Homework 9

Jing Leng (GSI: Jiahe)

November 30, 2014

1

a

$$L(\lambda) = \prod P(y_i|\lambda, x_i) = \prod \frac{(\lambda x_i)^{y_i} e^{-\lambda x_i}}{y_i!}$$

$$l(\lambda) = \sum (y_i \log \lambda + y_i \log x_i - \lambda x_i - \log(y_i!))$$

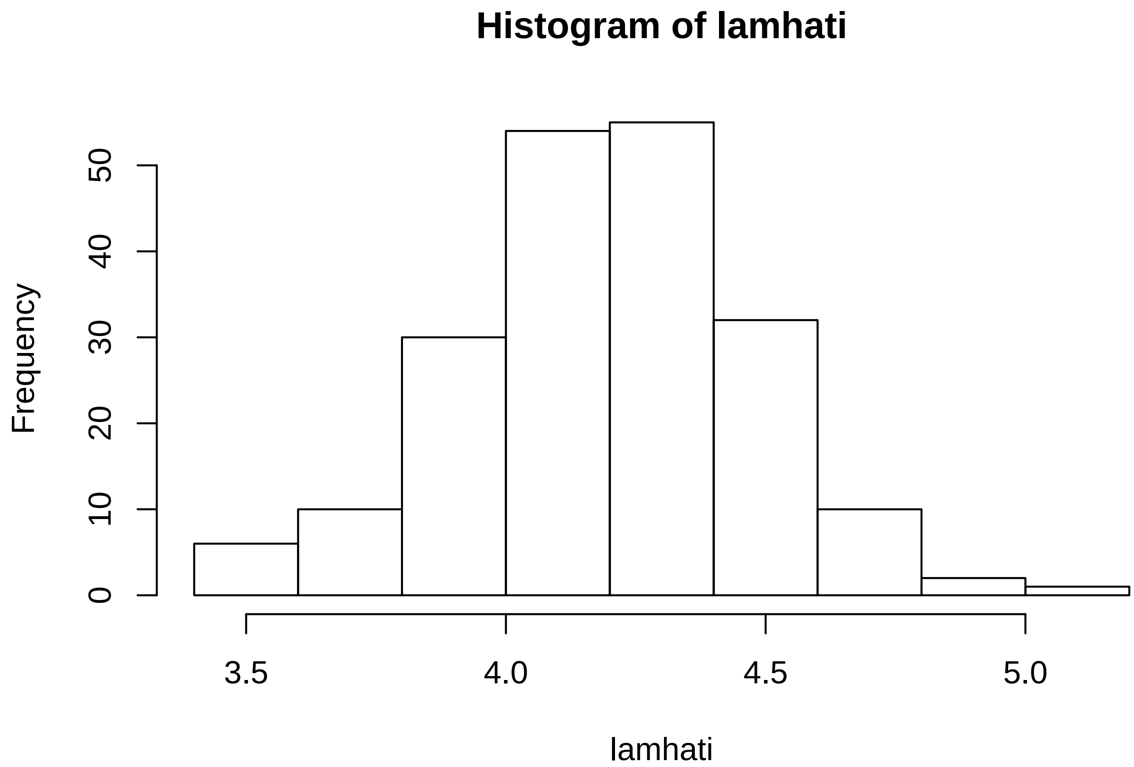
$$\hat{\lambda} = \frac{\sum y_i}{\sum x_i}$$

```
airline <- read.table('airlines.dta', head = F)
airline$V3 <- airline$V3/1000
lamhat <- sum(airline[2])/sum(airline[3])
B = 200
n = nrow(airline)
lamhati <- numeric(B)
for ( i in 1:B) {
  bs <- rpois(n, lamhat * airline[[3]])
  lamhati[i] <- sum(bs)/sum(airline[3])
}
bias <- 1/B*sum(lamhati) - lamhat
mse <- 1/B* sum((lamhati - lamhat)^2)
bias;mse
```

```
## [1] 0.01452
```

