M C M C

Computer Intensive Statistics in Ecology

Department of Life Science
Yu-Jen Lin 林妤真
B04B01036

HW 14

M C M C

Estimate the posterior distributions of parameters using the method Markov chain Monte Carlo (MCMC).

sigma=0.032

Prior distributions

Linf~U[40,100]

K~U[0.1,0.6]

HW14

MCMC

1

Data Input

2

Likelihood

3

MCMC

4

Results

HW14

1

Data Input

female Pacific Hake

Age = c(1,2,3.3,4.3,5.3,6.3,7.3,8.3,9.3,10.3,11.3,12.3,13.3)

Length = c(15.4,28.03,41.18,46.2,48.23,50.26,51.82,54.27,56.98,58.93,59,60.91,61.83)

HW14

2

Likelihood

Template

population dynamics model

```
pop_model <- function(r,N0){
        N = NULL
        Dev2 = NULL
        N[1] = N0
        for(t in 2:33){
                N[t] = N[t-1]*(1+r)
                Dev2[t]=(log(obs_N[t]/N[t]))^2
         }
```

HW14

von Bertalanffy Growth Function

```
VBGF <-function(x, Linf, K){
   y = Linf * (1 - exp(- K * (x - 0)))
   return(y)
}
```

$$L(t) = L_{\infty}\left(1-\exp\left(-K\left(t-t_0\right)\right)\right)e^{\varepsilon} \quad \varepsilon \sim N(0,\sigma^2)$$

```
Like_1 = 1/obs_N[9]*(1/sqrt(2*pi*sigma[9]*sigma[9]))*exp(-
Dev2[9]/(2*sigma[9]*sigma[9]))
 Like_2 = 1/obs_N[16]*(1/sqrt(2*pi*sigma[16]
*sigma[16]))*exp(-Dev2[16]/(2*sigma[16]*sigma[16]))
 Like_3 = 1/obs_N[26]*(1/sqrt(2*pi*sigma[26]
*sigma[26]))*exp(-Dev2[26]/(2*sigma[26]*sigma[26]))
 neg_likelihood =  -(log(Like_1)+log(Like_2)+log(Like_3))

 if (is.na(neg_likelihood)) neg_likelihood = 100000
```

```
totoal_like = exp(-neg_likelihood)

Outs <- NULL
Outs$totoal_like <- totoal_like
Outs$neg_likelihood <- neg_likelihood
Outs$r <- r
Outs$N0 <- N0
Outs$N_lastyr <- N[33]

return(Outs)
}
```

Template

```
lognormal_like = function(Linf, K){

 Like = numeric(length(Length))
 neg_likelihood = numeric(length(Length))
 ypred = VBGF(Age, Linf, K)
 Dev2 =(log(Length) - log(ypred)) ^ 2
 sigma = 0.032
 for (i in 1:length(Length)){
   Like[i] = (1 / (Age[i] * sqrt(2 * pi) * sigma)) * exp(-Dev2[i] / (2 * sigma
^ 2))
   neg_likelihood[i] = -log(Like[i])
 }
```

```
 totoal_Like = exp(-sum(neg_likelihood))

 Outs <- NULL
 Outs$totoal_Like <- totoal_Like
 Outs$neg_likelihood <- sum(neg_likelihood)
 Outs$ypred <- ypred
 Outs$Linf <- Linf
 Outs$K <- K

 return(Outs)
}
```

HW14

HW14

3

MCMC

DoMCMC ← Function

for

simulation

jump function

if

evaluate the ratio

if

thinning

if

burn in

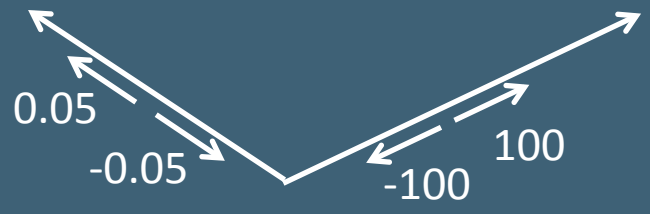
DoMCMC ←

```
function(Xinit,Ndim,Nsim=1000,Nburn=0,Nthin=1)

{                          # Step size
 Linf_jump_max =    100
 Linf_jump_min =  - 100

   K_jump_max =     0.05
   K_jump_min =   - 0.05
```

