Algorithm 1: The Metropolis-Hastings algorithm

Data: $Y \sim p(\vec{\theta})$, $\vec{\theta} = (\lambda_{1:n}, \beta)$
Result: generates $\vec{\theta}_k^{(1)}, ..., \vec{\theta}_k^{(s)} \sim$ iid $p(\vec{\theta}|y)$
Initialization: $\alpha = 1.802, \gamma = 0.01, \delta = 1$; $\lambda_{1:n}^{(0)} = d_{1:n}/t_{1:n}$; $\tilde{\beta} = \beta^{(0)} = 2.459$;
for *the number of chains* $k \leftarrow 1$ **to** K **do**
  for *a chain* $s \leftarrow 1$ **to** S **do**
    1.Select two separate symmetric proposal distributions for $\lambda_{1:n}$ and β ;

$$\lambda_{1:n} \sim Gamma(\alpha + d_{1:n}, \tilde{\beta} + t_{1:n})$$
$$\beta \sim Gamma(\gamma + n\alpha, \delta + \sum \lambda)$$

$$\pi(\lambda, \beta) = \prod_{i=1}^{10} \left\{ \frac{\lambda_i^{\alpha+d_i-1}(\beta + t_i)^{\alpha+d_i}}{\Gamma(\alpha + d_i)} \exp\left[-(\beta + t_i)\lambda_i\right] \right\} \cdot \frac{(\sum \lambda + \delta)^{\gamma+10\alpha} \beta^{\gamma+10\alpha-1}}{\Gamma(\gamma + 10\alpha)} \exp\left\{-(\sum \lambda + \delta)\beta\right\}$$

$$\frac{\pi(\lambda^\star, \beta^\star)}{\pi(\lambda, \beta)} = \prod_{i=1}^{10} \left\{ (\frac{\lambda_i^\star}{\lambda_i})^{\alpha+d_i-1} (\frac{\beta^\star + t_i}{\beta + t_i})^{\alpha+d_i} \exp\left[-(\beta^\star + t_i)\lambda_i^\star + (\beta + t_i)\lambda_i\right] \right\} (\frac{\sum \lambda + \delta}{\sum \lambda^\star + \delta})^{\gamma+10\alpha}$$
$$\cdot (\frac{\beta^\star}{\beta})^{\gamma+10\alpha-1} \cdot \exp\left\{-(\sum \lambda + \delta)\beta^\star + (\sum \lambda + \delta)\beta\right\}$$

Let

$$q(\theta^\star|\theta^{(s)}) = g(\theta^\star) = \prod_{i=1}^{10} \left\{ \frac{\lambda_i^{\alpha+d_i-1}(\tilde{\beta} + t_i)^{\alpha+d_i}}{\Gamma(\alpha + d_i)} \exp\left[-(\tilde{\beta} + t_i)\lambda_i\right] \right\} \cdot \frac{(\sum \lambda + \delta)^{\gamma+10\alpha} \beta^{\gamma+10\alpha-1}}{\Gamma(\gamma + 10\alpha)} \exp\left\{-(\sum \lambda + \delta)\beta\right\}$$

OR $g(\theta^\star) = \text{dNormal}(\theta|\mu, \sigma^2)$ OR $g(\theta^\star) = 1$;

$$r = \frac{\pi(\lambda^\star, \beta^\star) g(\lambda, \beta)}{\pi(\lambda, \beta) g(\lambda^\star, \beta^\star)} = \prod_{i=1}^{10} \left\{ (\frac{\beta^\star + t_i}{\beta + t_i})^{\alpha+d_i} \right\} \cdot \exp\left\{(\beta - \tilde{\beta})\lambda + (\tilde{\beta} - \beta^\star)\lambda^\star\right\}$$

    2. update $\lambda_{1:n}$;
    a) sample $\lambda_{1:n}^\star \sim Gamma(\alpha + d_{1:n}, \beta^{(s)} + t_{1:n})$;
    b) compute $r(\lambda_{1:n}, \lambda_{1:n}^\star, \tilde{\beta}, \beta)$;
    c) set **if** *the ratio* $r > 1$ **then**
      $\lambda_{1:n}^{(s+1)} \longleftarrow \lambda_{1:n}^\star$ with probability $\min(1, r)$
    else
      $\lambda_{1:n}^{(s+1)} \longleftarrow \lambda_{1:n}^{(s)}$ with probability $\max(0, 1 - r)$
    3. update β;
    a) sample $\beta^\star \sim Gamma(\gamma + n\alpha, \delta + \sum \lambda^\star)$;
    b) compute $r = (\lambda_{1:n}, \lambda_{1:n}^\star, \beta, \beta^\star)$;
    c) set **if** *the ratio* $r > 1$ **then**
      $\beta^{(s+1)} \longleftarrow \beta^\star$ with probability $\min(1, r)$
    else
      $\beta^{(s+1)} \longleftarrow \beta^{(s)}$ with probability $\max(0, 1 - r)$
  generates a set of $\theta^{(s+1)}$ given $\theta^{(s)}$;

Table 1: The list of function setting

Distribution	Candidate	transition probability kernel $g(x)$			Sampler	
		Initial values	Shape	Rate	Shape	Rate
Note	$\lambda_{1:n}$	$\lambda_0 = (\tilde{\lambda}, 100\tilde{\lambda}, \tilde{\lambda}/100, \lambda_0)$	/	/	$d+\alpha$	$\beta^\star + t$
	β	$\beta_0 = (\tilde{\beta}, 100\tilde{\beta}, \tilde{\beta}/100, \beta_0)$	/	/	$\gamma + n\alpha$	$\delta + \sum \lambda^\star$
Gamma	$\lambda_{1:n}$	$\lambda_0 = (\tilde{\lambda}, 8\tilde{\lambda}, \tilde{\lambda}/8, \lambda_0)$	$d+\alpha$	$\beta_0 + t$	$d+\alpha$	$\beta^\star + t$
	β	$\beta_0 = (\tilde{\beta}, 2\tilde{\beta}, \tilde{\beta}/8, \tilde{\beta}/4)$	$\gamma + n\alpha$	$\delta + \sum \lambda$	$\gamma + n\alpha$	$\delta + \sum \lambda^\star$
			Rate		Rate	
Expo	$\lambda_{1:n}$	$(\tilde{\lambda}, 10\tilde{\lambda}, \tilde{\lambda}/10, \lambda_0)$	λ_0		$(d+\alpha)/(\beta^\star + t)$	
	β	$(\tilde{\beta}, 10\tilde{\beta}, \tilde{\beta}/10, \beta_0)$	β_0		$(\gamma + n\alpha)/(\delta + \sum \lambda^\star)$	
			mean	sd	mean	sd
Normal	$\lambda_{1:n}$	$(\tilde{\lambda}, 10\tilde{\lambda}, \tilde{\lambda}/10, \lambda_0)$	λ_0	1	$\frac{d+\alpha}{\beta^\star+t}$	1
	β	$(\tilde{\beta}, 10\tilde{\beta}, \tilde{\beta}/10, \beta_0)$	β_0	1	$\frac{\gamma+n\alpha}{\delta+\sum\lambda^\star}$	1
			Range		Range	
Uniform	$\lambda_{1:n}$	$(\tilde{\lambda}, 10\tilde{\lambda}, \tilde{\lambda}/10, \lambda_0)$	$(0, 2\lambda_0)$		$(0, 2\frac{d+\alpha}{\beta^\star+t})$	
	β	$(\tilde{\beta}, 10\tilde{\beta}, \tilde{\beta}/10, \beta_0)$	$(0, 2\beta_0)$		$(0, 2\frac{\gamma+n\alpha}{\delta+\sum\lambda^\star})$	

The initial setting

```
pump <- matrix(c(5, 94.320,1, 15.720,5, 62.880,14, 125.760,3, 5.240,
                 19, 31.440,1, 1.048,1, 1.048,4, 2.096,22, 10.480),2,10)
pump <-t(pump)
colnames(pump) <- c("Failures", "Times")
d <- pump[,1]
t <- pump[,2]
n <- length(d)

alpha<-1.802; gamma <- 0.01; delta<- 1 #def hyperparameters
beta <- beta0 <- gamma/delta #initialize lambda and beta
lambda <- lambda0 <- d/t; lambda.star <- rep(NA,n)
beta_true  <- 2.459
lambda_true<-  c(0.065757300, 0.136413079, 0.098165113, 0.121128692, 0.57794838, 0.60457447, 0.70749039
S <- 10000; # set.seed(121)
K <-  100
skeep<-seq(1,S,by=10); skeep2<-seq(1,S,by=20)
burnin <- 1:(S/2)
THETA_0 <- THETA_g <- THETA_n<-THETA_u <- THETA_e <- THETA_1<- matrix(NA,K,12)
par.names <- c("Beta", "Lambda1","Lambda2","Lambda3","Lambda4","Lambda5","Lambda6","Lambda7","Lambda8",
colnames(THETA_0)<-colnames(THETA_g)<-colnames(THETA_e)<-colnames(THETA_n)<-colnames(THETA_u)<-colnames
```

The kernel setting

```
lambda_0 <- matrix(c(lambda_true,lambda_true*100,lambda_true/100,lambda0),10,4)
beta_0 <- c(beta_true,beta_true*100,beta_true/100,beta0)
```

```
lambda_g <- matrix(c(lambda_true,lambda_true*8,lambda_true/8,lambda0),10,4)
beta_g <- c(beta_true,beta_true*2,beta_true/8,beta_true/4)
lambda_e <- matrix(c(lambda_true,lambda_true*2,lambda_true*3/4,lambda0),10,4)  # matrix(c(lambda0/2,lam
beta_e <- c(beta_true,beta_true*2,beta_true*3/4,beta0) # c(beta0,beta0,beta0,beta0)
lambda_n <- matrix(c(lambda_true,lambda_true*2,lambda_true/2,lambda0),10,4)
beta_n <- c(beta_true,beta_true*2,beta_true/2,beta0)
lambda_u <- matrix(c(lambda_true,lambda_true/2,lambda_true/8,lambda0),10,4)
beta_u <- c(beta_true,beta_true/2,beta_true/8,beta0)
```