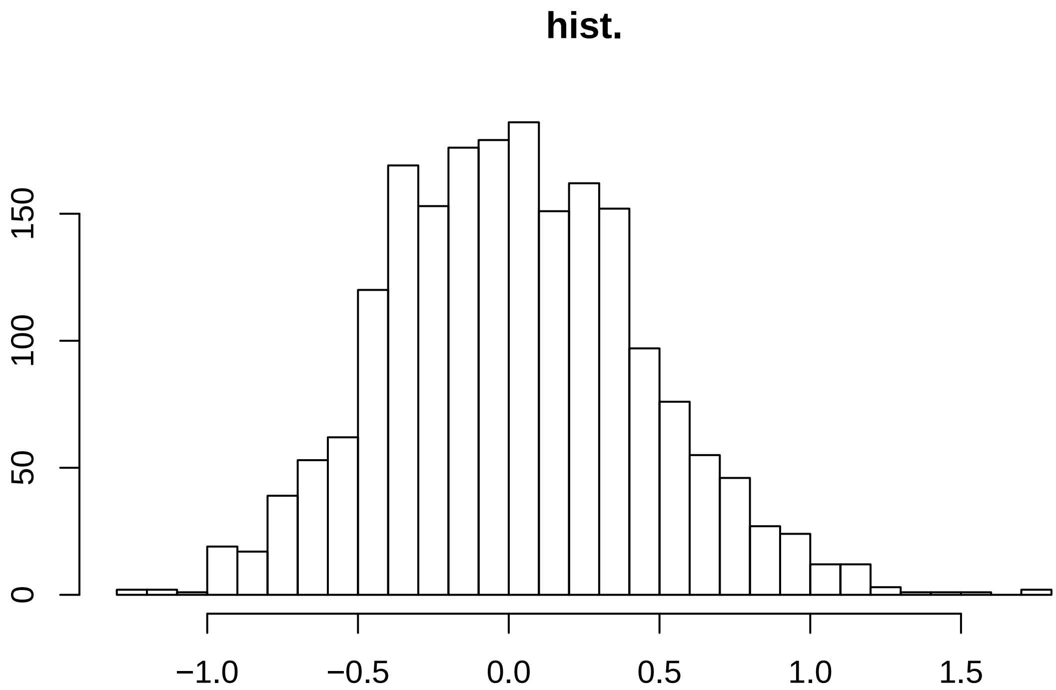
```
## [1] 0.07563
```

```
hist(lamhati)
```



b

```
B=2000
lamhati=numeric(B)
for (i in 1:B){
  U=1+n*runif(n,0,1); S=airline[floor(U), ]
  lamhati[i] <- sum(S[2])/sum(S[3])
}
hist(lamhati - lamhat,30,main='hist.',xlab='',ylab='')
```




```r
mean(lamhati - lamhat)
```

```
## [1] 0.02123
```

```r
mse <- mean((lamhati - lamhat)^2)
bias; mse
```

```
## [1] 0.01452
```

```
## [1] 0.1834
```

c

$$\pi(\lambda|\underline{y}) \propto \prod(\frac{e^{-\lambda x_i}(\lambda x_i)^{y_i}}{y_i!}) \times \frac{\beta_0^{\alpha_0}}{\Gamma(\alpha_0)}\lambda^{\alpha_0-1}e^{-\beta_0\lambda}$$

