第七次作业

PB18000029 舒文炫

目录

6.4 题

(1)

解: 柯西分布的密度函数为

$$f(x|\mu,\lambda)=\frac{\lambda}{\pi[\lambda^2+(x-\mu)^2]}$$

这里提议分布 $N(X_t,\sigma^2)$ 是对称的，所以有

$$\alpha(x_t,y)=min\left\{1,\frac{f(y)}{f(x_t)}\right\}=min\left\{1,\frac{\lambda^2+(x_t-\mu)^2}{\lambda^2+(y-\mu)^2}\right\}$$

下面是随机游动 Metroplis 算法的 R 代码 (基本和书上代码一样), 并且给出代入标准差为 0.05，0.5，2.5，16 之后的拒绝概率和轨迹图

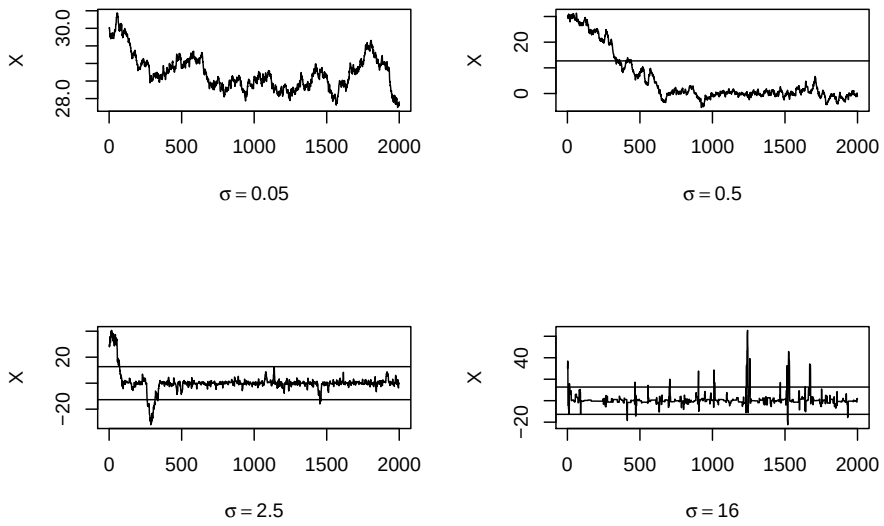
```
set.seed(3)
rwMetro<-function(lambda,mu,sigma,x0,N){
  #lambda,mu 为柯西分布的两个参数
  #sigma 为提议分布标准差
  #x0 为初始值
```

```
  #N 为链跑的次数
  x<-numeric(N)## 生成一个长为 N 的数组，且每个元素初始化为 0
  x[1]<-x0
  u<-runif(N)## 创建 (0,1) 均匀分布随机偏差，因为跑了 N 次，所以需要 N 个这样的偏差
  k<-0##k 为最终拒绝的个数
  for(i in 2:N){## 每次循环与接受概率比较，大就接受，小就舍去
    y<-rnorm(1,x[i-1],sigma)
      if(u[i]<=(lambda^2+(x[i-1]-mu)^2)/(lambda^2+(y-mu)^2))
        x[i]<-y
      else{
        x[i]=x[i-1]
        k=k+1
      }
  }
  return(list(x=x,k=k))
}
# 下面是代入数据部分
lambda<-1
mu<-0
N<-2000
sigma<-c(0.05,0.5,2.5,16)
x0<-30
cauchy1<-rwMetro(lambda,mu,sigma[1],x0,N)
cauchy2<-rwMetro(lambda,mu,sigma[2],x0,N)
cauchy3<-rwMetro(lambda,mu,sigma[3],x0,N)
cauchy4<-rwMetro(lambda,mu,sigma[4],x0,N)
# 下面计算拒绝概率
print(c(cauchy1$k,cauchy2$k,cauchy3$k,cauchy4$k)/N)
```

```
## [1] 0.0025 0.0955 0.4450 0.8100
```

可以看到随着 σ 的增大, 拒绝的概率也在增大，下面我给出四条链的轨迹图

```
sigma=c(0.05,0.5,2.5,16)
par(mfrow=c(2,2))## 方便比较四个图以 2*2 的形式排列
refline<-qcauchy(c(0.025,0.975))## 标准柯西分布的参考线
cauchyline<-cbind(cauchy1$x,cauchy2$x,cauchy3$x,cauchy4$x)
for(j in 1:4){
  plot(cauchyline[,j],type="l",xlab=bquote(sigma==.(round(sigma[j],3))),ylab="X",ylim=r
  abline(h=refline)
}
```