Define Pi function

```
pi <- function(a,b){
d.lambda<- dgamma(a,(d+alpha),(b+t))
d.beta <-dgamma(b,(gamma+n*alpha),(delta+sum(a)))
return(prod(d.lambda)*d.beta)
}
```

Conduct Gibbs Algorithm

```
Gibbs_G <- function(lambda_g,beta_g,S){ #Gibbs Sampler
Theta<-matrix(NA,S,11) ;  set.seed(121)
lambda <- lambda0; beta <- beta0
Theta.gibbs <- matrix(NA,ncol=(n+1),nrow=S) #Gibbs Sampelr variables
for(s in 1:S){
lambda<- rgamma(n,shape=(d+alpha),rate=(beta+t))  # sample lambda
beta <-  rgamma(1,shape=(gamma+n*alpha),rate=(delta+sum(lambda)))  # sample beta
Theta.gibbs[s,] <- c(beta,lambda)   # store draws
}
return(Theta.gibbs)
}
Theta.gibbs<- Gibbs_G(lambda_g[,4],beta_g[4],10000)
```

Use $g(.)$ in the notes

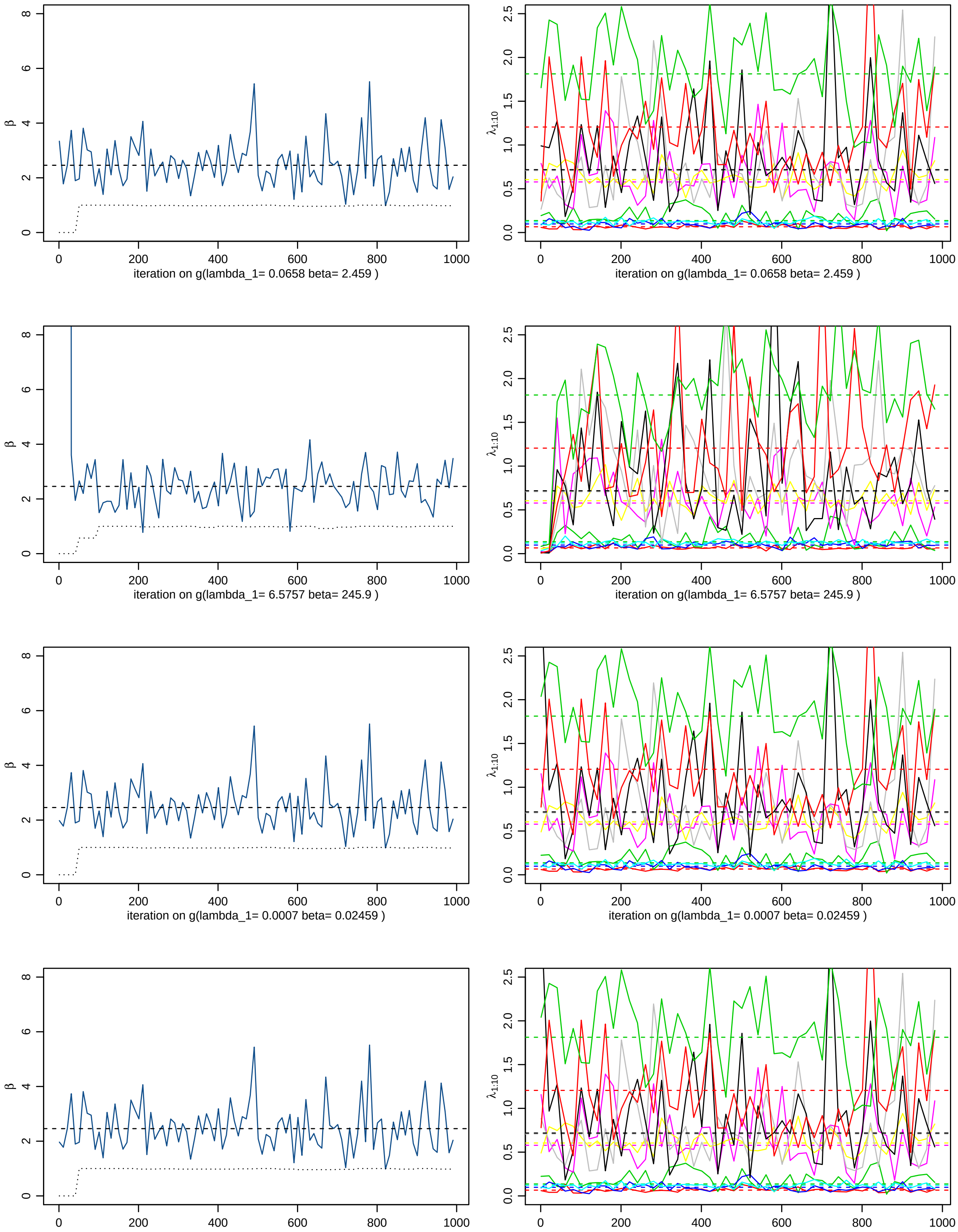
- One Chain

```
ratio<- function(lambda,lambda.star,beta,beta.star){
for(i in 1:n){
lik <- ((beta.star+t[i])/(beta+t[i]))^(alpha+d[i])
expo <- exp((beta-beta_0)*lambda+(beta_0-beta.star)*lambda.star)
return(prod(lik)*expo)
}}
MH_0 <- function(lambda_0,beta_0,S){
Theta<-matrix(NA,S,12) ; acr <- acs<-0 ;  set.seed(121)
lambda <- lambda_0; beta <-beta_0
for(s in 1:S) {
lambda.star <- rgamma(n,(d+alpha),(beta+t)) # sample lambda
r_lambda<-ratio(lambda,lambda.star,beta,beta)
```

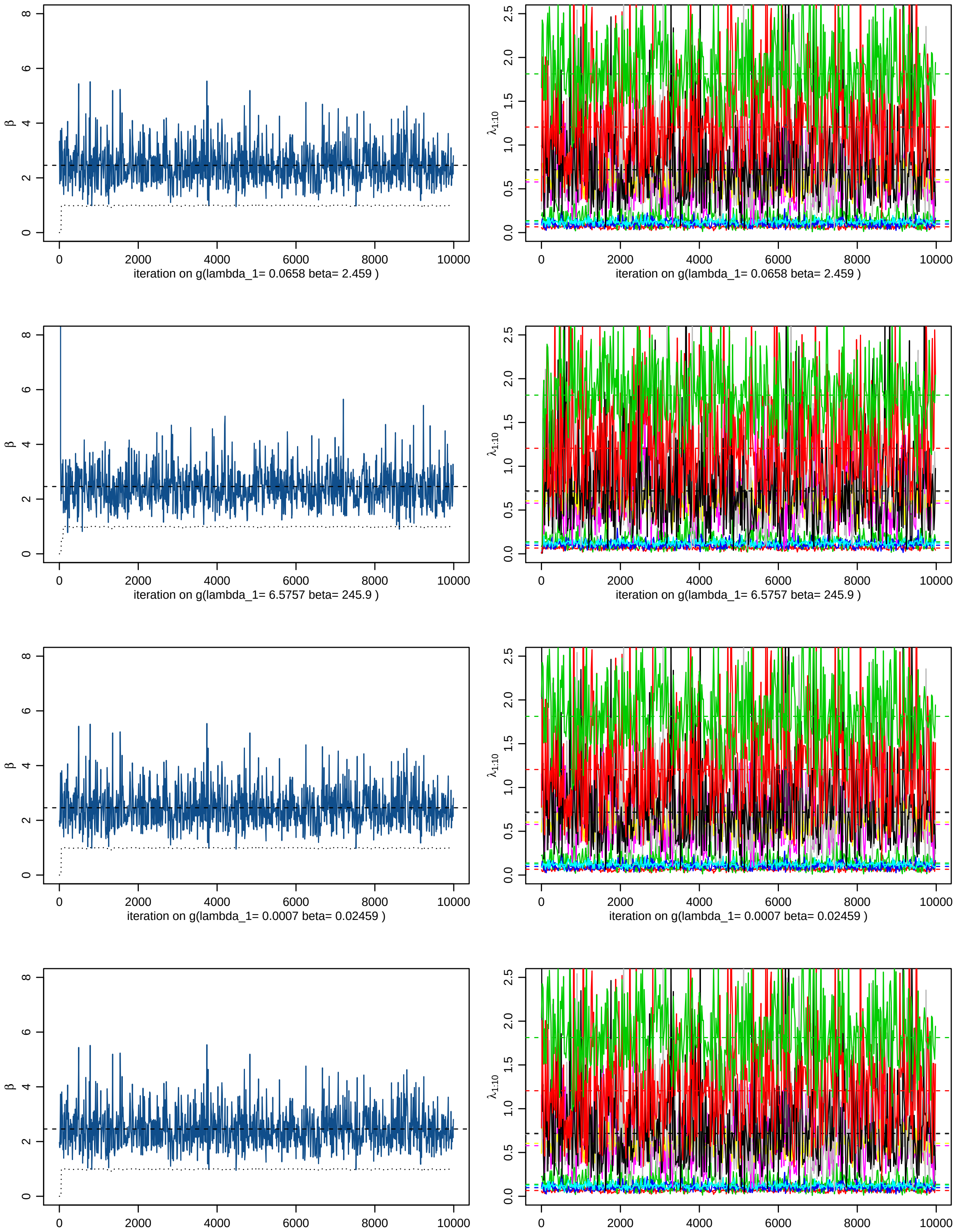
```
  if((runif(1))<r_lambda) { lambda<-lambda.star; acs<-acs+1
beta.star <- rgamma(1,(gamma+n*alpha),(delta+sum(lambda)))  # sample beta
r_beta<-ratio(lambda,lambda,beta,beta.star)
  if((runif(1))<r_beta) { beta<-beta.star ; acs<-acs+1}}
    if(s%%50==0) {acr <- acs/100; acs<-0}
  Theta[s,]<-c(beta,lambda,acr)
}
return(Theta)
}
```

```
# ONE STEP version
ratio<- function(lambda,lambda.star,beta,beta.star){
for(i in 1:n){
lik <- ((beta.star+t[i])/(beta+t[i]))^(alpha+d[i])
expo <- exp((beta-beta_0)*lambda+(beta_0-beta.star)*lambda.star)
return(prod(lik)*expo)
}}

MH_0 <- function(lambda_0,beta_0,S){
Theta<-matrix(NA,S,12) ; acr <- acs<-0 ; # set.seed(121)
lambda <- lambda_0; beta <-beta_0
for(s in 1:S)
{
lambda.star <- rgamma(n,(d+alpha),(beta+t)) # sample lambda
beta.star <- rgamma(1,(gamma+n*alpha),(delta+sum(lambda.star)))  # sample beta
  r<-ratio(lambda,lambda.star,beta,beta.star)
  if((runif(1))<r) { beta<-beta.star; lambda<-lambda.star;acs<-acs+1 }
  if(s%%50==0) {acr <- acs/50; acs<-0}
  Theta[s,]<-c(beta,lambda,acr)
}
return(Theta)
}
```



