Posterior with vague Prior

Binary parametric regression

Consider $y_{i,j} \sim Bernoulli\left(\frac{\theta x_i}{1+\theta x_i}\right)$, which is of the form $y_{i,j} \sim Bernoulli\left(H(\theta x_i)\right)$ with $H(t) = \frac{t}{1+t}$ being a known, increasing, Lipschitz continuous, cumulative distribution function for positive support of both θ and x_i. Here, the prior for $\tilde{x}_i$ is the uniform distribution on

Here we investigate consistency of the posterior of $\tilde{x}_i$. We generate the data by simulating $\theta \sim unif(1,2)$ and $x_i \sim unif(1,3)$ for $i \in \{1, \cdots, n\}$ and then by generating $y_{i,j} \sim Bernoulli\left(\frac{\theta x_i}{1+\theta x_i}\right)$ for $i \in \{1, 2, \cdots, n\}$ and $j \in \{1, 2, \cdots, m\}$.

We will run a simple metropolis sampling algorithm of mcmc size=10000 using uniform prior over $(0, \infty)$, which is a vague prior/non informative prior. **Surprisingly, we obtain convergence in this scenerio also.**

```
rm(list=ls())
require(MASS)
```

```
## Loading required package: MASS
```

```
seed = 100
set.seed(seed)
```

```
theta = runif(1,0.5,2)
th0=theta
#x0=x[1]
M=1000
N=1000

X = runif(N,1,2);
x0=X[1];
Y = matrix(nrow=N,ncol=M)
H=function(t){return(t/(1+t))}
for( i in 1:N )
{ lambda = th0*X[i]
Y[i,] = rbinom(M,1,H(lambda))
}
```

```
c = 15
```

```
fratio <- function(t1, t2){
  th1=t1[1];
  vecx1=t1[2];

  th2=t2[1];
  vecx2=t2[2];
 f11=((m*mean(y[1,])*log((th1*vecx1)/(th2*vecx2)))-(m*log((1+th1*vecx1)/(1+th2*vecx2))))
```

```
#  print(f11)
     sum2= 0
     for(h in 2:n)
     {sum2=sum2+(m*mean(y[h,])*log(th1/th2))-(m*log((1+th1*x[h])/(1+th2*x[h])))
     }
     f22= sum2
   #print(f22)
   return(exp(f11+f22))
}

ff <- function(t){
  th=t[1];
  vecx=t[2];


  f1=((m*mean(y[1,])*log((th*vecx)))-(m*log((1+th*vecx))))

  #print(f1)
  sum1= 0
  for(h in 2:n)
  {sum1=sum1+(m*mean(y[h,])*log((th*vecx)))-(m*log((1+th*vecx)))
  }
  f2= sum1
  print(f2)
  return(exp(f1+f2))
}


#print(fratio(c(2,3),c(1.5,1.4)))
#print(ff(c(2,3)))
#print(fratio(c(7,3),c(28,3)))
```

```
q <- function(t) {
  e = abs(rnorm(1,0,1))
  th=t[1]
  x_st=t[2]
  #rnorm(1, x, 0.1)
  u = runif(1,0,1)
  if( u<0.5 )
  { new_x_st = x_st+0.5*e
  }
  else
  { new_x_st = x_st-0.5*e
  }
  if(new_x_st<0.0001)
  {
    new_x_st=x_st;
  }
  #new_eta[1] = rnorm(1,0,1)+new_b*new_x_star

   v = runif(1,0,1)
  if( v<0.5 )
  { new_th = th +0.05*e
  }
```

```
  else
  { new_th =  th-0.05*e
  }
   if(new_th<0.0001)
   {
     new_th=th;
   }
   g=c(new_th, new_x_st)
  return(g)

  }
#q2 <- function(x) rnorm(1, x, 0.08)
```

```
step <- function(t, q) {

  ## Pick new point
  tp <- q(t)

    ## Acceptance probability:
  alpha <- min(1, fratio(tp,t))
  ## Accept new point with probability alpha:
  if (runif(1) < alpha)
    t <- tp
  ## Returning the point:
  return(t)
}

#step(c(1.5,1.5),q)
```

```
#mcmc_size=10000
run <- function(t, q, nsteps) {
  mcmc_size=nsteps;
  res <- matrix(NA, nsteps, length(t))
  ptm <- proc.time()
  for (i in seq_len(nsteps)){
    res[i,] <- t <- step(t, q)

   # print(i)
   # if (i == nsteps) cat(': Done')
    # else cat('\014')

     if(i==10|i==100|i==500|i==1000|i==3000|i==5000|i==7000|i==9000|i==10000)
     #{print(i)}
   # progress(i,progress.bar = T)
   { cat(paste0('current sample:[', i,']   mcmc_run: ', round(i/ (mcmc_size-1) * 100), '% completed'))
         print("**")
   }

   if (i == nsteps)
     { print("***");
         cat(': Done :');
     print("***");

    # else cat('\014')
```

