Basic SIR fitting - Details

June 19, 2017

This vignette provides technical details of the `fitsir` package.

Contents

1 Starting function

To avoid falling into local minima, it is important to use a good starting parameter. `fitsir` provides a function (`startfun`) that estimates a reasonable starting parameter using a built-in algorithm. This section describes the method used in findiing starting parameters.

1.1 Prevalence

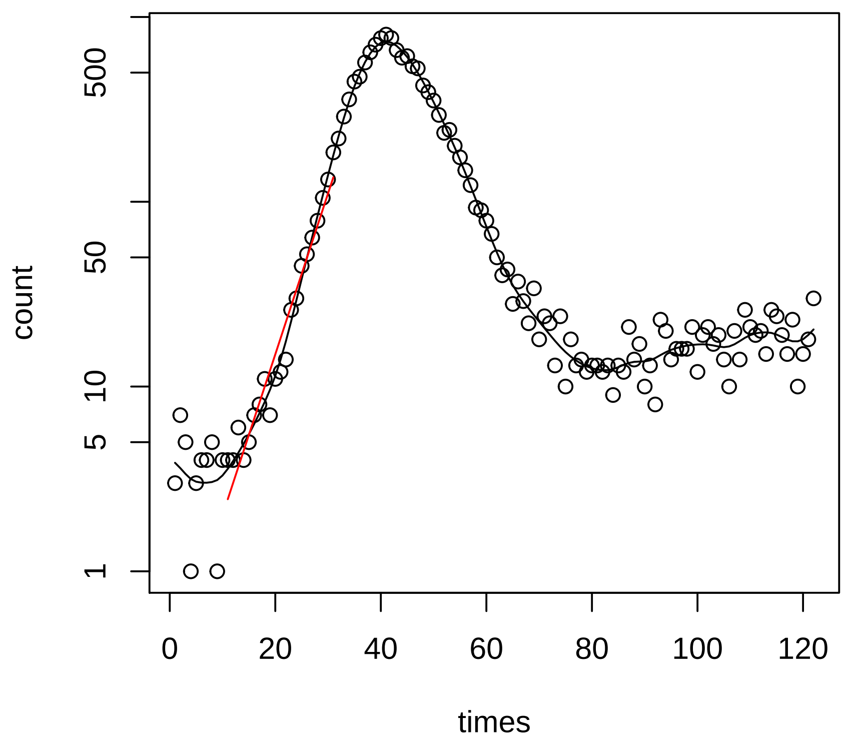
First, let's start with the simplest case. Although it is practically impossible to obtain a prevalence data, prevalence trajectory of an SIR model has important properties that allow us to estimate the parameter based on the trajectory only.

Let's take a look at the 1918 Philadelphia Flu data as an example (we're assuming that the data represents prevalence). We start by fitting a spline to the data to obtain a continuous trajectory.

```
library(fitsir)
times <- seq_along(phila1918$date)
count <- phila1918$pim
ss <- fitsir:::smooth.spline2(times, count, itmax = 100, relpeakcrit = 0.1)
```

Then, we estimate the little r ($\beta - \gamma$) by finding an appropriate fitting window using spline fit and fitting a linear model to the real data. Hereafter, we denote this quantity as a_0 instead of r.

```
ss.data <- data.frame(times = times, count = exp(predict(ss)$y))
ss.tmax <- ss.data$times[which.max(ss.data$count)]
ss.t1 <- min(times)+0.25*(ss.tmax-min(times))
ss.t2 <- min(times)+0.75*(ss.tmax-min(times))

m <- lm(log(count)~times,data=subset(ss.data,times<=ss.t2 & times>=ss.t1))
(r <- unname(coef(m)[2])) ##beta - gamma

## [1] 0.2005722
```



Notice that this data has trails in the beginning of the epidemic and does not follow a typical SIR shape. So taking the initial value from the data or from the spline will not correctly estimate the initial size of the epidemic. To avoid this issue, we predict the initial value, $I(0)$, by extrapolating the linear fit. We denote this quantity as b_0 hereafter.

```
(iniI <- unname(exp(predict(m, data.frame(times=times[1])))))

## [1] 0.3304682
```

Then, we look at the second derivative at the peak:

$$d^2 \log I/dt^2 = -\beta S'/N = \beta(-\beta SI/N)/N = -\beta^2/N^2(\gamma N/\beta)I = -\beta\gamma I/N,$$