the M−H with kernel in Note,length of chain = 1000

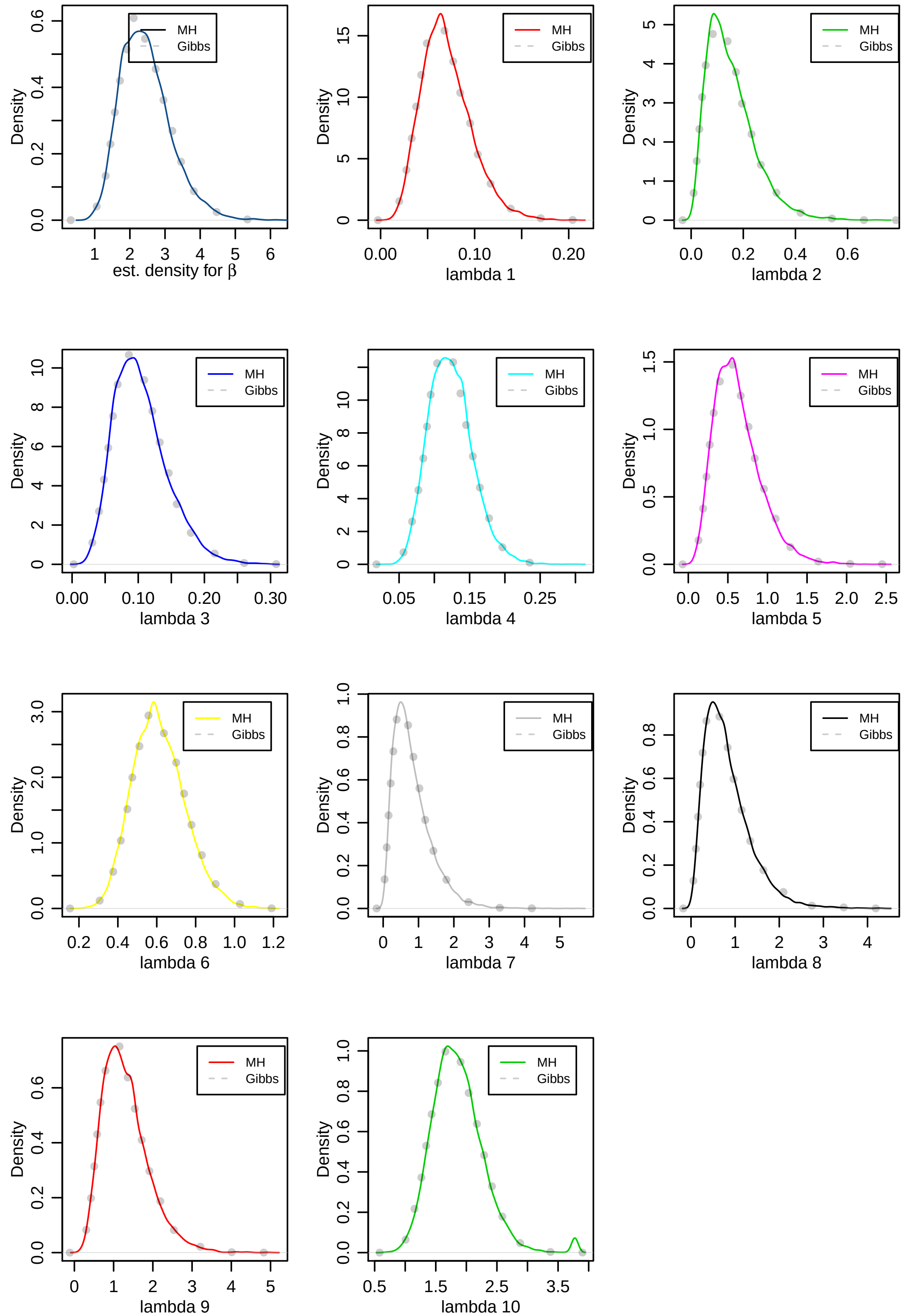


the M−H with kernel in Note,length of chain = 10000

	mean	2.5%	50%	97.5%
Beta	2.468	1.335	2.401	4.117
Lambda1	0.0702	0.0273	0.0668	0.1296
Lambda2	0.1543	0.0298	0.137	0.3804
Lambda3	0.1047	0.0413	0.0994	0.1966
Lambda4	0.1226	0.0705	0.1201	0.1904
Lambda5	0.6185	0.1916	0.5756	1.31
Lambda6	0.612	0.3755	0.601	0.9085
Lambda7	0.8238	0.1467	0.7012	2.138
Lambda8	0.8229	0.1516	0.7146	2.076
Lambda9	1.305	0.4396	1.214	2.725
Lambda10	1.837	1.151	1.807	2.689
Acceptance Rate	0.9871	0.96	0.99	1

```r
S <- 1000; K <-  100
skeep<-seq(1,S,by=10); skeep2<-seq(1,S,by=20); burnin <- 1:(S/2)
ptm <- proc.time()
for(k in 1:K) { # Run K Number of chains by Hint function g(.)
THETA_0[k,] <- MH_0(lambda_0[,4],beta_0[4],S)[S,]
}
ptm_0 <- proc.time() - ptm
```