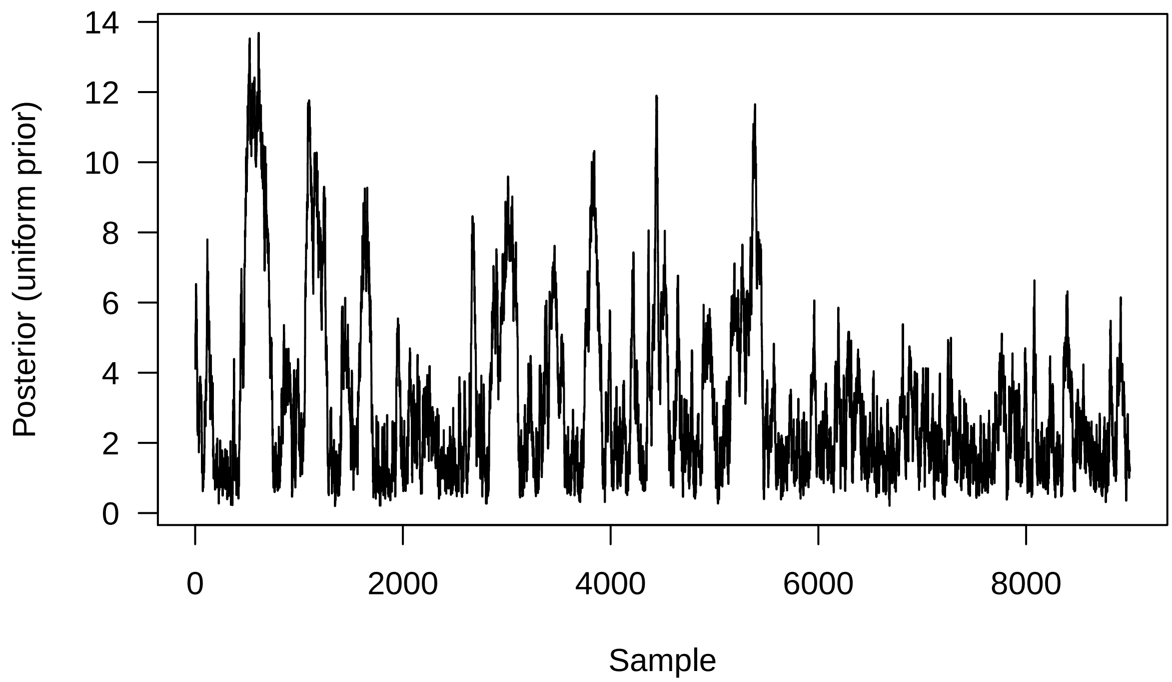
```
  }
  }
   print(proc.time()-ptm)
  drop(res)

}
```

```
m=10
n=10
x=c()
for(i in 1:n)
{x[i]=X[i]
}
y = matrix(nrow=n,ncol=m);

for(i in 1:n)
{
  for(j in 1:m)
  {
    y[i,j]=Y[i,j]
  }
}
k = 1
y_bar = mean(y[k,])
y_std = sd(y[k,])
x_k = sum(x)-x[k]

#a = sum(y)+1

ress_10 <- run(c(1.2,1.2),  q, 10000)
```

```
## current sample:[10]   mcmc_run: 0% completed[1] "**"
## current sample:[100]   mcmc_run: 1% completed[1] "**"
## current sample:[500]   mcmc_run: 5% completed[1] "**"
## current sample:[1000]   mcmc_run: 10% completed[1] "**"
## current sample:[3000]   mcmc_run: 30% completed[1] "**"
## current sample:[5000]   mcmc_run: 50% completed[1] "**"
## current sample:[7000]   mcmc_run: 70% completed[1] "**"
## current sample:[9000]   mcmc_run: 90% completed[1] "**"
## current sample:[10000]   mcmc_run: 100% completed[1] "**"
## [1] "***"
## : Done :[1] "***"
##    user  system elapsed
##    0.78    0.00    0.78
```

```
#ress_10[1:100,]
```

```
#resss<- run(c(1.5,1.5),  q, 100000)
plot(ress_10[1001:10000,2], type="s", xpd=NA, ylab="Posterior (uniform prior)", xlab="Sample", las=1)
```