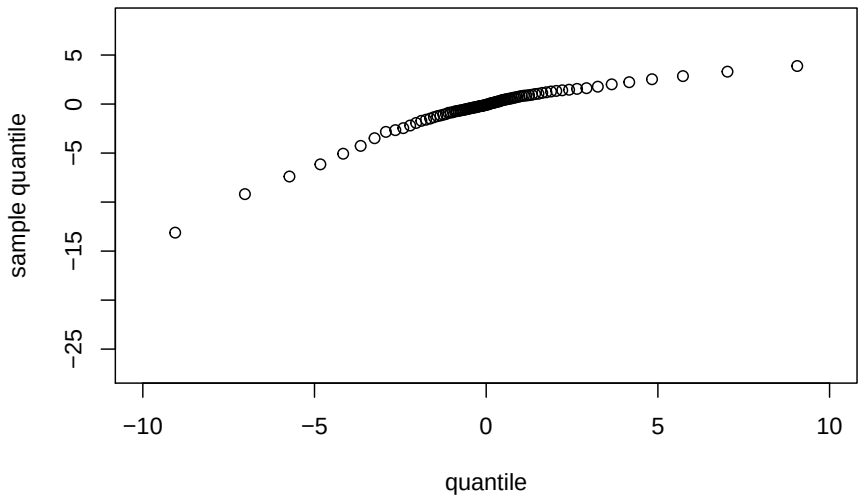



```
par(mfrow=c(1,1))
```

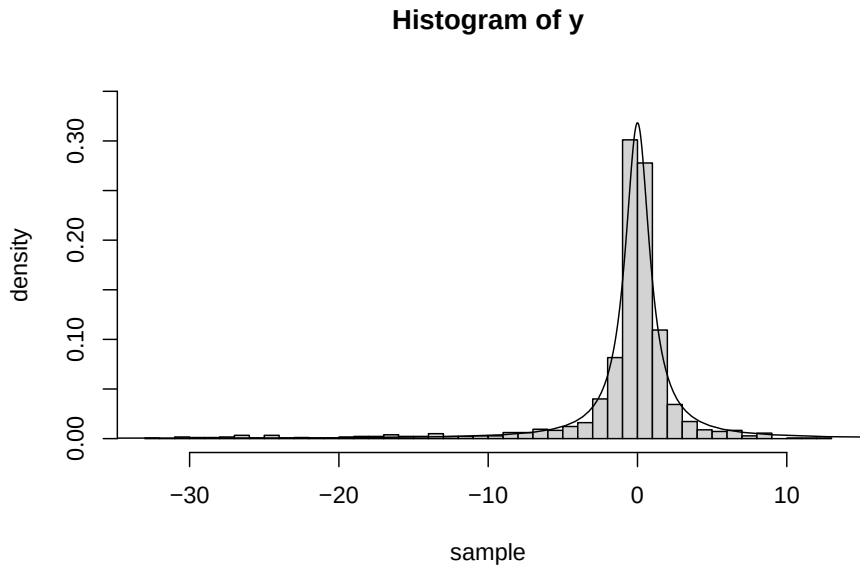
可以看到 $\sigma = 0.05$ 时，链甚至还没有进入到分布的参考线内，此时离收敛还很远，$\sigma = 0.5$ 时，可以收敛，但是很慢，$\sigma = 2.5$ 时很快就进入了参考线内，可以认为收敛了，$\sigma = 16$ 时接受概率小，虽然看出来收敛，但是效率低，而且这中间还有明显的毛刺，可以认为收敛性最好的是 $\sigma = 2.5$ 时的那条链，下面我们画 QQ 图和直方图

```
N=2000
b<-201## 丢弃掉前 200 个点
```

```
y<-cauchy3$x[b:N]
a<-ppoints(100)##100 个分位点
cauchyref<-qcauchy(a)## 柯西分布分位点参考值
mycauchy<-quantile(y,a)## 我得到的链对应的分位点
qqplot(cauchyref,mycauchy,xlab="quantile",ylab="sample quantile",xlim=c(-10,10))
```




```
hist(y,breaks = "scott",xlab="sample",ylab="density",ylim=c(0,0.35),freq=FALSE)
lines(cauchyref,dcauchy(cauchyref))
```