Rearranging, we get

$$c_0 = -d^2 \log I/dt^2/I_{\text{peak}} = \beta\gamma/N$$

We can obtain this quanity by using the spline fit.

```
Qp.alt <- predict(ss,ss.tmax,deriv=2)$y ## second derivative
Ip <- exp(max(predict(ss)$y)) ## I_peak
(c <- -Qp.alt/Ip)

## [1] 4.360793e-05
```

Finally, we can obtain the fourth equation as follows:

$$\begin{aligned}
dI/dt &= \beta SI/N - \gamma I \\
&= \beta(N - I - R)I/N - \gamma I \\
&= \beta\left(N - I - \int_0^t \gamma I(s)ds\right) I/N - \gamma I \\
0 &= (\beta - \gamma)I_{\text{peak}} - \frac{\beta}{N} I_{\text{peak}}^2 - \frac{\beta\gamma}{N} I_{\text{peak}} \int_0^{t_{\text{peak}}} I(s)ds
\end{aligned}$$

For convenience, we denote $d_0 = \int_0^{t_{\text{peak}}} I(s)ds$. Here, the integral is calculated numerically:

```
t.diff <- diff(times)
t.diff <- c(t.diff[1], t.diff)
ss.int <- transform(ss.data, int = count * t.diff)
ss.int <- ss.int[times<ss.tmax,]

d0 <- sum(ss.int[,3])
```

Altogether, we have the following system of non-linear equations:

$$\begin{aligned}
a_0 &= \beta - \gamma \\
b_0 &= N i_0 \\
c_0 &= \beta\gamma/N \\
d_0 &= a_0/c_0 - I_{\text{peak}}/\gamma
\end{aligned}$$

Rearranging the four equations provided, we get

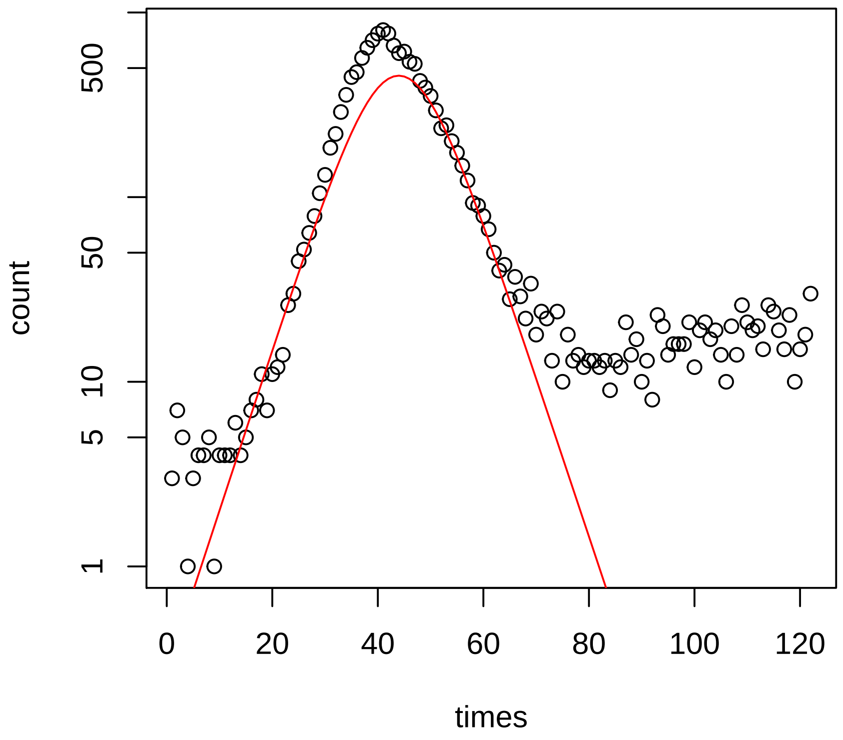
$$\begin{aligned}
\gamma &= c_0 I_{\text{peak}}/(a_0 - c_0 d_0) \\
\beta &= \gamma + a_0 \\
N &= \beta\gamma/c_0 \\
i_0 &= b_0/N
\end{aligned}$$

Clearly, $a_0 - c_0 d_0$ must be positive in order to get a biologically meaningful parameter ($\gamma > 0$). If the data is noisy, we may get a negative quantity due to an inaccurate estimation of the second derivative. So we adjust d_0 to avoid this value from becoming negative:

```
while(r - c * d0 < 0){
    ss.int <- ss.int[-nrow(ss.int),]
    d0 <- sum(ss.int[,3])
}
```

