

1. Create `pomp` object exactly as normal
2. Create the prior distributions for parameters
3. Run an initial `pmcmc` with `mvn_diag_rw` proposal to make sure it's working and diagnose
4. Randomly sample 3-5 parameter starting conditions
5. Run a PMCMC chain with `mvn_rw()` proposal and the variance/covariance matrix from the first run for each of the initial parameter combinations
6. Diagnose mixing and convergence (alter components as needed)
7. Summarize parameters

Activity: how do stochastic and deterministic models differ?

1. Go to https://plewis.github.io/applets/mcmc-robot/, and play with different MCMC parameters and variations
2. Download the exercise code for the influenza boarding school example and test the impact of the following on mixing and traceplots:
    - Different parameters for the proposal distribution
    - Different starting parameter values
    - Different number of particles

License, acknowledgments, and links

- This lesson is prepared for the Simulation-based Inference for Epidemiological Dynamics module at the Summer Institute in Statistics and Modeling in Infectious Diseases, SISMID.
- The materials build on previous versions of this course and related courses.

- Produced with R version 4.4.1 and pomp version 5.9.
- Compiled on 2024-06-13.

Back to Lesson

pomp homepage