可以看到 qq 图近似一条直线，且直方图来看也是符合这个趋势的

(2)

这里丢掉前 1000 个值，用列表表示分位数如下, 一般比较分位数不会去找 0 和 1 对应的，我们会取 0.05 和 0.95

```
N<-2000
b<-1001## 丢掉前 1000 个
a<-c(0.05,seq(0.1,0.9,0.1),0.95)## 各个十分位数
cauchyref<-qcauchy(a)## 参考十分位数
discauchyline<-cauchyline[b:N,]## 丢弃前 1000 个后的链
mycauchy1<-quantile(discauchyline[,1],a)## 我的分位数，每一列分别算出分位数然后拼接起来
mycauchy2<-quantile(discauchyline[,2],a)
mycauchy3<-quantile(discauchyline[,3],a)
mycauchy4<-quantile(discauchyline[,4],a)
mycauchy<-cbind(mycauchy1,mycauchy2)
mycauchy<-cbind(mycauchy,mycauchy3)
mycauchy<-cbind(mycauchy,mycauchy4)
```

```
print(round(cbind(cauchyref,mycauchy),3))## 结果保留三位小数打印出来
```

```
##      cauchyref mycauchy1 mycauchy2 mycauchy3 mycauchy4
## 5%      -6.314    27.990    -2.505    -3.625    -3.872
## 10%     -3.078    28.088    -1.733    -2.205    -2.377
## 20%     -1.376    28.227    -0.967    -0.975    -1.376
## 30%     -0.727    28.325    -0.642    -0.529    -0.704
## 40%     -0.325    28.409    -0.373    -0.232    -0.198
## 50%      0.000    28.519    -0.081     0.106     0.019
## 60%      0.325    28.639     0.171     0.401     0.471
## 70%      0.727    28.760     0.503     0.711     1.047
## 80%      1.376    28.906     0.908     1.110     1.655
## 90%      3.078    29.108     1.631     2.101     3.645
## 95%      6.314    29.380     2.356     3.617    10.837
```

通过比较发现十分位数拟合最好的链是 $\sigma = 2.5$ 对应的那条链

(3)

代码已经在前两题给出

6.8 题

(1)