Therefore, we get

```
prev.pars <- list()
(prev.pars <- within(prev.pars,{
    gamma <- c * Ip/(r - c * d0)
    beta <- gamma + r
    N <- beta*gamma/c
    i0 <- iniI/N
}))

## $i0
## [1] 1.753605e-07
##
## $N
## [1] 1884507
##
## $beta
## [1] 9.166129
##
## $gamma
## [1] 8.965557

plot(times, count, log="y")
lines(times, SIR.detsim(times, unlist(prev.pars)[c(3, 4, 2, 1)]), col="red")
```



1.2 Death

Estimating a starting parameter from a death cases curve is very similar to what we did so far. Note that death cases are counted by using $R(t_i + \delta t) - R(t_i)$. Approximately, we have

$$\frac{R(t_i + \Delta t) - R(t_i)}{\Delta t} \approx \frac{dR}{dt} = \gamma I$$

So the estimation of the starting parameter is based on the transformed data rather than the raw data:

```
t.diff <- diff(times)
t.diff <- c(t.diff[1], t.diff)
count.orig <- count
count <- count/t.diff
```

Note that we have to fit a spline to a transformed data instead. However, since $\Delta t = 1$ for this data, we ignore this step.

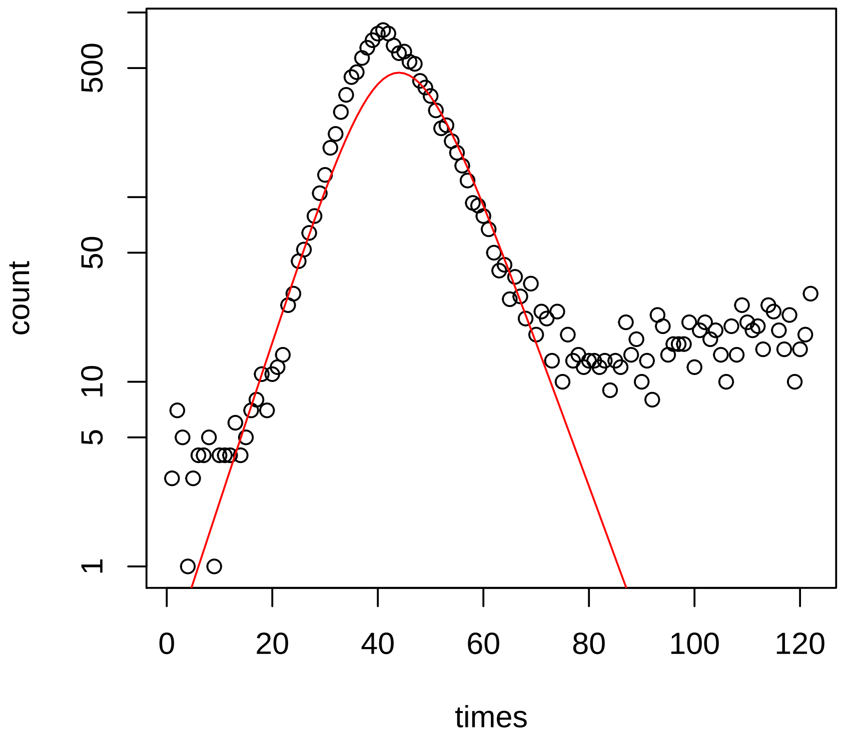
As this transformed data is just a multiple of the prevalence trajectory, b_0 and d_0 of the transformed data will just be a multiple of those of the prevalence data:

$$\begin{aligned}
a_0 &= \beta - \gamma \\
b_0 &= \gamma N i_0 \\
c_0 &= \beta\gamma/N \\
d_0 &= \gamma a_0/c_0 - I_{\text{peak}}
\end{aligned}$$

Notice that c_0 stays the same as before because the it's the ratio between the second derivative at the peak and the peak value itself. Rearranging, we get

$$\begin{aligned}
\gamma &= c_0(d_0 + I_{\text{peak}})/a_0 \\
\beta &= \gamma + a_0 \\
N &= \beta\gamma/c_0 \\
i_0 &= b_0/(\gamma N)
\end{aligned}$$

```r
death.pars <- list()
(death.pars <- within(prev.pars,{
    gamma <-  c*(d0 + Ip)/r
    beta <- gamma + r
    N <- beta*gamma/c
    i0 <- iniI/(gamma*N)
}))

## $i0
## [1] 8.224505e-06
##
## $N
## [1] 35175.73
##
## $beta
## [1] 1.342863
##
## $gamma
## [1] 1.142291

plot(times, count, log="y")
lines(times, SIR.detsim(times, unlist(death.pars)[c(3, 4, 2, 1)], type="death"), col="red")
```