- Comparing with Gibbs

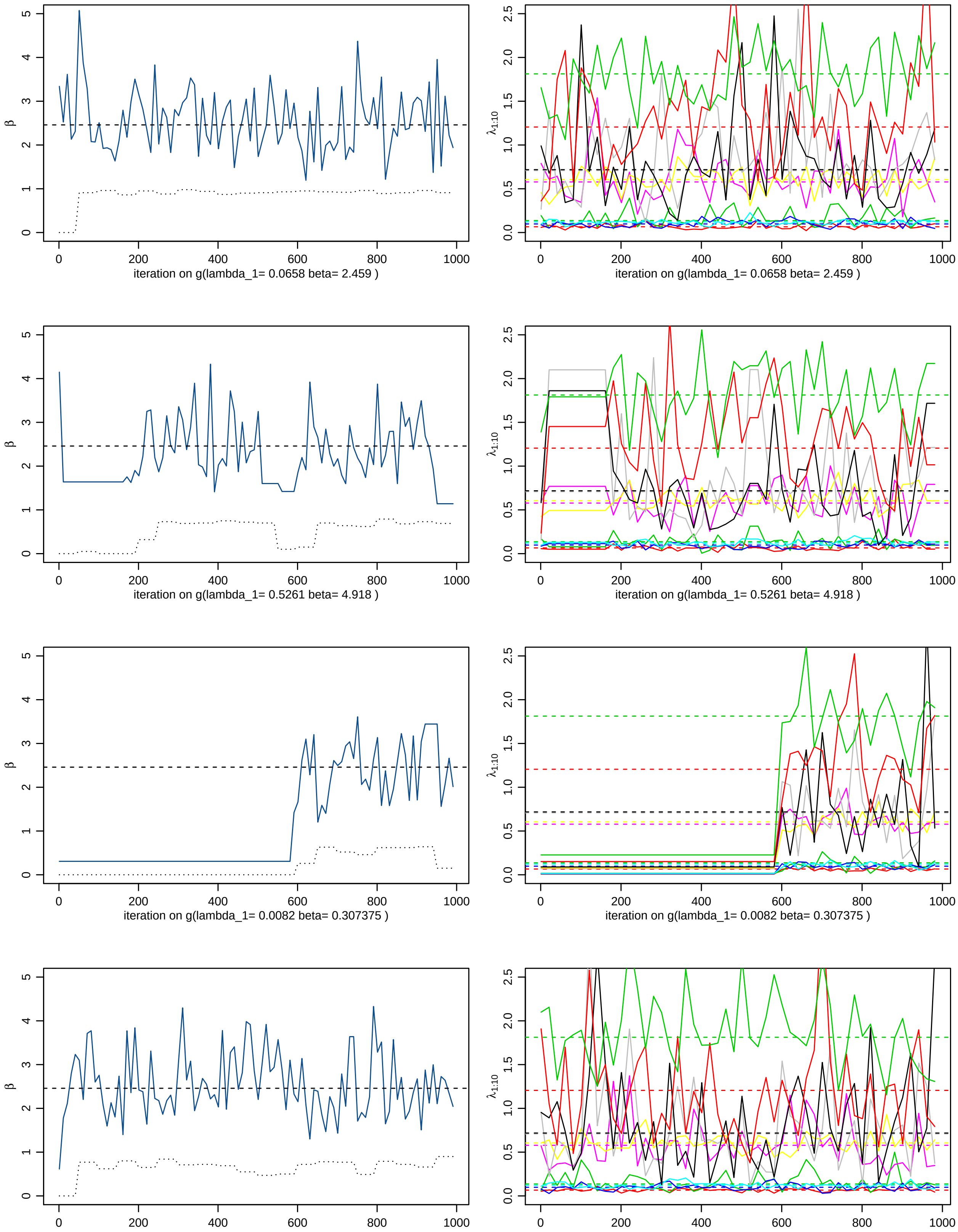


Comparing Gibbs and M−H with kernel in Note

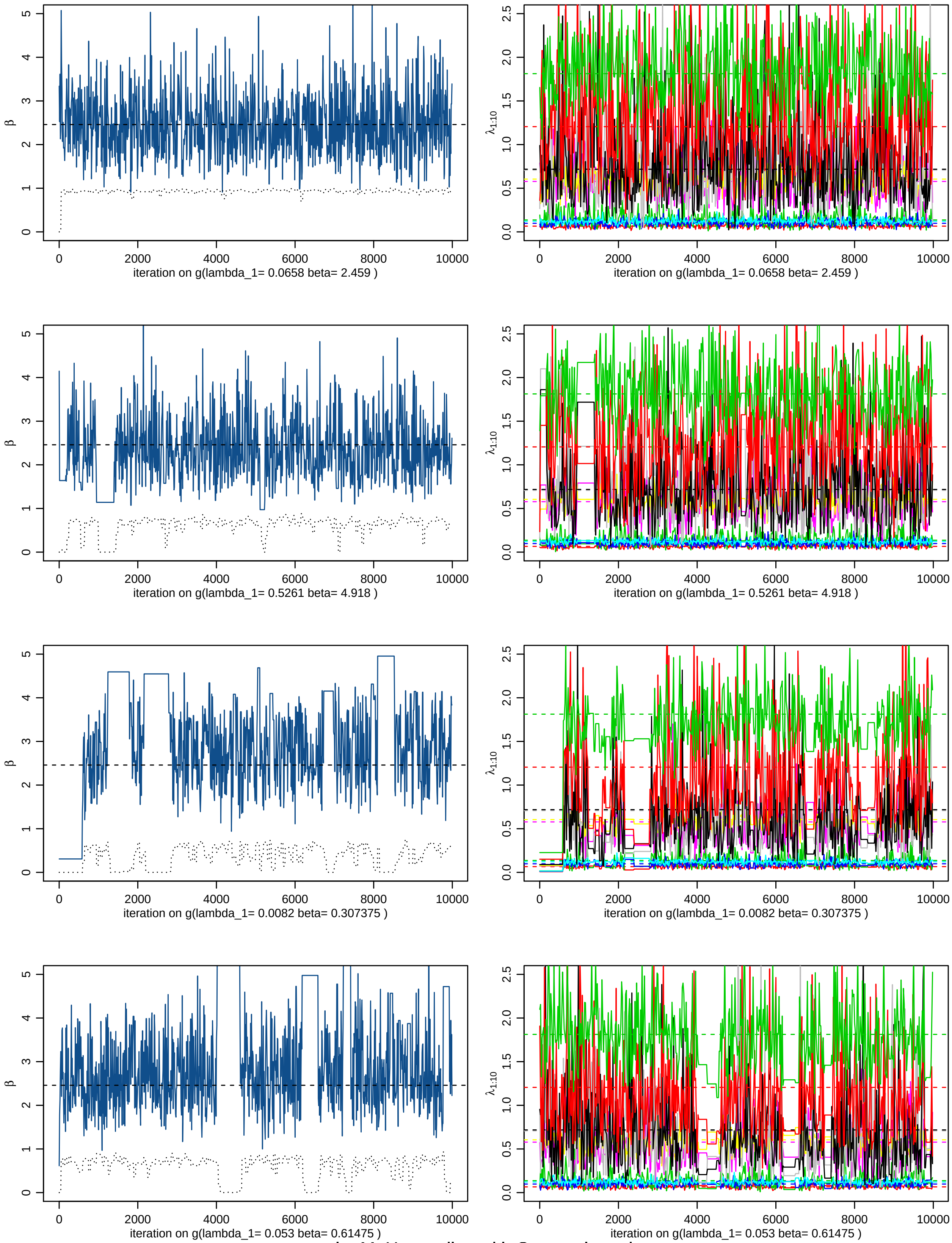
The kenel of Gamma version

```
MH_G <- function(lambda_g,beta_g,S){ # Gamma version
Theta<-matrix(NA,S,12) ; acr <- acs<-0 ;  set.seed(121)
lambda <- lambda_g; beta <-beta_g
g <- function(a,b){
d.lambda<- dgamma(a,(d+alpha),(beta_g+t))
d.beta <-dgamma(b,(gamma+n*alpha),(delta+sum(a)))
return(prod(d.lambda)*d.beta)
}
for(s in 1:S) {
lambda.star <- rgamma(n,(d+alpha),(beta+t)) # sample lambda
r_lambda <- pi(lambda.star,beta)/pi(lambda,beta)*g(lambda,beta)/g(lambda.star,beta)
  if((runif(1))<g(lambda.star,beta)*min(r_lambda,1)) { lambda<-lambda.star; acs<-acs+1

beta.star <- rgamma(1,(gamma+n*alpha),(delta+sum(lambda)))  # sample beta
r_beta<- pi(lambda,beta.star)/pi(lambda,beta)*g(lambda,beta)/g(lambda,beta.star)
  if((runif(1))<g(lambda,beta.star)*min(r_beta,1)) { beta<-beta.star ; acs<-acs+1}}
    if(s%%50==0) {acr <- acs/100; acs<-0}
  Theta[s,]<-c(beta,lambda,acr)
}
return(Theta)
}
```



the M−H sampling with Gamma kernel

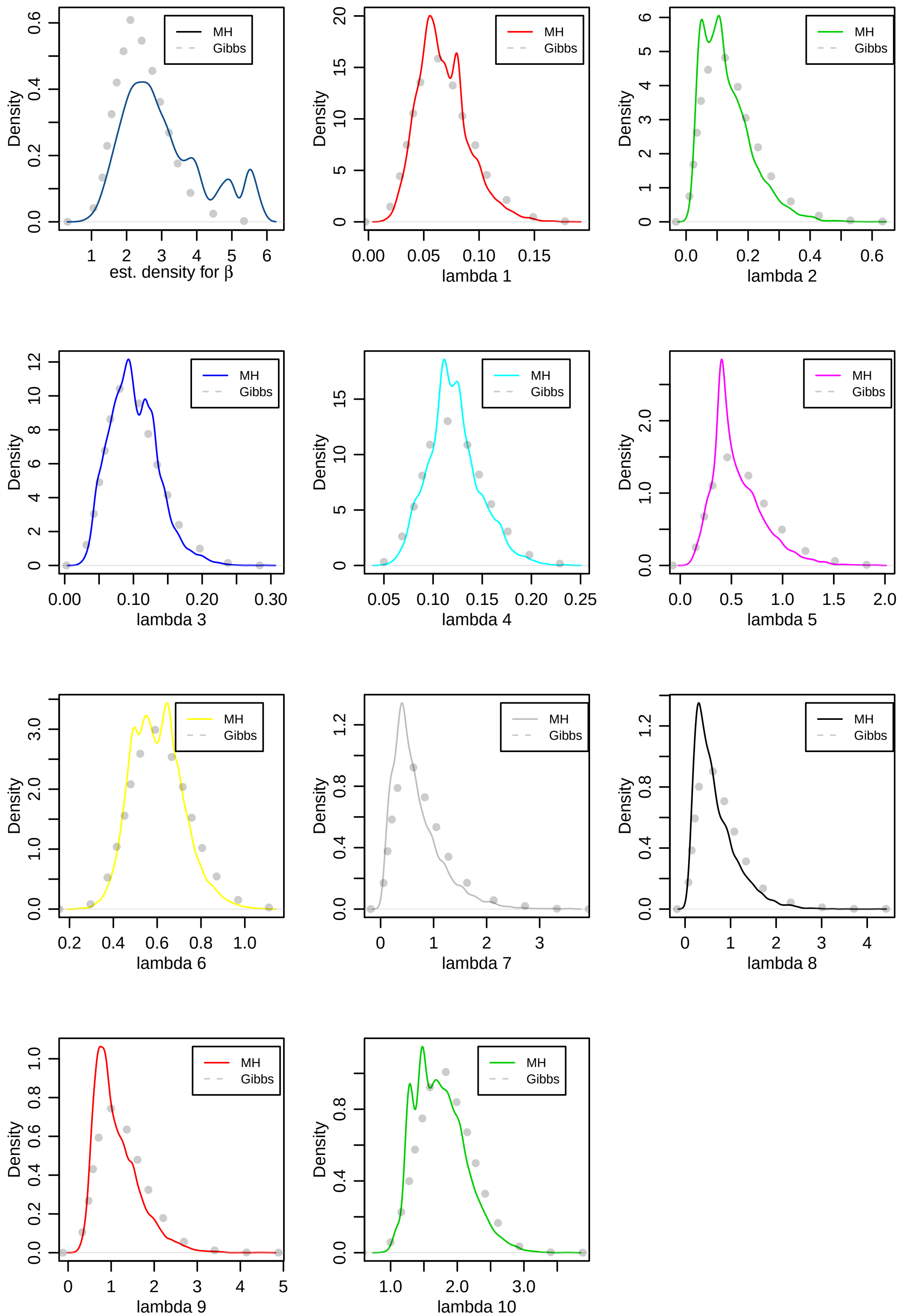


the M−H sampling with Gamma kernel

	mean	2.5%	50%	97.5%
Beta	3.129	1.426	2.832	5.755
Lambda1	0.0673	0.0307	0.0633	0.1213
Lambda2	0.1279	0.035	0.1105	0.3302
Lambda3	0.0997	0.0462	0.0974	0.178
Lambda4	0.1218	0.0743	0.1206	0.1778
Lambda5	0.5469	0.1958	0.4812	1.159
Lambda6	0.601	0.4069	0.5991	0.8472
Lambda7	0.6569	0.1268	0.5251	1.876
Lambda8	0.6484	0.1329	0.5152	1.789
Lambda9	1.111	0.4908	0.9713	2.388
Lambda10	1.727	1.23	1.685	2.576
Acceptance Rate	0.5803	0	0.7	0.87

```
ptm <- proc.time()
for(k in 1:K) { # Run K Number of chains by drawing from Gamma
THETA_g[k,] <- MH_G(lambda_g[,4],beta_g[4],S)[S,]
}
ptm_g <- proc.time() - ptm
```