这里题目说要独立抽样，但是提议分布又和 X_t 有关，这很奇怪，这里我考虑把 X_t 换成 $0.8 \times 5 = 4$, 下面结果按这个来。此时

$$\alpha(X_t, Y) = min\left\{1, \frac{f(Y)g(X_t)}{f(X_t)g(Y)}\right\}$$

下面写出 R 代码

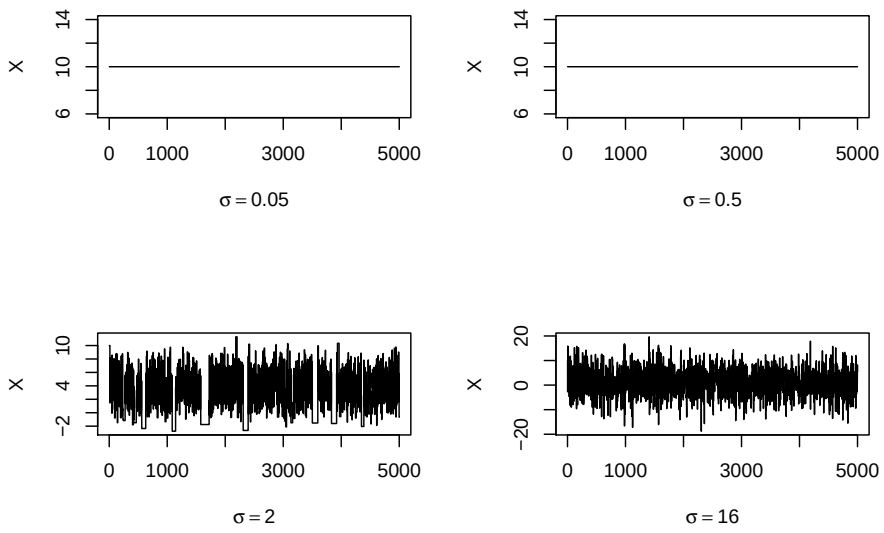
```
indepmetro<-function(mu,sigma,m,p,samsigma,sammu){
  # sigma,mu 为提议正态分布的参数
  # m 为链的长度
  # p,samsigma,sammu 为要抽样的分布的参数
  xt<-numeric(m)
  u<-runif(m)
  ## 对提议分布抽样
  y<-rnorm(m,mu,sigma)
  k<-0
  xt[1]=10
  for(i in 2:m){
    fy<-p*dnorm(y[i],sammu[1],samsigma[1])+(1-p)*dnorm(y[i],sammu[2],samsigma[2])
    fx<-p*dnorm(xt[i-1],sammu[1],samsigma[1])+(1-p)*dnorm(xt[i-1],sammu[2],samsigma[2])
    r<-fy/fx*dnorm(xt[i-1],mu,sigma)/dnorm(y[i],mu,sigma)
    if(u[i]<=r)
      xt[i]<-y[i]
    else{
      xt[i]<-xt[i-1]
      k<-k+1
      }

    }

  return(list(x=xt,k=k))
}
mu<-4
sigma<-c(0.05,0.5,2,16)## 这里我从小到大取了一些值
m<-5000
p<-0.2
sammu<-c(0,5)
samsigma<-c(1,1)
hynorm<-numeric(4)
for(i in 1:4){
```

```
  hynorm[i]<-indepmetro(mu,sigma[i],m,p,sammu,samsigma)$k
  print(hynorm[i]/m)
}
```

```
## [1] 0.9998
## [1] 0.9998
## [1] 0.703
## [1] 0.6182
```

这是取不同的标准差对应的拒绝率，下面就这 4 个标准差画图

```
set.seed(1)
mu<-4
sigma<-c(0.05,0.5,2,16)
m<-5000
p<-0.2
sammu<-c(0,5)
samsigma<-c(1,1)
hynorm1<-indepmetro(mu,sigma[1],m,p,sammu,samsigma)
hynorm2<-indepmetro(mu,sigma[2],m,p,sammu,samsigma)
hynorm3<-indepmetro(mu,sigma[3],m,p,sammu,samsigma)
hynorm4<-indepmetro(mu,sigma[4],m,p,sammu,samsigma)
hynorm<-cbind(hynorm1$x,hynorm2$x,hynorm3$x,hynorm4$x)
par(mfrow=c(2,2))## 方便比较四个图以 2*2 的形式排列
for(j in 1:4){
  plot(hynorm[,j],type="l",xlab=bquote(sigma==.(round(sigma[j],3))),ylab="X",ylim=range
}
```




```
par(mfrow=c(1,1))
```

从这个图看到，前面 2 个由于拒绝过多，基本没有收敛的意思，后面都可以收敛，这里可以看到收敛性最好的链是 $\sigma = 2$ 的时候，这里链的波动最小，这时对应的样本均值和方差计算如下, 输出结果前面为均值, 后面为方差

```
mean1=mean(hynorm3$x)
var1=var(hynorm3$x)
print(c(mean1,var1))
```

```
## [1]  2.609605 10.975103
```

(2)

提议分布改为柯西分布，同理，不妨取提议分布为 $C(4, \lambda)$ 计算的 R 代码如下

```
cindepmetro<-function(mu,lambda,m,p,samsigma,sammu){
  # lambda,mu 为提议正态分布的参数
  # m 为链的长度
```