MLE results

Linf = 61.09

K = 0.30



```
Xcurr <- Xinit
Fcurr <- -1* lognormal_like (r=Xcurr[1],N0=Xcurr[2])$neg_likelihood
Outs2 <- matrix(-9999,
                nrow=(Nsim-Nburn),
                ncol=(Ndim+1))
Ipnt <- 0; Icnt <- 0
```

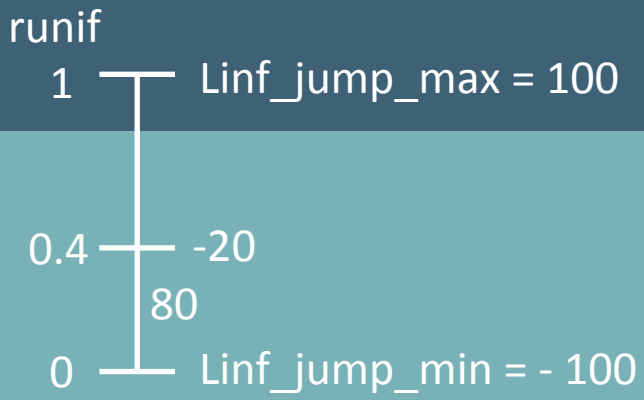
```
for (Isim in 1:Nsim)
{
  # jump function
    Xnext = NULL


    Xnext[1] =    Xcurr[1]
                  + runif(1, 0, 1) * (Linf_jump_max - Linf_jump_min)
                  + Linf_jump_min


    Xnext[2] =    Xcurr[2]
                  + runif(1, 0, 1) * (K_jump_max - K_jump_min)
                  + K_jump_min
```

for (Isim in 1:Nsim)

{

jump function

Xnext = NULL



Xnext[1] = Xcurr[1]
+ runif(1, 0, 1) * (Linf_jump_max - Linf_jump_min)
+ Linf_jump_min