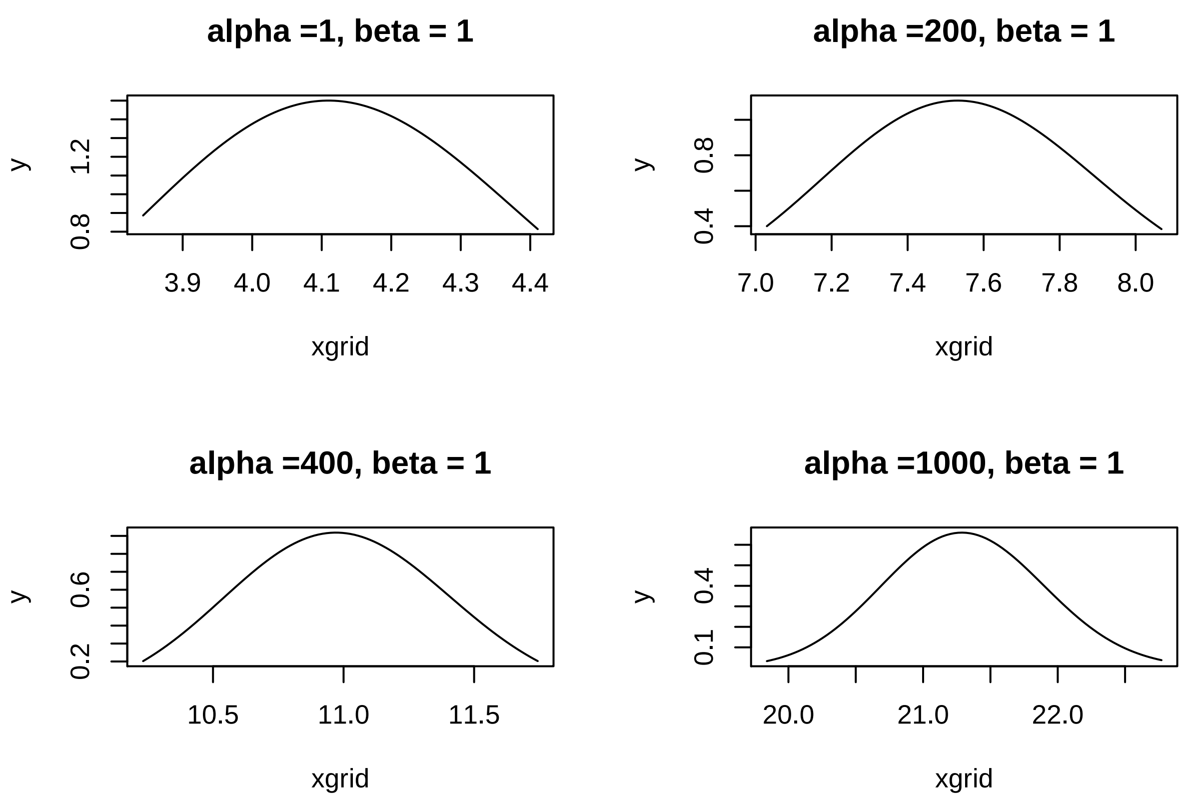
$$\beta^* = \beta_0 + \sum x_i$$

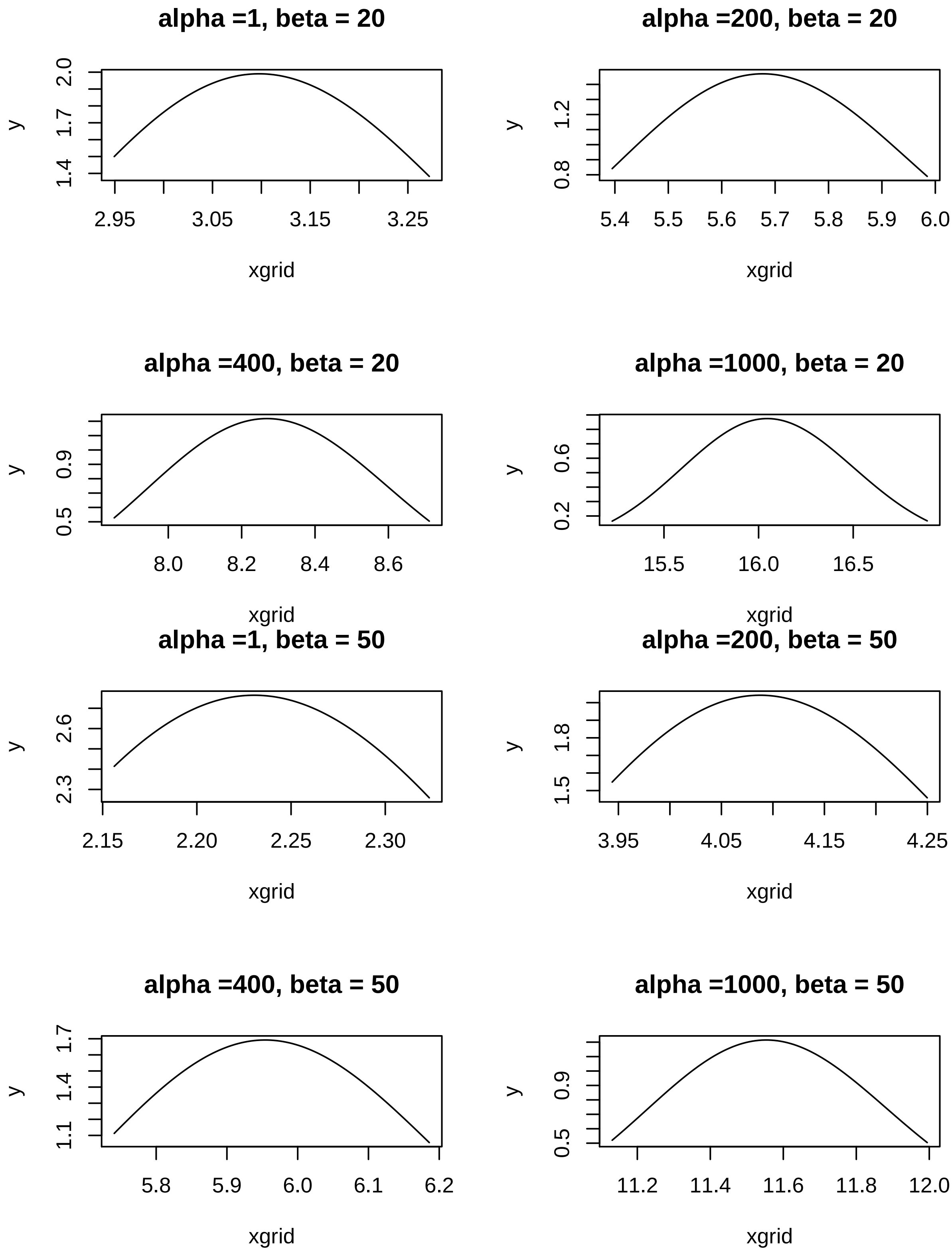
$$\alpha^* = \alpha_0 + \sum y_i$$

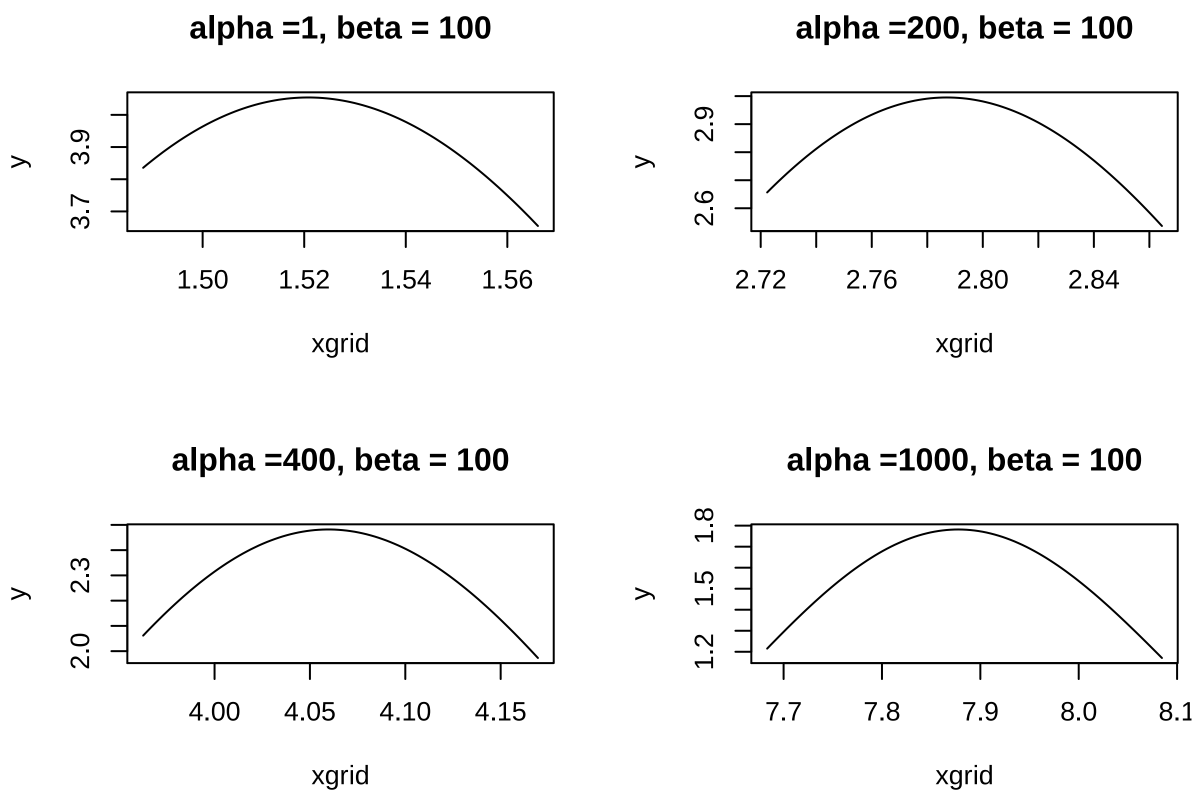
The posterior distribution of λ is $Gamma(\alpha_0 + \sum y_i, \beta_0 + \sum x_i)$ Posterior mean: α^*/β^* Posterior variance: $\alpha^*/(\beta^*)^2$

```r
B = 2000
lamhati = numeric(B)


trial <- function(a, b) {
beta0 <- b
alpha0 <- a
```

```
betastar <- beta0 + sum(airline[3])
alphastar <- alpha0 + sum(airline[2])
alphastar; betastar

mean <- alphastar/betastar
var <- alphastar/(betastar^2)
mean;var

xgrid <- seq(mean - 4*var, mean + 4*var, length.out = 100)
y <- dgamma(xgrid, shape = alphastar, scale = 1/betastar)
plot(xgrid, y, 'l', main = paste0("alpha =", a, ", beta = ", b))
return (c(a, b, mean, var))
}

a <- c(1, 200, 400, 1000)
b <- c(1, 20, 50, 100)
abgrid <- expand.grid(a, b)
par(mfrow = c(2, 2))
miumiu <- sapply(1:16, function(i) {trial(abgrid[i,1], abgrid[i,2])})
```