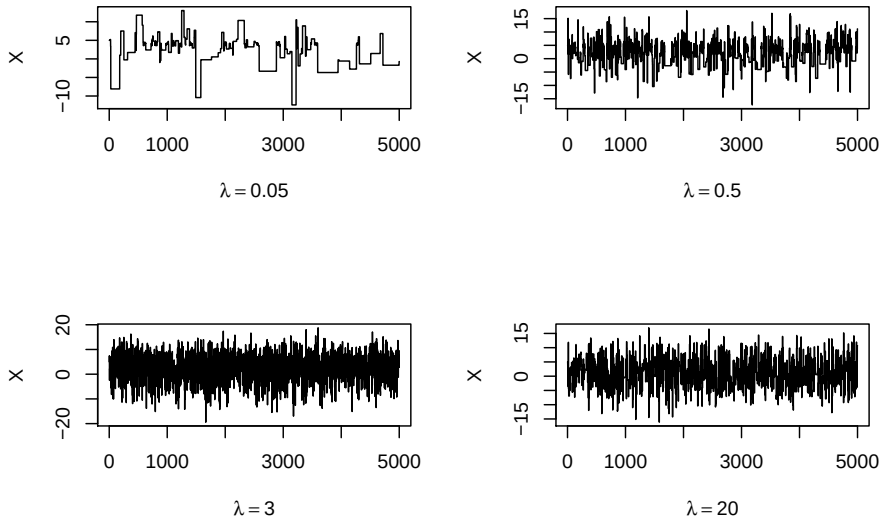
```r
  # p,samsigma,sammu 为要抽样的分布的参数
  xt<-numeric(m)
  u<-runif(m)
  ## 对提议分布抽样
  y<-rcauchy(m,mu,lambda)
  k<-0
  xt[1]=5
  for(i in 2:m){
    fy<-p*dnorm(y[i],sammu[1],samsigma[1])+(1-p)*dnorm(y[i],sammu[2],samsigma[2])
    fx<-p*dnorm(xt[i-1],sammu[1],samsigma[1])+(1-p)*dnorm(xt[i-1],sammu[2],samsigma[2])
    r<-fy/fx*dcauchy(xt[i-1],mu,lambda)/dcauchy(y[i],mu,lambda)
    if(u[i]<=r)
      xt[i]<-y[i]
    else{
      xt[i]<-xt[i-1]
      k<-k+1
      }

    }

  return(list(x=xt,k=k))
}
mu<-4
lambda<-c(0.05,0.5,3,20)## 这里我从小到大取了一些值
m<-5000
p<-0.2
sammu<-c(0,5)
samsigma<-c(1,1)
chynorm<-numeric(4)
## 这一部分主要验证代码确实可以运行
for(i in 1:4){
  chynorm[i]<-cindepmetro(mu,lambda[i],m,p,sammu,samsigma)$k
  print(chynorm[i]/m)
```

```
}
```

```
## [1] 0.9672
## [1] 0.7326
## [1] 0.3968
## [1] 0.7548
```

同样的我们画图如下

```
set.seed(1)
mu<-4
lambda<-c(0.05,0.5,3,20)
m<-5000
p<-0.2
sammu<-c(0,5)
samsigma<-c(1,1)
chynorm1<-cindepmetro(mu,lambda[1],m,p,sammu,samsigma)
chynorm2<-cindepmetro(mu,lambda[2],m,p,sammu,samsigma)
chynorm3<-cindepmetro(mu,lambda[3],m,p,sammu,samsigma)
chynorm4<-cindepmetro(mu,lambda[4],m,p,sammu,samsigma)
chynorm<-cbind(chynorm1$x,chynorm2$x,chynorm3$x,chynorm4$x)
par(mfrow=c(2,2))## 方便比较四个图以 2*2 的形式排列
for(j in 1:4){
  plot(chynorm[,j],type="l",xlab=bquote(lambda==.(round(lambda[j],3))),ylab="X",ylim=ra
}
```




```
par(mfrow=c(1,1))
```

这四种标准差对应的链都能收敛，不过当标准差较小时拒绝的比较多, 而且很慢，$\sigma = 3$ 时表现可以认为是其中最好的，这里可以看到链是最为平稳的，对应的样本均值和方差为

```
mean2=mean(chynorm3$x)
var2=var(chynorm3$x)
print(c(mean2,var2))
```

```
## [1]  1.25050 26.54995
```

不过和第一小题比较一下，好像表现没有直接用正态当提议分布好。

6.11 题

(1)

证明: 由题我们得到 $\pi(\theta_i|\lambda) = \lambda e^{-\lambda\theta_i}$, 且其在给定了 λ 条件下独立 $\pi(\lambda) = e^{-\lambda}$ 从而得到 θ 规范先验

$$\begin{aligned}
\pi(\theta) &= \pi(\theta_1, ...\theta_k) \\
&= \int_0^{+\infty} (\prod_{i=1}^{k} \pi(\theta_i|\lambda))\pi(\lambda)d\lambda \\
&= \int_0^{+\infty} \lambda^k e^{-\lambda(\sum_{i=1}^{k}\theta_i+1)}d\lambda \\
&= (\sum_{i=1}^{k}\theta_i + 1)^{-(k+1)} \int_0^{+\infty} \lambda^k e^{-\lambda}d\lambda \\
&\propto (\sum_{i=1}^{k}\theta_i + 1)^{-(k+1)}
\end{aligned}$$