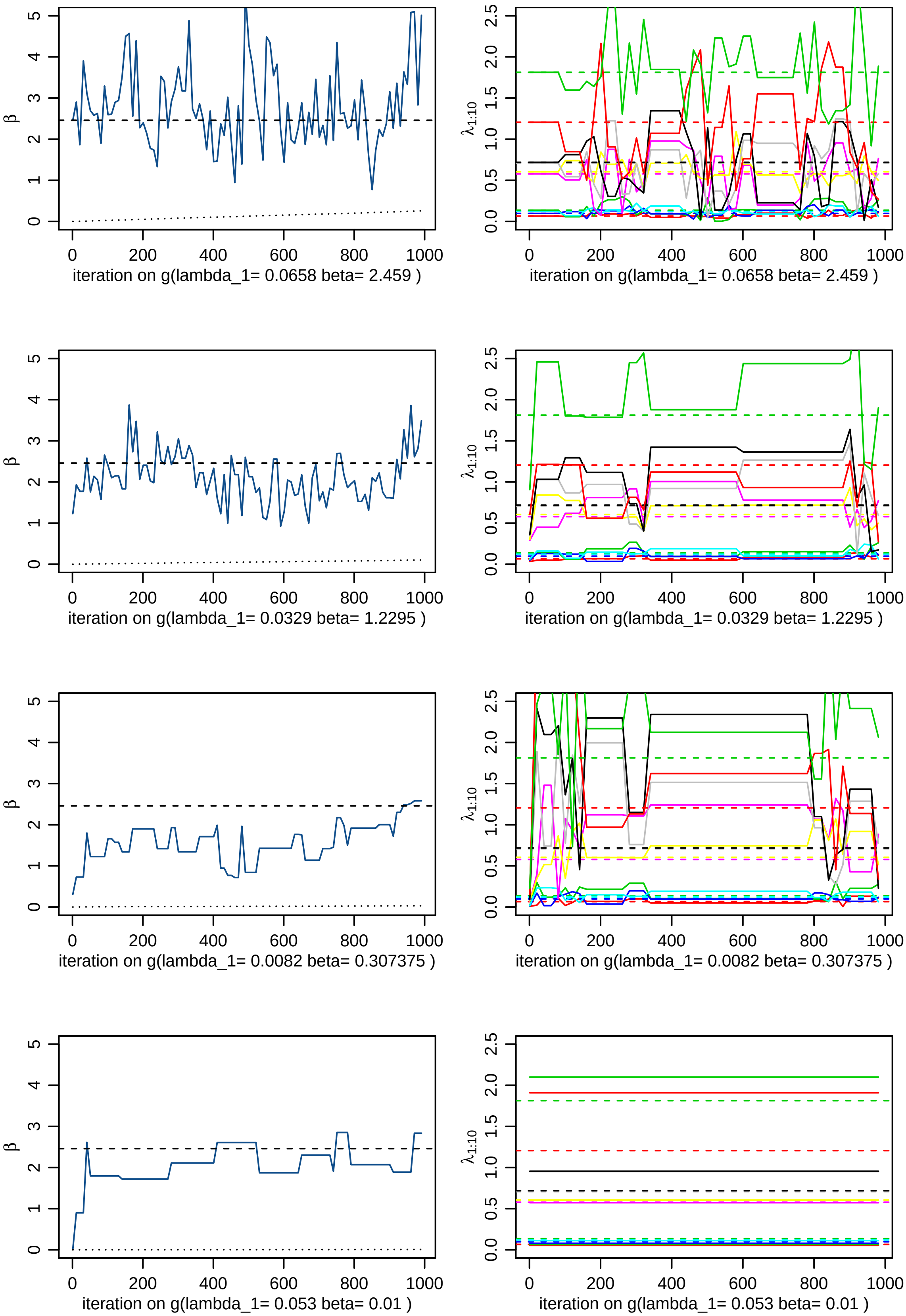
- Comparing with Gibbs



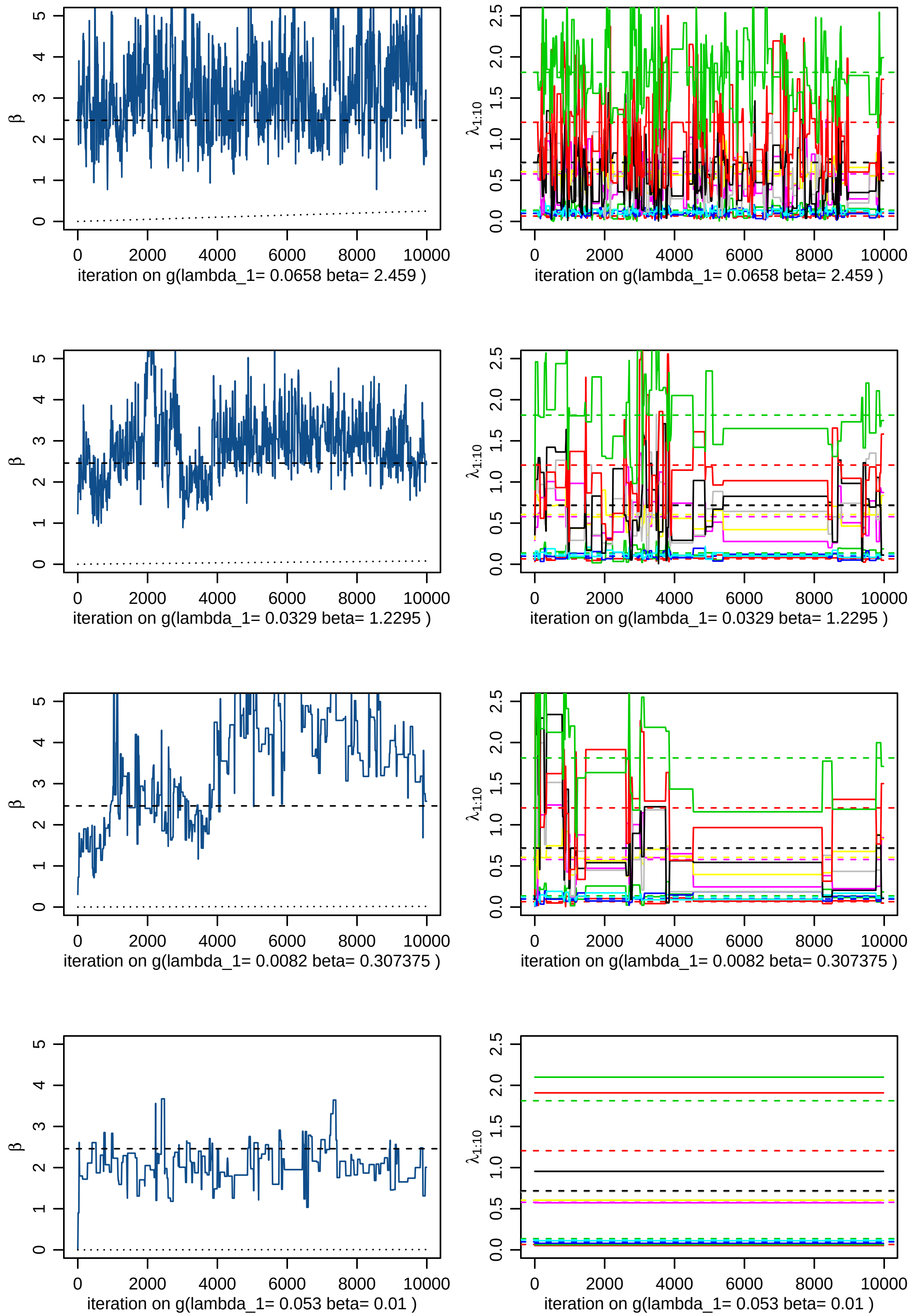
Comparing Gibbs and M−H with Gamma kernel

The kenel of Uniform version

```
MH_U <- function(lambda_u,beta_u,S){ # Uniform version
Theta<-matrix(NA,S,12) ; acr <- acs<-0 ;  set.seed(121)
lambda <- lambda_u; beta <-beta_u
g <- function(a,b){
d.lambda<- dunif(a,0,2*(d+alpha)/(beta_u+t)) #(d+alpha)/(beta0+t),beta0
d.beta <-dunif(b, 0, 2*(gamma+n*alpha)/(delta+sum(a))) #(gamma+n*alpha)/(delta+sum(lambda)),beta0
return(prod(d.lambda)*d.beta)
}
for(s in 1:S) {
lambda.star <- runif(n,0,2*(d+alpha)/(beta+t)) # sample lambda
r_lambda <- pi(lambda.star,beta)/pi(lambda,beta)*g(lambda,beta)/g(lambda.star,beta)
  if((runif(1))<g(lambda.star,beta)*min(r_lambda,1)) { lambda<-lambda.star; acs<-acs+1 }
beta.star <- runif(1,0,2*(gamma+n*alpha)/(delta+sum(lambda)))  # sample beta
r_beta<- pi(lambda,beta.star)/pi(lambda,beta)*g(lambda,beta)/g(lambda,beta.star)
  if((runif(1))<g(lambda,beta.star)*min(r_beta,1)) { beta<-beta.star ; acs<-acs+1}
  acr <- acs/S/2
  Theta[s,]<-c(beta,lambda,acr)
}
return(Theta)
}
```