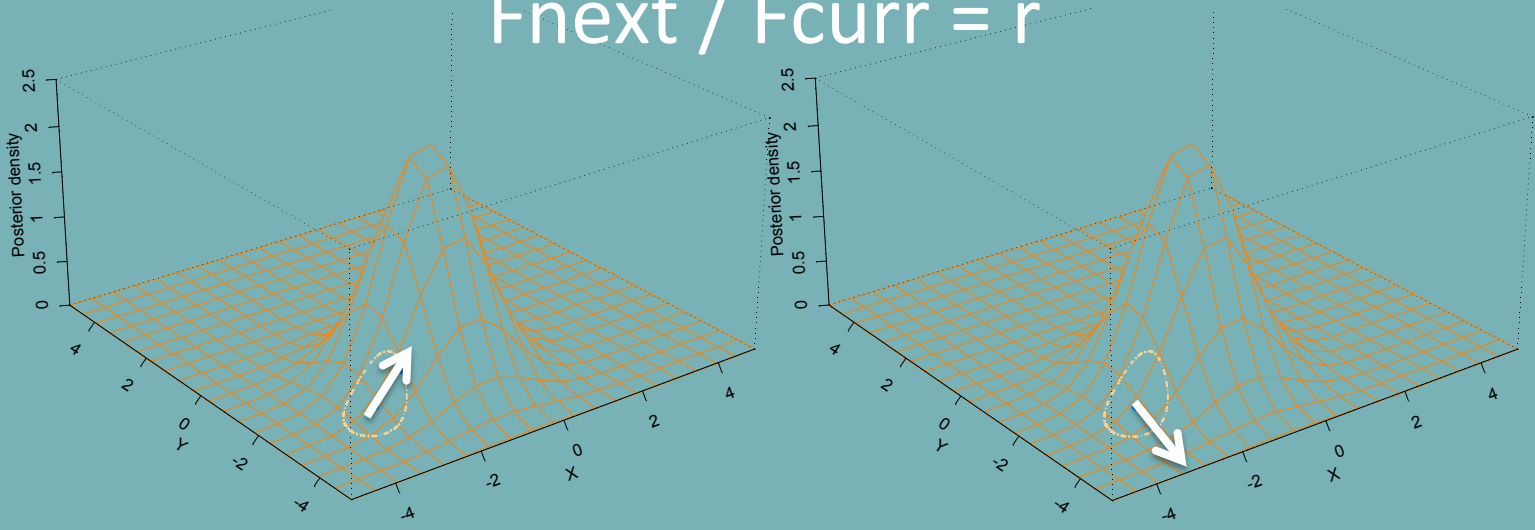
Xnext[2] = Xcurr[2]
+ runif(1, 0, 1) * (K_jump_max - K_jump_min)
+ K_jump_min

```
for (Isim in 1:Nsim)
{
    # jump function
    Xnext = NULL

            # Linf prior: Linf ~ U [40, 100]
     Xnext[1] =     Xcurr[1]
                    + runif(1, 0, 1) * (Linf_jump_max - Linf_jump_min)
                    + Linf_jump_min

            # K prior: K ~ U [0.1, 0.6]
     Xnext[2] =     Xcurr[2]
                    + runif(1, 0, 1) * (K_jump_max - K_jump_min)
                    + K_jump_min
```

```
for (Isim in 1:Nsim)
{
  # jump function
    Xnext = NULL

  repeat{ # Linf prior: Linf ~ U [40, 100]
    Xnext[1] =      Xcurr[1]
                    + runif(1, 0, 1) * (Linf_jump_max - Linf_jump_min)
                    + Linf_jump_min
    if(100>=Xnext[1] && Xnext[1]>=40){
      break}
  }

  repeat{ # K prior: K ~ U [0.1, 0.6]
    Xnext[2] =      Xcurr[2]
                    + runif(1, 0, 1) * (K_jump_max - K_jump_min)
                    + K_jump_min
    if(0.6>=Xnext[2] && Xnext[2]>=0.1){
      break}
  }
```

```
# evaluate the ratio
 Fnext <- -1 * lognormal_like(Linf = Xnext[1], K = Xnext[2])$neg_likelihood
 Rand1 <- log(runif(1,0,1))
 if (Fnext > Fcurr+Rand1)
       {Fcurr <- Fnext; Xcurr <- Xnext}
```

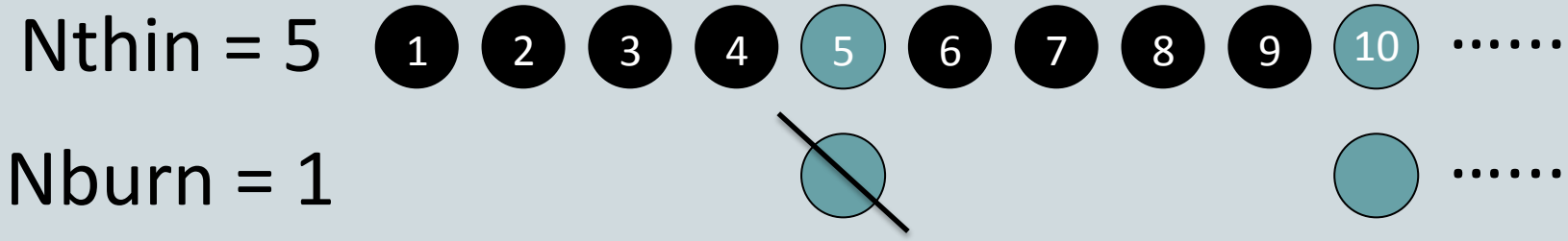
Fnext / Fcurr = r



r =>1
Accept the new point

r <1
Accept the new point with probability r

```
    if (Isim %% Nthin == 0)     # thinning
     {   Ipnt <- Ipnt + 1

         if (Ipnt > Nburn)     # burn in
           {  Icnt <- Icnt + 1
              Outs2[Icnt, ] <- c(Xcurr, Fcurr)
           }
     }
   }
 return(Outs2[1:Icnt,])
}
```