从而我们得到 $\pi(\theta) \propto (\sum_{i=1}^{k}\theta_i + 1)^{-(k+1)}$

(2)

(a) 证明: $\theta_j|\theta_{-j}, \lambda, x$ 的条件分布，这个是给定了 x 之后的后验分布，注意到给定 λ 后那些 θ_i 是独立的，所以可以计算如下

$$\begin{aligned}
\pi(\theta_j|\theta_{-j}, \lambda, x) &= \pi(\theta_j|\lambda, x_j) \\
&\propto p(x_j|\theta_j, \lambda)\pi(\theta_j|\lambda) \\
&\propto \frac{\theta_j^{x_j}}{x_j!}e^{-\theta_j}\lambda e^{-\lambda\theta_j} \\
&\propto \theta_j^{x_j}e^{-(\lambda+1)\theta_j}
\end{aligned}$$

从而该全条件分布为 $\Gamma(x_j + 1, \lambda + 1)$

(b) **证明:** 条件分布可计算如下:

$$\begin{aligned}\pi(\lambda|\theta,x) &\propto [\prod_{i=1}^{k}\pi(x_i|\lambda,\theta_i)\pi(\theta_i|\lambda)]\pi(\lambda)\\ &\propto [\prod_{i=1}^{k}\lambda e^{-\lambda\theta_i}]e^{-\lambda}\\ &\propto \lambda^k e^{-(\sum_{i=1}^{k}\theta_i+1)\lambda}\end{aligned}$$

从而我们得到了 $\lambda|\theta,x$ 的全条件分布为 $\Gamma(k+1,1+\sum_{i=1}^{k}\theta_i)$

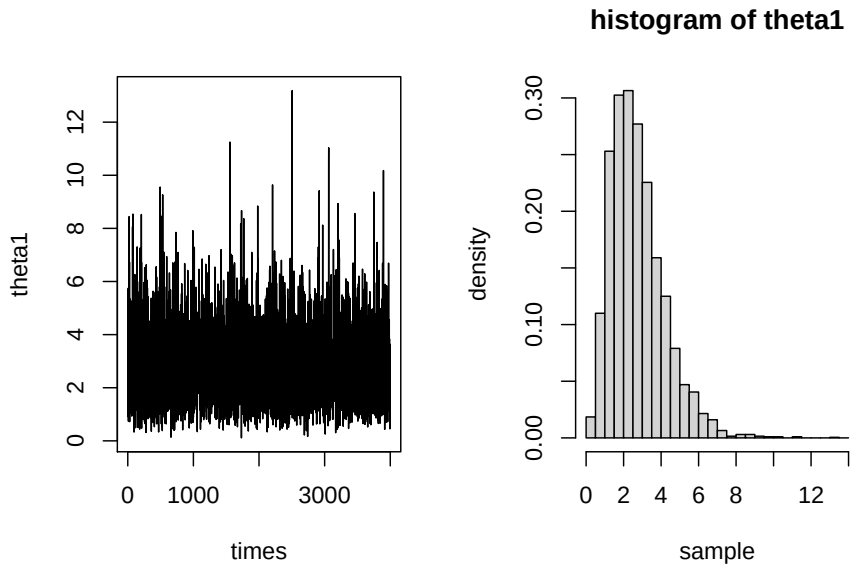
(3)

下面我们使用 Gibbs 抽样方法得到 (θ,λ) 的随机数

```
set.seed(1)## 这里固定一组随机数，不然每次编译都会变，中位数和众数也会变
N<-5000## 链长度
burn<-1000## 预烧期
k<-10
X<-matrix(0,N,k+1)
x<-c(3,1,4,2,5,3,2,2,0,4)
# 初值
X[1,]<-c(rep(1,times=k+1))
for(i in 2:N){
  for(j in 1:k){
    xt<-rgamma(1,x[j]+1,X[i-1,k+1]+1)
    X[i,j]<-xt
  }
  lambdat<-rgamma(1,1+k,1+sum(X[i,1:k]))
  X[i,k+1]<-lambdat
}
b<-burn+1
Y<-X[b:N,]
```