alpha =1, beta = 20
y
1.4
1.7
2.0
2.95
3.05
3.15
3.25
xgrid
alpha =200, beta = 20
y
0.8
1.2
5.4
5.5
5.6
5.7
5.8
5.9
6.0
xgrid
alpha =400, beta = 20
y
0.5
0.9
8.0
8.2
8.4
8.6
xgrid
alpha =1000, beta = 20
y
0.2
0.6
15.5
16.0
16.5
xgrid
alpha =1, beta = 50
y
2.3
2.6
2.15
2.20
2.25
2.30
xgrid
alpha =200, beta = 50
y
1.5
1.8
3.95
4.05
4.15
4.25
xgrid
alpha =400, beta = 50
y
1.1
1.4
1.7
5.8
5.9
6.0
6.1
6.2
xgrid
alpha =1000, beta = 50
y
0.5
0.9
11.2
11.4
11.6
11.8
12.0
xgrid




```
table <- data.frame(alpha = miumiu[1,], beta = miumiu[2, ], mean = miumiu[3, ], var = miumiu[4,])
table
```

```
##    alpha beta   mean      var
## 1      1    1  4.127 0.070964
## 2    200    1  7.549 0.129805
## 3    400    1 10.988 0.188941
## 4   1000    1 21.305 0.366351
## 5      1   20  3.111 0.040317
## 6    200   20  5.690 0.073746
## 7    400   20  8.282 0.107343
## 8   1000   20 16.059 0.208134
## 9      1   50  2.240 0.020902
## 10   200   50  4.097 0.038233
## 11   400   50  5.963 0.055651
## 12  1000   50 11.563 0.107906
## 13     1  100  1.527 0.009718
## 14   200  100  2.793 0.017775
## 15   400  100  4.066 0.025873
## 16  1000  100  7.884 0.050167
```

d

$$p(\lambda|\underline{y}) = \frac{1}{C}\lambda^{\alpha^*-1}e^{-\beta^*\lambda}$$

Importance function:

$$\omega(\lambda) = \lambda^{\alpha^*-1}e^{-(\beta^*-1)\lambda}$$

Sample λ from exponential distribution, then use the importance to calculate weighted mean and variance.

```r
func <- function(a, b) {
  beta0 <- b
  alpha0 <- a
  betastar <- beta0 + sum(airline[3])
  alphastar <- alpha0 + sum(airline[2])
  lami <- rexp(n, 40)
  weight <- lami^(alphastar-1)*exp(-(betastar-1)*lami)
  sum(weight)
  mean <- mean(lami*weight/sum(weight))
  var <- var(lami*weight/sum(weight))
  return (c(a, b, mean, var))
}

miumiu <- sapply(1:16, function(i) {func(abgrid[i,1], abgrid[i,2])})
table <- data.frame(alpha = miumiu[1,], beta = miumiu[2, ], mean = miumiu[3, ], var = miumiu[4,])
table
```

```
##    alpha beta     mean       var
## 1      1    1 0.004853 0.0002355
## 2    200    1      NaN        NA
## 3    400    1      NaN        NA
## 4   1000    1      NaN        NA
## 5      1   20 0.006034 0.0003641
## 6    200   20      NaN        NA
## 7    400   20      NaN        NA
## 8   1000   20      NaN        NA
## 9      1   50      NaN        NA
## 10   200   50      NaN        NA
## 11   400   50      NaN        NA
## 12  1000   50      NaN        NA
## 13     1  100 0.007166 0.0005135
## 14   200  100      NaN        NA
## 15   400  100      NaN        NA
## 16  1000  100      NaN        NA
```

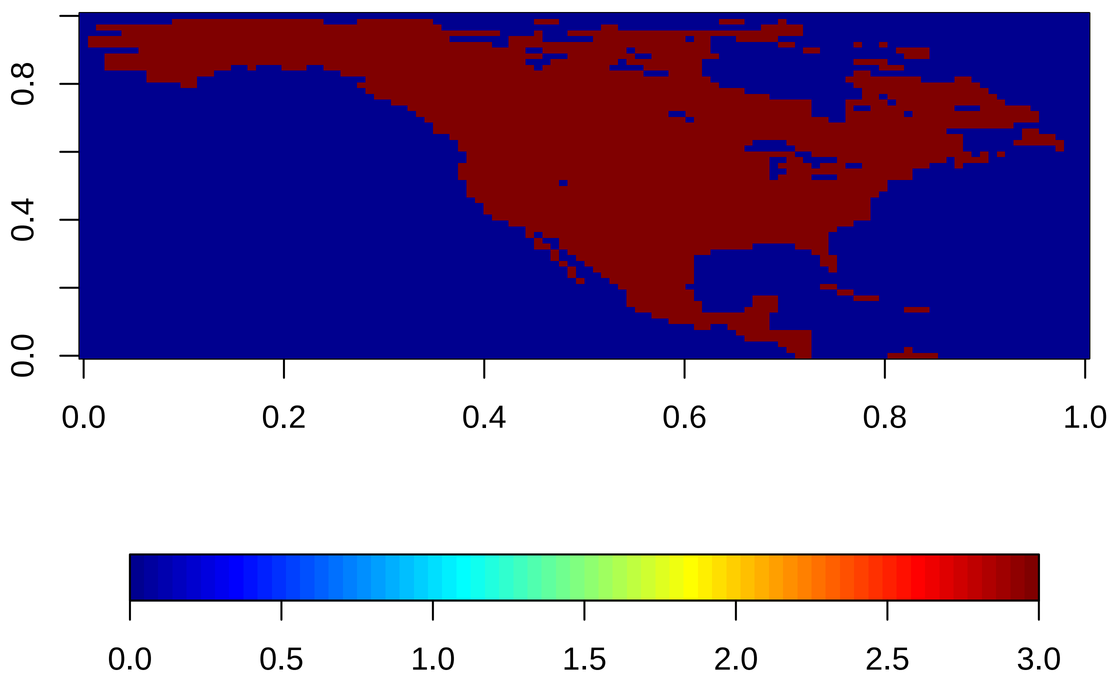
2

```r
LAI = read.table("LAI.csv", sep = ",")
LAI = as.matrix(LAI)

T = dim(LAI)[1]
Tlist = 1:T
LAI = sapply(Tlist, function(i) matrix(LAI[i,], nrow=120, ncol=60, byrow=T),
             simplify="array")
library(fields)
```

```
## Loading required package: spam
## Loading required package: grid
## Spam version 1.0-1 (2014-09-09) is loaded.
## Type 'help( Spam)' or 'demo( spam)' for a short introduction
## and overview of this package.
```