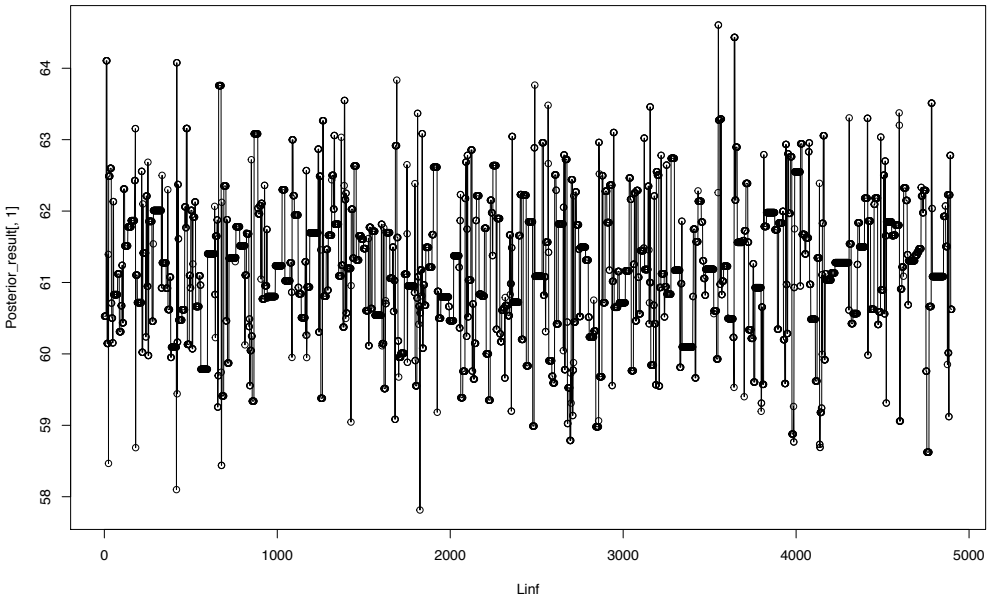


4

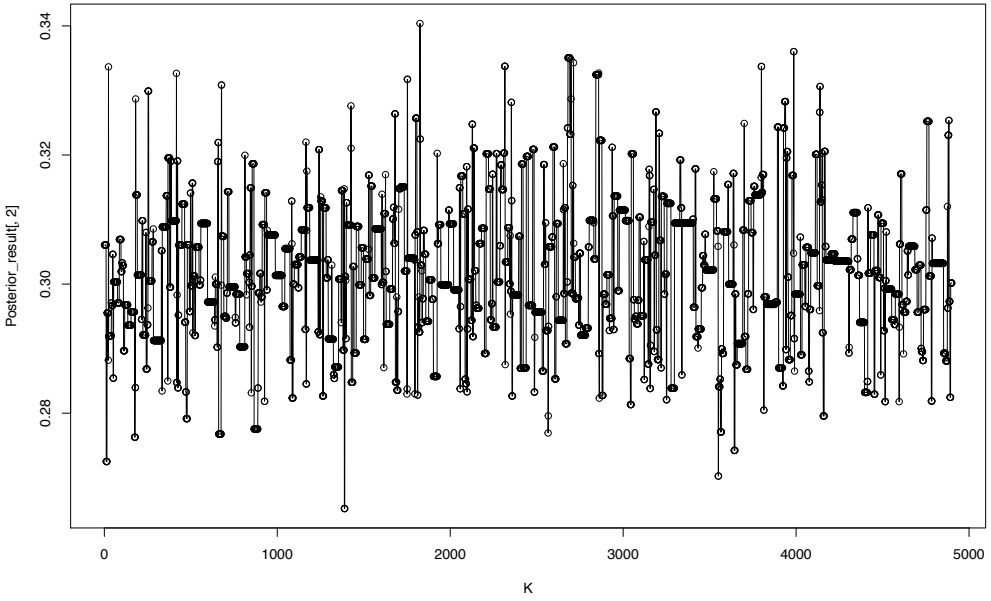
Results

```
Posterior_result <- DoMCMC( Xinit = c(100, 0.1),
                                    Ndim = 2,
                                    Nsim = 50000,
                                    Nburn = 100,
                                    Nthin = 10)

plot(Posterior_result[,1],main="",xlab="Linf",type="o")


hist(Posterior_result[,1], main="”, xlab=“Linf”, col="gray",
    breaks = seq(min(Posterior_result[,1]), max(Posterior_result[,1]),
                 length.out = 1000))
abline(v=median(Posterior_result[,1]), col ="red”)
```