the M−H sampling with Uniform kernel

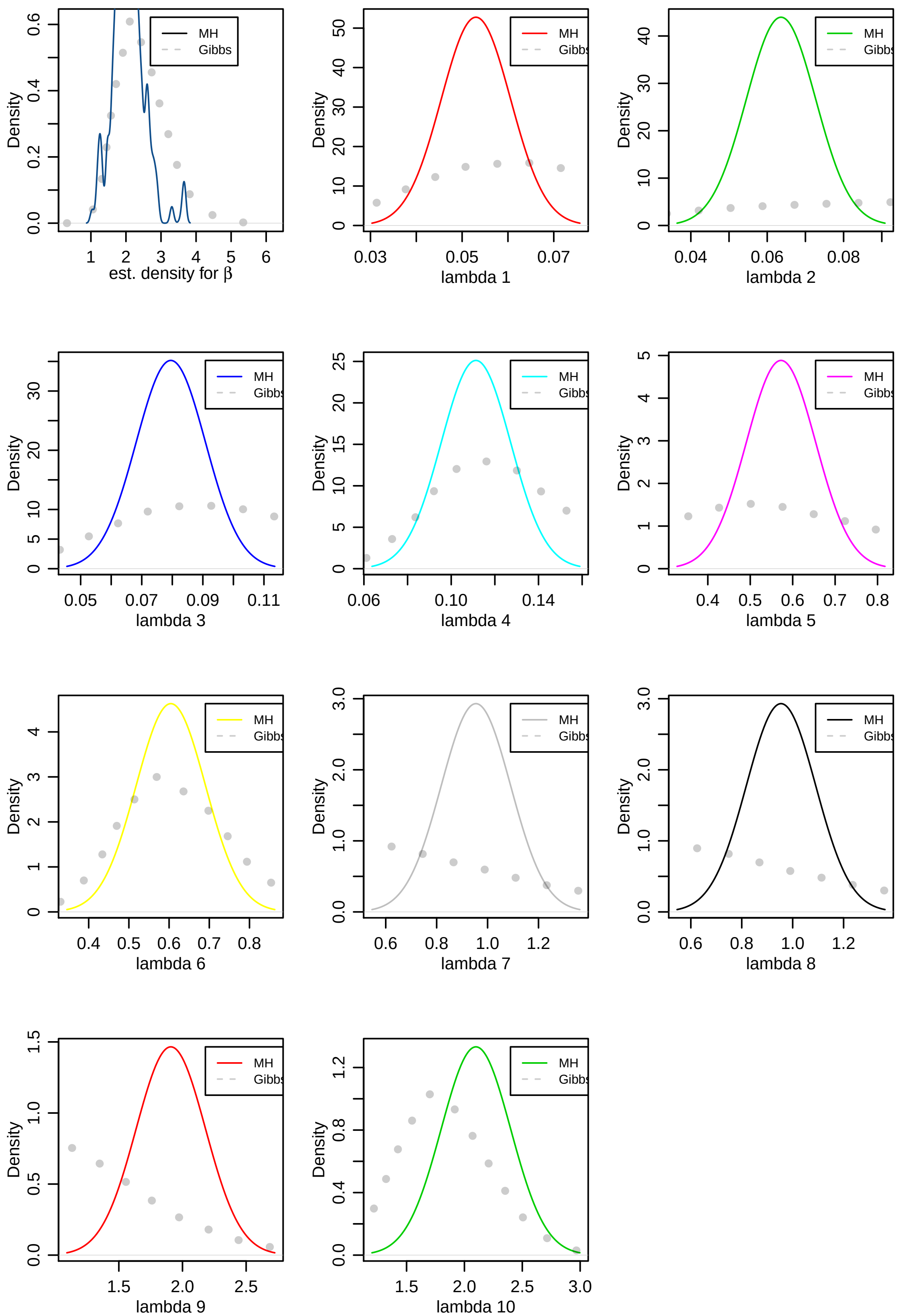


the M−H sampling with Uniform kernel

	mean	2.5%	50%	97.5%
Beta	2.093	1.313	2.074	3.309
Lambda1	0.053	0.053	0.053	0.053
Lambda2	0.0636	0.0636	0.0636	0.0636
Lambda3	0.0795	0.0795	0.0795	0.0795
Lambda4	0.1113	0.1113	0.1113	0.1113
Lambda5	0.5725	0.5725	0.5725	0.5725
Lambda6	0.6043	0.6043	0.6043	0.6043
Lambda7	0.9542	0.9542	0.9542	0.9542
Lambda8	0.9542	0.9542	0.9542	0.9542
Lambda9	1.908	1.908	1.908	1.908
Lambda10	2.099	2.099	2.099	2.099
Acceptance Rate	0.006	0.0043	0.006	0.0076

```r
ptm <- proc.time()
for(k in 1:K) { # Run K Number of chains by drawing from Uniform
THETA_u[k,] <- MH_U(lambda_u[,4],beta_u[4],S)[S,]
}
ptm_u <- proc.time() - ptm
```

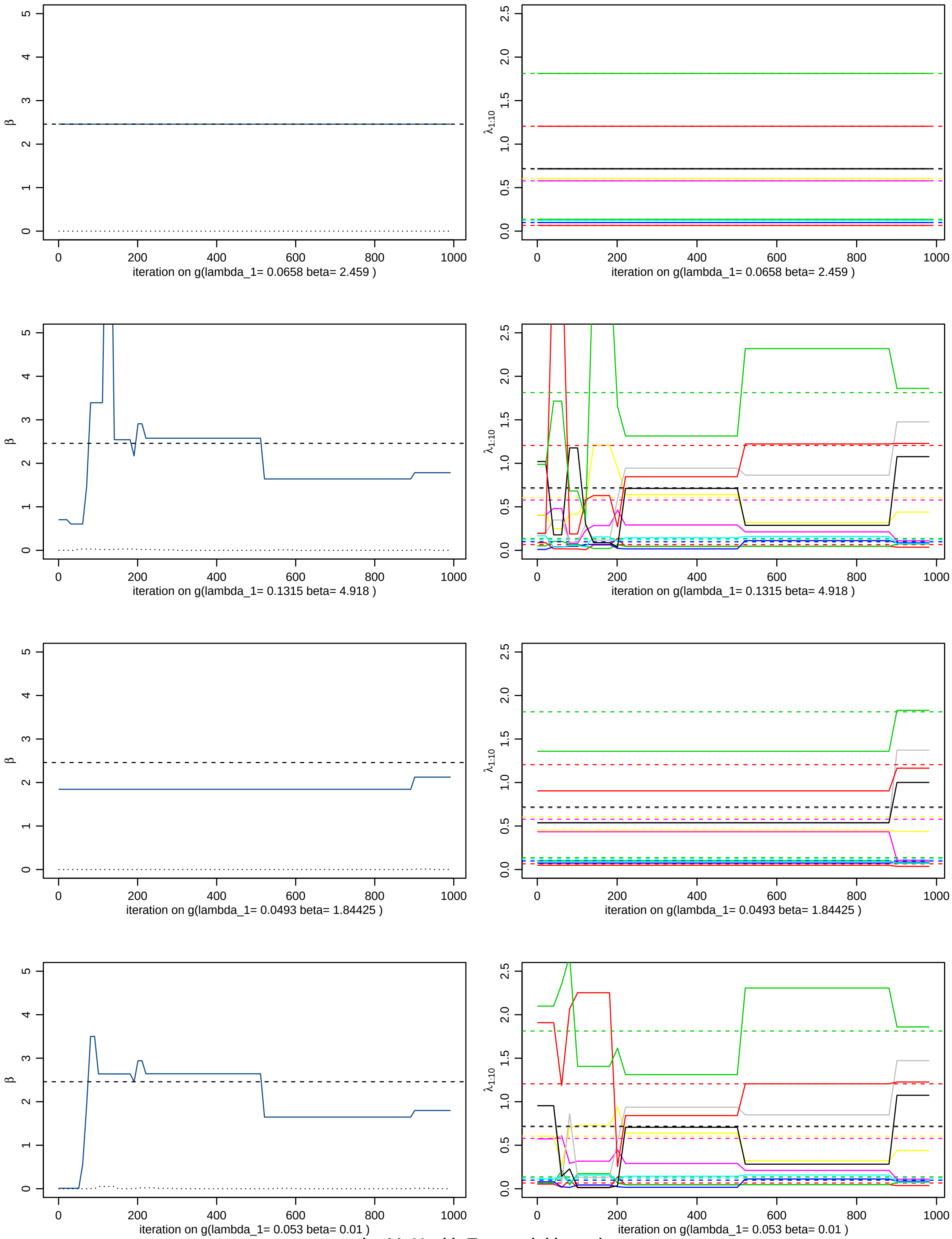
- Comparing with Gibbs



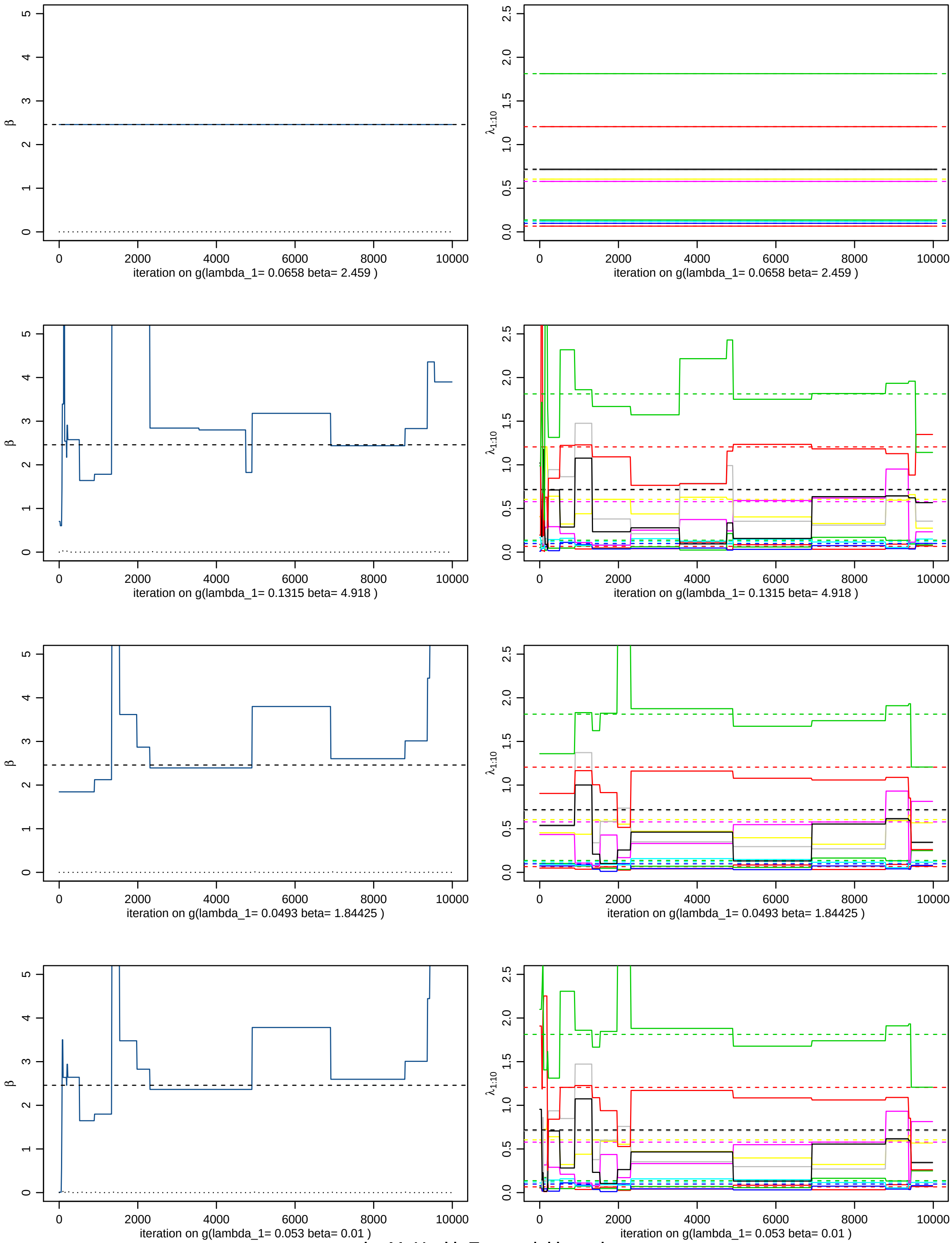
Comparing Gibbs and M−H with Uniform kernel