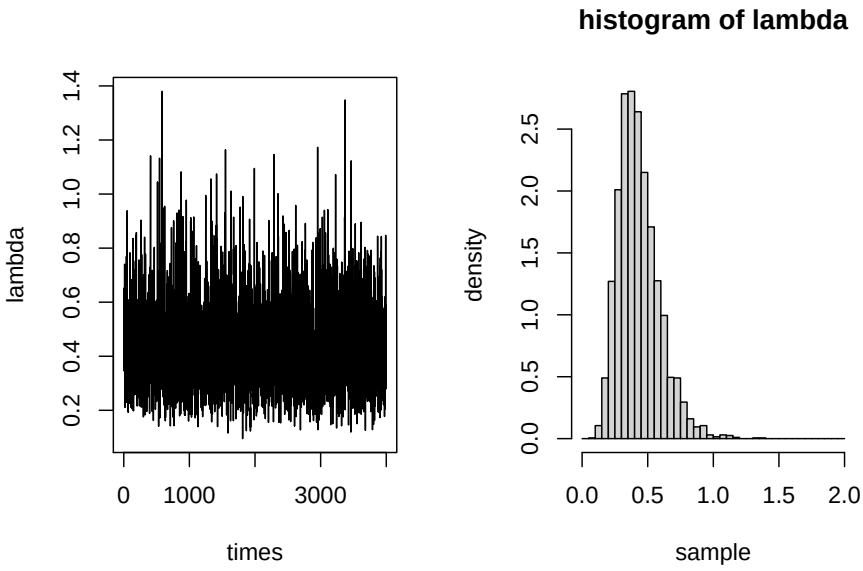
下面我们分别画 θ_1,λ 的轨迹图和直方图

```
par(mfrow=c(1,2))
plot(Y[,1],type='l',xlab = "times",ylab = "theta1")
hist(Y[,1],main="histogram of theta1",breaks=c(seq(0,14,0.5)),xlab = "sample",ylab = "d
```




```
par(mfrow=c(1,1))
```

```
par(mfrow=c(1,2))
plot(Y[,11],type='l',xlab = "times",ylab = "lambda")
hist(Y[,11],main="histogram of lambda",breaks=c(seq(0,2,0.05)),xlab = "sample",ylab = "
```




```
par(mfrow=c(1,1))
```

下面来看 θ_1, λ 的后验中位数和后验众数

```
thetafen<-quantile(Y[,1],0.5)
lambdafen<-quantile(Y[,11],0.5)
print(thetafen)##theta 中位数
```

```
##      50%
## 2.519996
```

```
print(lambdafen)##lambda 中位数
```

```
##       50%
## 0.4117845
```