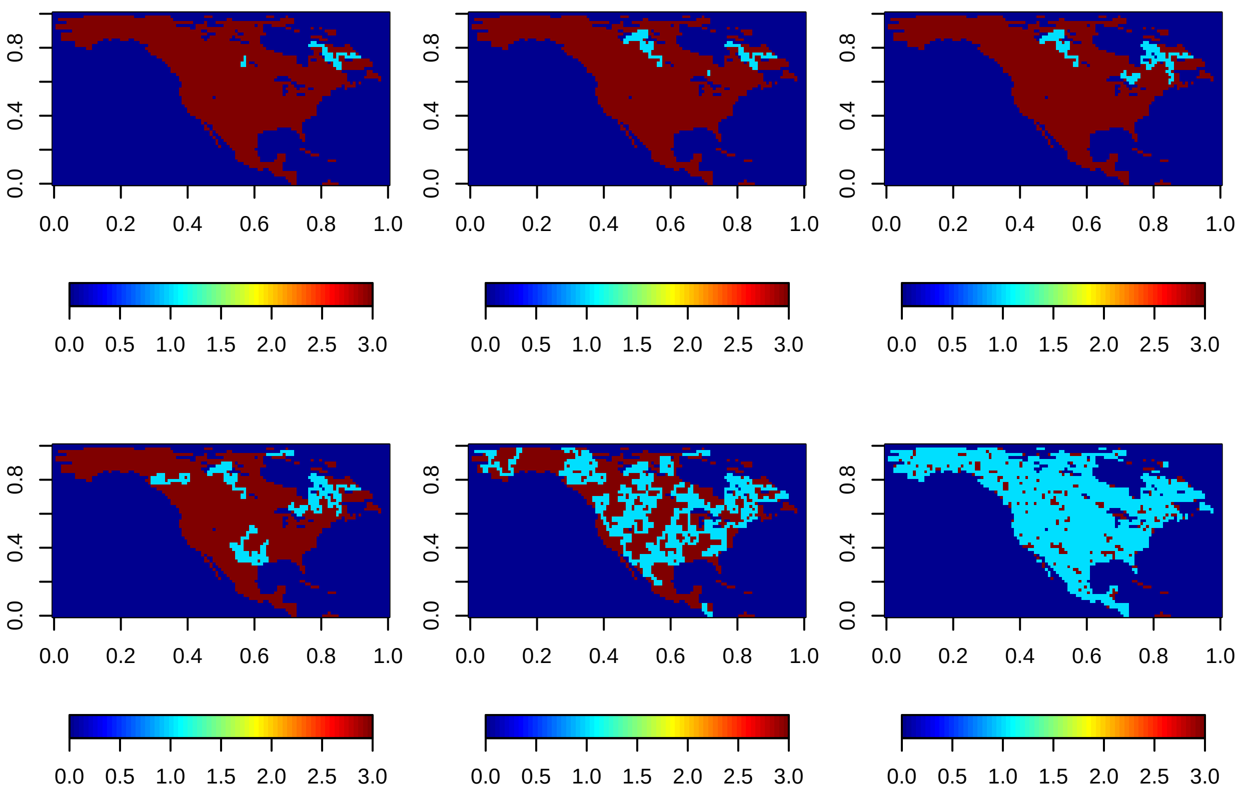
```
## Help for individual functions is also obtained by adding the
## suffix '.spam' to the function name, e.g. 'help( chol.spam)'.
##
## Attaching package: 'spam'
##
## The following objects are masked from 'package:base':
##
##     backsolve, forwardsolve
##
## Loading required package: maps
```

```
L = (LAI[,,7] > 0) * 3
image.plot(L, horizontal = T)
```




```
pool <- which(L > 0)
T = 0
# begein of loop
par(mfrow = c(2,3))
repeat {
  x <- sample(pool, 1)

  repeat {
    if (L[x] == 3) L[x] = 1
    pool <- pool[pool != x]
    if (length(pool) == 0) break
    if (x <= 120 | x == 1) {
      possible <- c(1, -1, 120) +x
    }else possible <- c(-1, 1, 120, -120) +x

    if (sum(L[possible] == 3) != 0)  {# find new point {
      newx <- ifelse(sum(L[possible] == 3) == 1, possible[L[possible] == 3], sample(possible[L[possible]
    } else {
```

```
      break
    }

    x = newx
    T = T + 1
    if (sum(T == c(50, 100, 150, 300, 1000, 2000)) == 1) {

      image.plot(L, horizontal = T)

    }

  }
    if (T >= 3000)break
  if (length(pool) == 0) break
}
```




```
image.plot(L, horizontal = T)
```