The kenel of Exponential version

```
MH_E <- function(lambda_e,beta_e,S){ # Expo version
Theta<-matrix(NA,S,12) ; acr <- acs<-0 ;  set.seed(121)
lambda <- lambda_e; beta <-beta_e
g <- function(a,b){
d.lambda<- dexp(a,rate=(b+t)/(d+alpha)) # 1/lambda_true
d.beta <-dexp(b,rate=(delta+sum(lambda_e))/(gamma+n*alpha))     #    1/b
return(prod(d.lambda)*d.beta)
}
for(s in 1:S) {
lambda.star <- rexp(n,(beta+t)/(d+alpha)) # sample lambda  & lambda.star<=1
beta.star <- rexp(1,(delta+sum(lambda))/(gamma+n*alpha))  # sample beta
r<- pi(lambda.star,beta.star)/pi(lambda,beta)*g(lambda,beta)/g(lambda.star,beta.star)
  if((runif(1))<g(lambda.star,beta.star)*min(r,1)) {lambda<-lambda.star; beta<-beta.star ; acs<-acs+1}
    if(s%%50==0) {acr <- acs/100; acs<-0}
  Theta[s,]<-c(beta,lambda,acr)
}
return(Theta)
}
```



the M−H with Expoential kernel

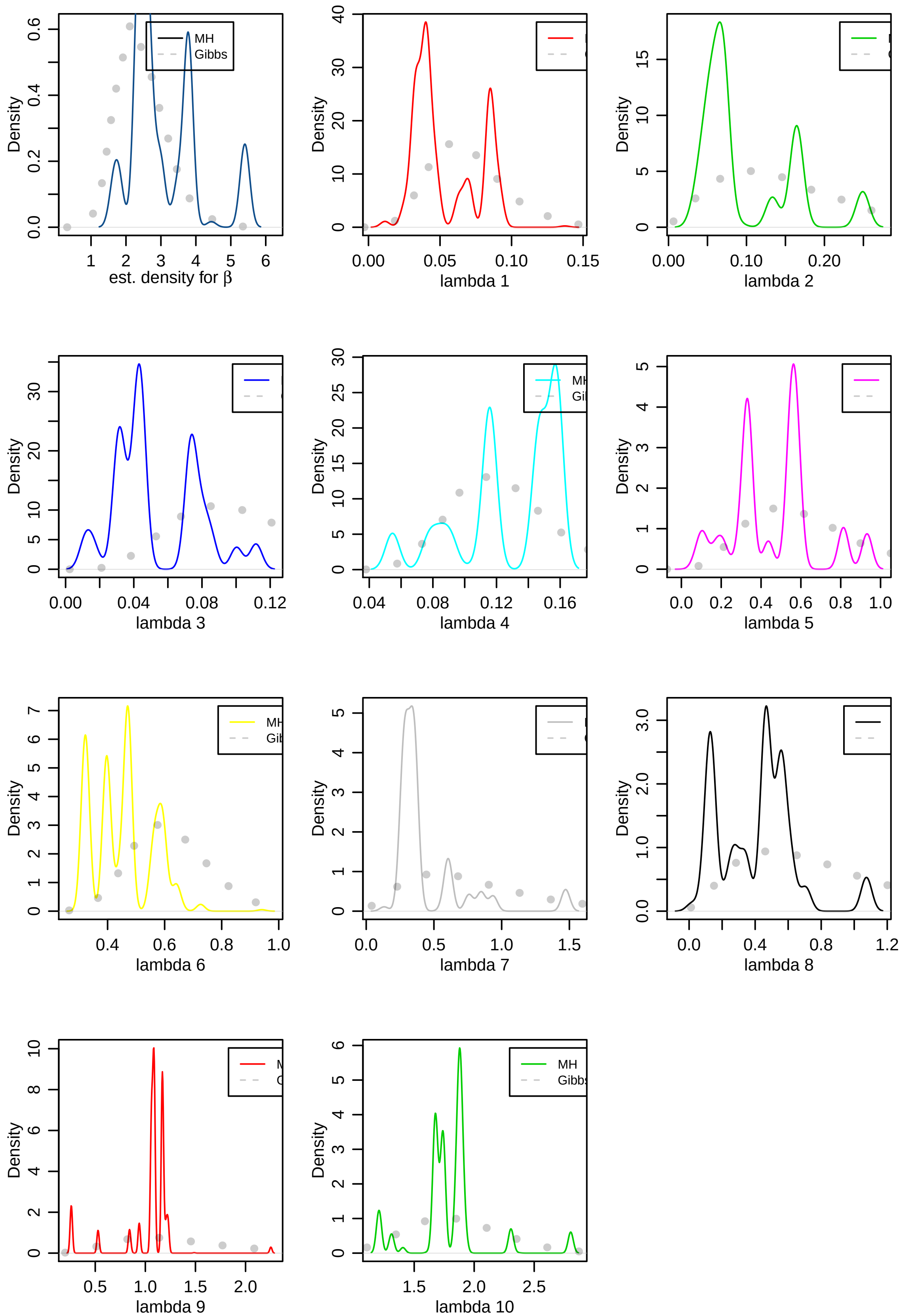


the M−H with Expoential kernel

	mean	2.5%	50%	97.5%
Beta	3.438	2.596	3.783	5.391
Lambda1	0.0636	0.0327	0.0698	0.0911
Lambda2	0.1288	0.0572	0.1332	0.2491
Lambda3	0.0539	0.0313	0.0402	0.0799
Lambda4	0.1207	0.0546	0.1155	0.1474
Lambda5	0.6293	0.549	0.5783	0.9322
Lambda6	0.4138	0.3223	0.397	0.5928
Lambda7	0.3325	0.2703	0.2976	0.609
Lambda8	0.3775	0.1321	0.5558	0.6171
Lambda9	0.9792	0.261	1.061	1.09
Lambda10	1.677	1.207	1.74	1.911
Acceptance Rate	0.0004	0	0	0.01

```r
ptm <- proc.time()
for(k in 1:K) { # Run K Number of chains by drawing from Uniform
THETA_e[k,] <- MH_E(lambda_e[,4],beta_e[4],S)[S,]
}
ptm_e <- proc.time() - ptm
```

- Comparing with Gibbs

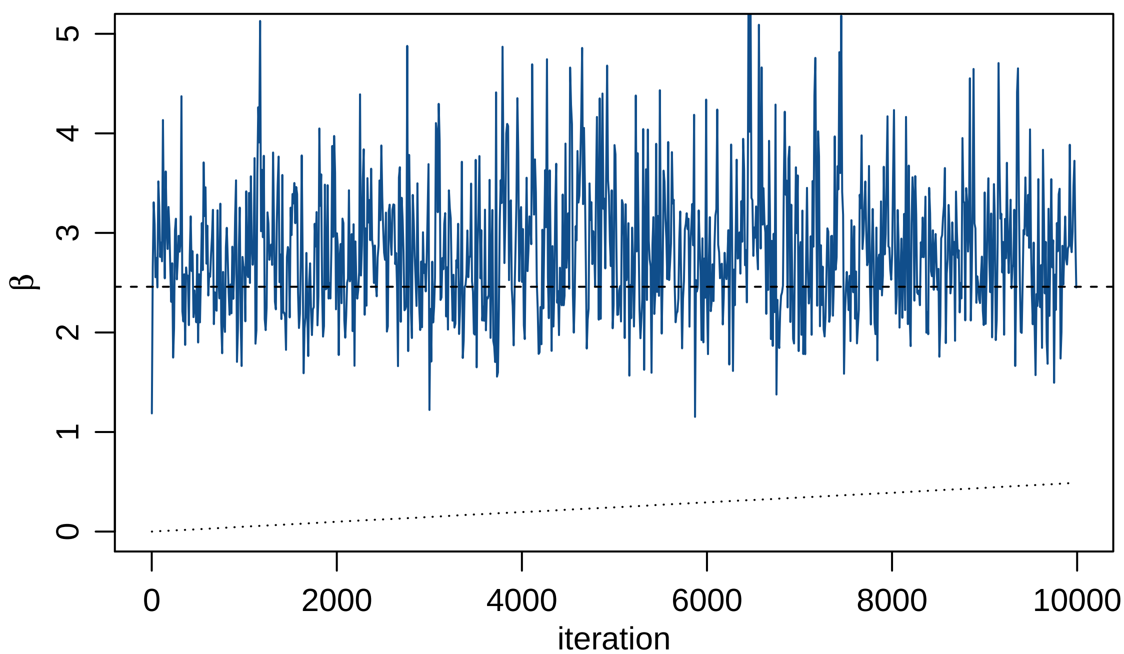


Comparing Gibbs and M−H with Uniform kernel

The Normal version of $g(.)$

The version of $g(.) = 1$

```
#g_b <- function(x){dbeta(x,beta0,5)}
MH_1 <- function(S){
Theta<-matrix(NA,S,12) ; acr <- acs<-0 ; # set.seed(121)
# g <- function(lambda,beta){1}
for(s in 1:S) {
lambda.star <- rgamma(n,shape=(d+alpha),rate=(beta+t)) # sample lambda
r_lambda <- pi(lambda.star,beta)/pi(lambda,beta)  #  *g(lambda,beta)/g(lambda.star,beta)
  if((runif(1))<r_lambda) { lambda<-lambda.star; acs<-acs+1 }
beta.star <- rgamma(1,shape=(gamma+n*alpha),rate=(delta+sum(lambda)))  # sample beta
r_beta<- pi(lambda,beta.star)/pi(lambda,beta)  #  *g(lambda,beta)/g(lambda,beta.star)
  if((runif(1))<r_beta) { beta<-beta.star ; acs<-acs+1}
  acr <- acs/S/2
  Theta[s,]<-c(beta,lambda,acr)
}
return(Theta)
}
```