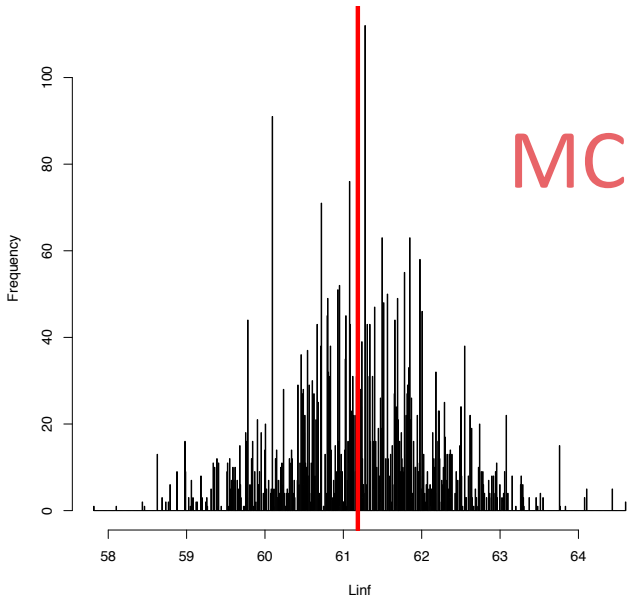
Linf


Posterior_result[, 1]
58
59
60
61
62
63
64
0
1000
2000
3000
4000
5000
Linf

K


Posterior_result[, 2]
0.28
0.30
0.32
0.34
0
1000
2000
3000
4000
5000
K

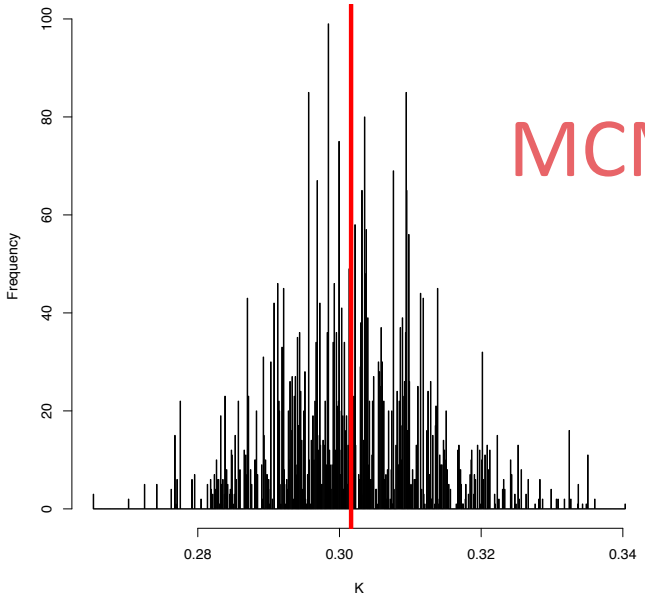
Linf



MCMC Medium = 61.13

MLE results = 61.09

K



MCMC Medium = 0.30

MLE results = 0.30

Thank you