1.3 Incidence

Incidence is calculated by $S(t_i) - S(t_i + \delta t)$. Approximately, we have

$$\frac{S(t_i) - R(t_i + \Delta t)}{\Delta t} \approx -\frac{dS}{dt} = \beta SI/N$$

Once again, we use the transformed data here. We still have $a_0 = \beta - \gamma$ but the initial value, b_0, is equal to $\beta S(0)I(0)/N$ in this case. As $S(0) \approx N$, we write $b_0 = \beta N i_0$. Finally, c_0 is obtained by taking the second derivative of βSI at the peak. Here we take the second derivative of $\log(\beta SI)$ instead for convenience:

$$\begin{aligned}
\frac{d\log(\beta SI)}{dt} &= \frac{\beta}{N}(S-I)-\gamma \\
\frac{d^2\log(\beta SI)}{dt^2} &= \frac{\beta}{N}\left(-2\frac{\beta SI}{N}+\gamma I\right) \\
&= \frac{\beta}{N}\left(-2I_{\text{peak}}+\gamma\frac{N}{\beta S}I_{\text{peak}}\right) \\
c_0 &= \frac{\beta}{N}\left(2-\gamma\frac{N}{\beta S}\right)
\end{aligned}$$

To allow for simple computation, we assume that $S \approx N$ near the peak. This is not accurate but this assumption makes the algorithm a lot simpler:

$$\begin{aligned}
c_0 &= \frac{\beta}{N}\left(2-\frac{\gamma}{\beta}\right) \\
&= \frac{\beta+a_0}{N}
\end{aligned}$$

Finally, we obtain the following set of equations:

$$\begin{aligned}
a_0 &= \beta-\gamma \\
b_0 &= \beta N i_0 \\
c_0 &= (\beta+a_0)/N
\end{aligned}$$

To obtain the last equation, we use the final size equation (CITE Miller 2012):

$$S(\infty)/N = \exp\left[-\frac{\beta}{\gamma}[1-S(\infty)/N]\right]$$

Given an incidence curve, final size can be estimated numerically. We estimate the initial increasing and later decreasing incidence counts by linear fits and in between by a spline:

```
ss.t3 <- floor(ss.tmax+0.25*ss.tmax)
ss.t4 <- ceiling(ss.tmax+0.75*ss.tmax)
m2 <- lm(log(count)~times,data=subset(ss.data,times<=ss.t4 & times>=ss.t3))

times.predict1 <- seq(min(times), ss.t2, by = t.diff[length(t.diff)])
times.predict2 <- seq(ceiling(ss.t3), 3*ss.tmax, by = t.diff[length(t.diff)])
count.predict1 <- exp(predict(m, data.frame(times = times.predict1)))
count.predict2 <- exp(predict(m2, data.frame(times = times.predict2)))
finalsize <- sum(count.predict1) + sum(count.orig[times > ss.t2 & times <= ss.t3]) + sum(cou
```