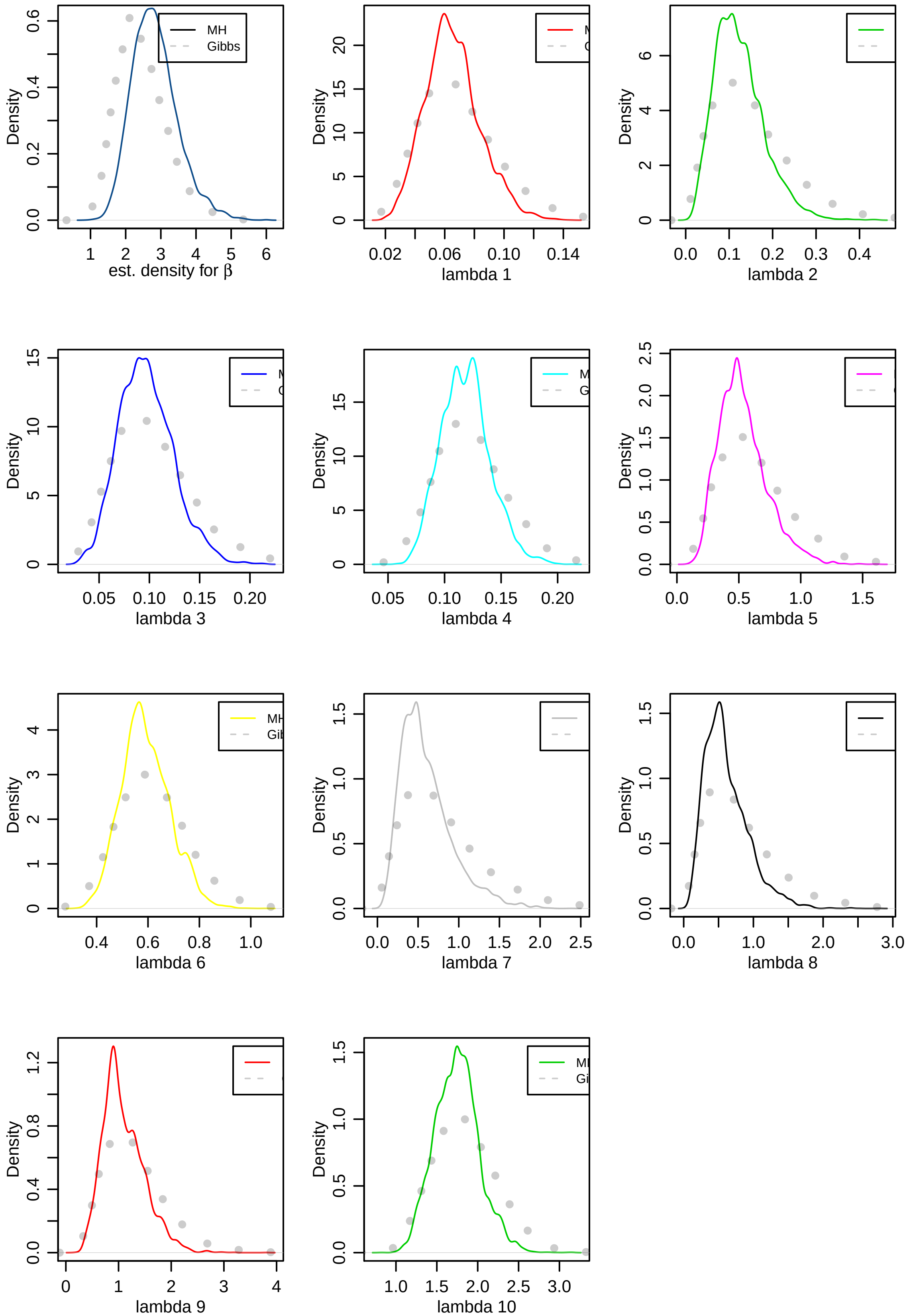


the M−H with kernel g(.)=1

	mean	2.5%	50%	97.5%
Beta	2.835	1.768	2.771	4.357
Lambda1	0.0654	0.0328	0.0634	0.1044
Lambda2	0.1217	0.0358	0.1125	0.2571
Lambda3	0.0965	0.0497	0.0948	0.1545
Lambda4	0.1192	0.0824	0.118	0.1602
Lambda5	0.5391	0.2419	0.5168	0.9936
Lambda6	0.5909	0.421	0.582	0.7915

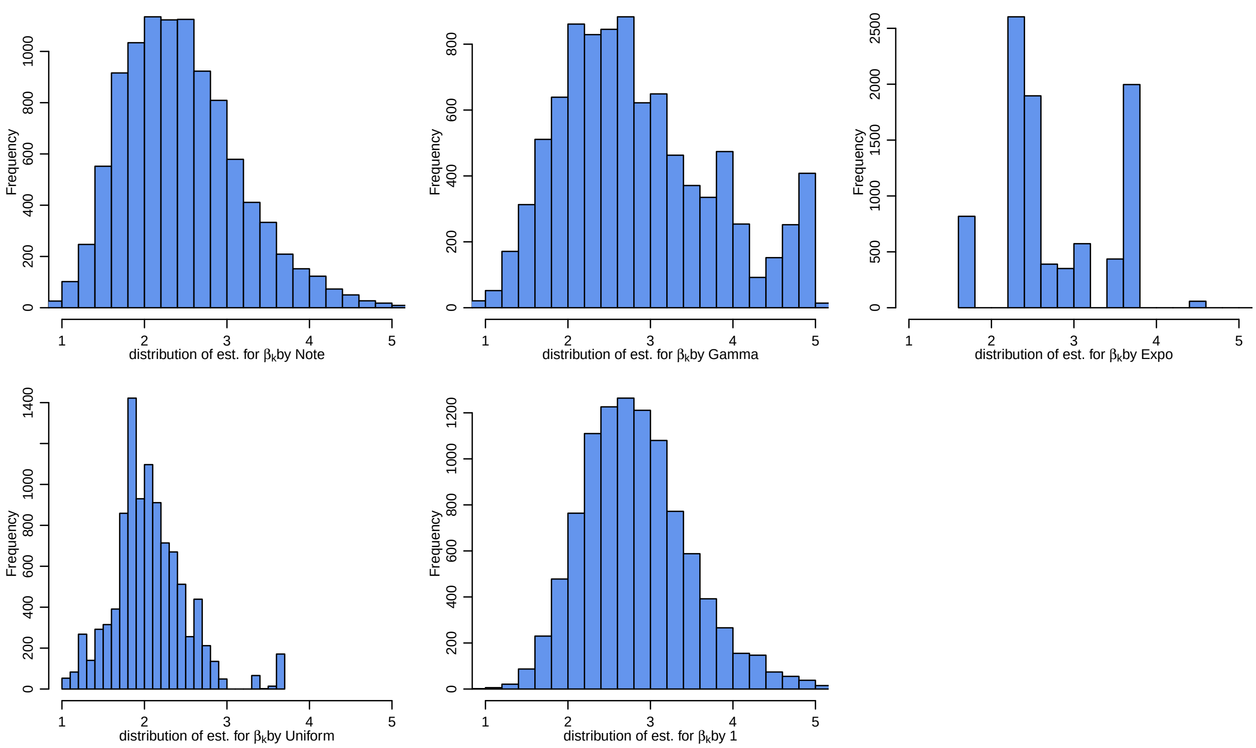
	mean	2.5%	50%	97.5%
Lambda7	0.6172	0.1665	0.5539	1.431
Lambda8	0.6061	0.1757	0.5565	1.389
Lambda9	1.09	0.4597	1.034	1.929
Lambda10	1.749	1.261	1.744	2.29

```
ptm <- proc.time()
for(k in 1:K) { # Run K Number of chains by drawing from Uniform
THETA_1[k,] <- MH_1(S)[S,]
}
ptm_1 <- proc.time() - ptm
```



Comparing Gibbs and M−H with kernel g(.)=1

Comparison histogram



Comparison table

```r
library(HDInterval)
g0 <- c(mean(THETA_0[,1]),quantile(THETA_0[,1],c(0.025,0.5,0.975)),hdi(THETA_0[,1], credMass=0.95),mean
gg <- c(mean(THETA_g[,1]),quantile(THETA_g[,1],c(0.025,0.5,0.975)),hdi(THETA_g[,1], credMass=0.95),mean
ge <- c(mean(THETA_e[,1]),quantile(THETA_e[,1],c(0.025,0.5,0.975)),hdi(THETA_e[,1], credMass=0.95),mean
# gn <- c(mean(THETA_n[,1]),quantile(THETA_n[,1],c(0.025,0.5,0.975)),hdi(THETA_n[,1], credMass=0.95),me
gu <- c(mean(THETA_u[,1]),quantile(THETA_u[,1],c(0.025,0.5,0.975)),hdi(THETA_u[,1], credMass=0.95),mean
g1 <- c(mean(THETA_1[,1]),quantile(THETA_1[,1],c(0.025,0.5,0.975)),hdi(THETA_1[,1], credMass=0.95),mean
pumps.par <- rbind(g0,gg,ge,gu,g1) # gn,
colnames(pumps.par) <- c('mean','2.5%','median','97.5%','95% HPD U', '95% HPD L','Acceptance Rate','runn
kableExtra::kable(round(pumps.par,4))
```

	mean	2.5%	median	97.5%	95% HPD U	95% HPD L	Acceptance Rate	running time
g0	2.453	1.326	2.375	4.121	1.168	3.872	0.9872	58.59
gg	3.061	1.403	2.770	5.493	1.437	5.493	0.5999	142.94
ge	3.031	1.647	2.596	5.391	1.798	5.441	0.0009	78.46
gu	2.062	1.257	2.007	3.309	1.180	2.788	0.0041	136.54
g1	2.829	1.735	2.770	4.341	1.556	4.104	0.2494	85.31