θ_1 后验中位数 2.520,λ 后验中位数为 0.412 从直方图可以很容易读出，θ_1 的后验众数为 2.25,λ 的后验众数为 0.375

下面计算向量的后验期望，协方差阵和相关系数阵

```r
posmean=colMeans(Y)## 样本均值来估计后验期望
poscov=cov(Y)## 样本协方差阵估计后验协方差阵
poscor=cor(Y)## 样本相关系数估计后验相关系数
print(posmean)
```

```
##  [1] 2.7572692 1.4008815 3.5753874 2.1097862 4.2417133 2.8774841 2.1436387
##  [8] 2.1117344 0.6985967 3.4917234 0.4324862
```

```r
print(poscov)
```

```
##              [,1]         [,2]        [,3]        [,4]        [,5]        [,6]
##  [1,]  2.05177444  0.023674802  0.07720560  0.04377098  0.12796682  0.07154110
##  [2,]  0.02367480  1.061357136  0.05063539  0.04567201  0.04016449  0.10344956
##  [3,]  0.07720560  0.050635389  2.64032347  0.06159369  0.15003832  0.10829583
##  [4,]  0.04377098  0.045672014  0.06159369  1.60300899  0.11054352  0.08791120
##  [5,]  0.12796682  0.040164491  0.15003832  0.11054352  3.31858420  0.11998772
##  [6,]  0.07154110  0.103449559  0.10829583  0.08791120  0.11998772  2.19664772
##  [7,]  0.04444166  0.044227983  0.05552431  0.08346156  0.08008909  0.06356538
##  [8,]  0.09824218  0.031618264  0.10171168  0.04547146  0.12148315  0.04028661
##  [9,]  0.02335113 -0.004721606  0.02315566  0.01768196  0.04647128  0.03734006
## [10,]  0.09862766  0.076092594  0.09948093  0.08711036  0.18685777  0.15754307
## [11,] -0.04306495 -0.024348351 -0.05378661 -0.03505887 -0.07294965 -0.05015113
##              [,7]         [,8]         [,9]       [,10]       [,11]
##  [1,]  0.04444166  0.098242178  0.023351133  0.09862766 -0.04306495
##  [2,]  0.04422798  0.031618264 -0.004721606  0.07609259 -0.02434835
##  [3,]  0.05552431  0.101711677  0.023155656  0.09948093 -0.05378661
##  [4,]  0.08346156  0.045471461  0.017681960  0.08711036 -0.03505887
##  [5,]  0.08008909  0.121483151  0.046471278  0.18685777 -0.07294965
##  [6,]  0.06356538  0.040286610  0.037340055  0.15754307 -0.05015113
##  [7,]  1.54448299  0.044486233  0.016444979  0.08205589 -0.03224058
##  [8,]  0.04448623  1.561547473  0.002073646  0.07626857 -0.03456439
##  [9,]  0.01644498  0.002073646  0.511209504  0.03864532 -0.01128794
## [10,]  0.08205589  0.076268566  0.038645323  2.64130338 -0.05766832
## [11,] -0.03224058 -0.034564386 -0.011287942 -0.05766832  0.02475921
```

```
print(poscor)
```

```
##              [,1]         [,2]        [,3]        [,4]        [,5]        [,6]
##  [1,]  1.00000000  0.016043192  0.03317076  0.02413534  0.04904064  0.03369850
##  [2,]  0.01604319  1.000000000  0.03024787  0.03501478  0.02140105  0.06775131
##  [3,]  0.03317076  0.030247871  1.00000000  0.02993916  0.05068707  0.04496794
##  [4,]  0.02413534  0.035014781  0.02993916  1.00000000  0.04792795  0.04684854
##  [5,]  0.04904064  0.021401050  0.05068707  0.04792795  1.00000000  0.04444063
##  [6,]  0.03369850  0.067751305  0.04496794  0.04684854  0.04444063  1.00000000
##  [7,]  0.02496513  0.034544173  0.02749559  0.05304286  0.03537573  0.03451031
##  [8,]  0.05488524  0.024560073  0.05009157  0.02874043  0.05336568  0.02175219
##  [9,]  0.02280046 -0.006410021  0.01993100  0.01953273  0.03567868  0.03523671
## [10,]  0.04236671  0.045446713  0.03767056  0.04233433  0.06311398  0.06540485
## [11,] -0.19106921 -0.150200155 -0.21036678 -0.17597927 -0.25449448 -0.21504635
##              [,7]         [,8]         [,9]        [,10]       [,11]
##  [1,]  0.02496513  0.054885241  0.022800464  0.04236671 -0.1910692
##  [2,]  0.03454417  0.024560073 -0.006410021  0.04544671 -0.1502002
##  [3,]  0.02749559  0.050091571  0.019930999  0.03767056 -0.2103668
##  [4,]  0.05304286  0.028740435  0.019532730  0.04233433 -0.1759793
##  [5,]  0.03537573  0.053365678  0.035678677  0.06311398 -0.2544945
##  [6,]  0.03451031  0.021752187  0.035236705  0.06540485 -0.2150464
##  [7,]  1.00000000  0.028645505  0.018507268  0.04062647 -0.1648705
##  [8,]  0.02864550  1.000000000  0.002320906  0.03755423 -0.1757854
##  [9,]  0.01850727  0.002320906  1.000000000  0.03325740 -0.1003337
## [10,]  0.04062647  0.037554228  0.033257404  1.00000000 -0.2255068
## [11,] -0.16487047 -0.175785417 -0.100333717 -0.22550685  1.0000000
```

(4)

代码已经在前面给出