Using $R(\infty) = N - S(\infty)$, we get

$$R(\infty)/N = 1-\exp\left[-\frac{\beta}{\gamma}R(\infty)/N\right]$$

Then, we can write

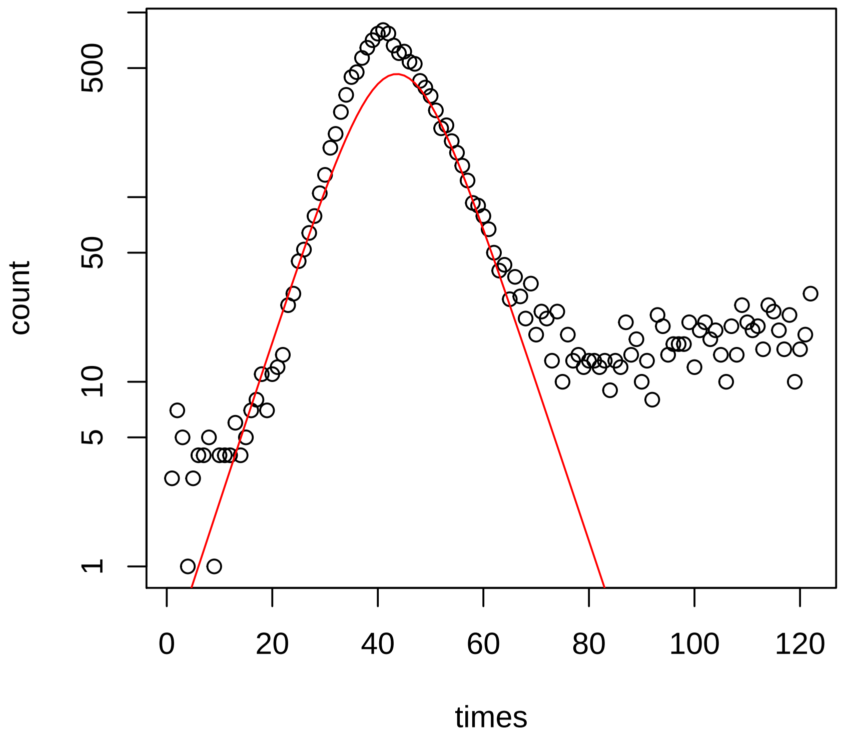
$$f(\beta) = \frac{R(\infty)}{N} - \left(1 - \exp\left(-\frac{\beta}{\beta - r}\frac{R(\infty)}{N}\right)\right),$$

The root of this function will give us the estimate of β. However, this may not always have a root if the data is noisy and estimate of c and/or r are not accurate. So we use the following method:

```
sizefun <- function(beta) {
    R0 <- beta/(beta-r)
    N <- (beta + r)/c
    (finalsize/N - (1 - exp(-R0 * finalsize/N)))^2
}
betavec <- seq(1.1 * r, 50*r, r/10)
sizevec <- sizefun(betavec)

if (all(diff(sizevec) < 0)) {
    beta <- betavec[head(which(sizevec < 1e-4), 1)]
} else {
    beta <- betavec[which.min(sizevec)]
}
```

Then, we get

```
inc.pars <- list()
(inc.pars <- within(prev.pars,{
    beta <- beta
    gamma <- beta - r
    N <- (beta + r)/c
    i0 <- iniI/beta/N
}))

## $i0
## [1] 1.678504e-07
##
## $N
## [1] 214793.5
##
## $beta
## [1] 9.166129
##
## $gamma
## [1] 8.965557

plot(times, count, log="y")
lines(times, SIR.detsim(times, unlist(inc.pars)[c(3, 4, 2, 1)], type="incidence"), col="red'
```



2 Sensitivity equations

Let $x_i(t, \theta)$ be the states of the SIR model and $\theta_{j,u}$ and $\theta_{j,c}$ be unconstrained and constrained parameters of the model. In order to employ gradient-based optimization algorithms (e.g. `BFGS`), we must solve for $dx_i(t, \theta)/d\theta_{j,u}$. To find the sensitivity equations, `fitsir` integrates the following set of differential equations along with the basic SIR model:

$$\begin{aligned}\frac{d}{dt}\frac{dx_i(t;\theta)}{d\theta_{j,u}} &= \left(\frac{d}{dt}\frac{dx_i(t;\theta)}{d\theta_{j,c}}\right)\frac{d\theta_{j,c}}{d\theta_{j,u}}\\ &= \left(\frac{\partial f_x}{\partial\theta_{j,c}} + \sum_i \frac{\partial f_x}{\partial x_i}\frac{dx_i(t;\theta)}{d\theta_{j,c}}\right)\frac{d\theta_{j,c}}{d\theta_{j,u}},\end{aligned}$$

where $f_x = dx/dt$. Essentially, we integrate sensitivity equations with respect to constrained parameters for simplicity but multiply $d\theta_{j,c}/d\theta_{j,u}$ after to obtain sensitiviy equations with respect to unconstrained parameters because optimization is done using unconstrained parameters.

For clarity, we write ν_{x_i,θ_j} to represent sensitivity equations with respect to constrained parameters. Then, we write

$$\nu_{x,\theta}(t;x;\theta) = \begin{bmatrix} \nu_{S,\beta} & \nu_{S,N} & \nu_{S,\gamma} & \nu_{S,i_0} \\ \nu_{I,\beta} & \nu_{I,N} & \nu_{I,\gamma} & \nu_{I,i_0} \end{bmatrix}.$$

So the sensitivity equations of the SIR model are given by

$$\frac{d}{dt}\nu_{x,\theta}(t;\cdot) = \begin{bmatrix} -\beta I/N & -\beta S/N \\ \beta I/N & \beta S/N - \gamma \end{bmatrix} \nu_{x,\theta}(t;\cdot) + \begin{bmatrix} -SI/N & \beta SI/N^2 & 0 & 0 \\ SI/N & -\beta SI/N^2 & -I & 0 \end{bmatrix}.$$

The following additional equations complete the sensitivity equations:

$$\nu_{x,\theta}(0;x(0);\theta) = \begin{bmatrix} 0 & 1-i_0 & 0 & -N \\ 0 & i_0 & 0 & N \end{bmatrix}$$
$$\left[\frac{d\theta_{j,c}}{d\theta_{j,u}}\right]_{j=1,2,3,4} = \begin{bmatrix} \beta & \gamma & N & (1-i_0)i_0 \end{bmatrix}.$$

Using the sensitivity equations of the SIR model, we can now compute the sensitivity equations of negative log-likelihood functions with respect to the unconstrained parameters by applying the chain rule. Given a log likelihood function $l(X;\mu;\theta)$, we have:

$$\frac{dl(X;\mu;\theta)}{d\theta_{j,u}} = \frac{dl}{d\mu}\frac{d\mu}{d\theta_{j,u}},$$

where X is the observed counts and μ is the expected trajectory. Precisely, μ is equal to I for prevalence fits, $S(\tau_n) - S(\tau_{n+1})$ for incidence fits, and $R(\tau_{n+1}) - R(\tau_n)$ for death fits.

3 Confidence intervals

`fitsir` provides three ways of estimating a cofidence interval around an estimated trajectory.

3.1 `delta` - Delta method

Delta method relies on the sensitivity equation. Borrowing the notation from the previous section, we write partial derivatives of an expected number of observations at time t with respect to unconstrained parameters as

$$\frac{d\mu}{d\theta_{j,u}}$$

We look at the partial derivative with respect to unconstrained parameters because fitting is done on the unconstrained scale and `mle2` returns a variance-covariance matrix of the unconstrained parameters.

Thus, we write

$$\nu_{\mu,\theta_u} = \begin{bmatrix} \nu_{\mu,\beta_u} & \nu_{\mu,N_u} & \nu_{\mu,\gamma_u} & \nu_{\mu,i_{o_u}}. \end{bmatrix}$$

Then, we find that

$$\mathrm{Var}(\mu) = \nu_{\mu,\theta_u} \cdot \mathrm{Cov}(\theta_u) \cdot \nu^T_{\mu,\theta_u}.$$

Taking the square root, we obtain the standard error of the estimated trajectory at each time step. Finally, confidence interval is calculated using the standard error by assuming normality.

3.2 `mvrnorm` - Multivariate normal approximation

It is assumed that the parameters are normally distributed. We take random samples of parameters based on this assumption using the variance-covariance matrix provided by `mle2`. For each sample, expected trajectory is calculated. Then, confidence interval at each time step is found by taking the appropriate quantile.

3.3 `wmvrnorm` - Weighted multivariate normal approximation

Like `wmvrnorm`, we start by taking random samples of parameters based on normal approximation. However, instead of taking the quantiles from the estimated trajectories, we take a weighted quantile instead with each trajectories weighted by likelihood of a trajectory divided by probability of the parameter sample that generated the trajectory.

The reason why we do this is based on the Metropolis-Hastings rule: a Markov chain over values of the parameter vecotr x_i will converge to the correct distribution if for any t

$$\frac{P(\text{jump} x_t \to x_{t+1}) P(\text{accept} x_{t+1}|x_t)}{P(\text{jump} x_{t+1} \to x_t) P(\text{accept} x_t|x_{t+1})} = \frac{\mathrm{Post}(x_{t+1})}{\mathrm{Post}(x_t)} = \frac{\mathcal{L}(x_{t+1})\mathrm{Prior}(x_{t+1})}{\mathcal{L}(x_t)\mathrm{Prior}(x_t)}.$$

When we are taking random samples from a multivariate normal distribution, the probability of sampling x_{t+1} is independent of x_t. Furthermore, we're assuming that the prior probabilities are all equal. Therefore, we get

$$\frac{P(\text{sample} x_{t+1}) P(\text{accept} x_{t+1})}{P(\text{sample} x_t) P(\text{accept} x_t)} = \frac{\mathcal{L}(x_{t+1})}{\mathcal{L}(x_t)}.$$

Therefore, letting

$$P(\text{accept} x_{t+1}) = \frac{\mathcal{L}(x_{t+1})}{P(\text{sample} x_{t+1}}$$

satisfies the criterion.