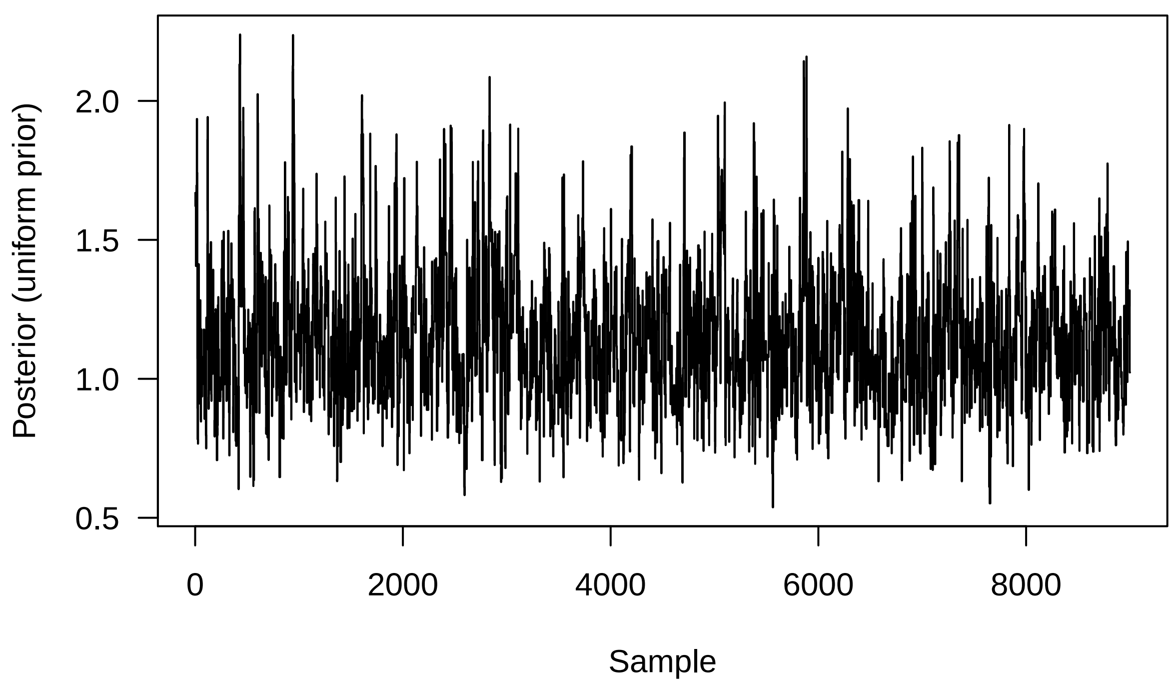



```
m=100
n=100
x=c()
for(i in 1:n)
{x[i]=X[i]
}
y = matrix(nrow=n,ncol=m);

for(i in 1:n)
{
  for(j in 1:m)
  {
    y[i,j]=Y[i,j]
  }
}
k = 1
y_bar = mean(y[k,])
y_std = sd(y[k,])
x_k = sum(x)-x[k]

#a = sum(y)+1

ress_100<- run(c(1.2,1.2),  q, 10000)
```

```
## current sample:[10]   mcmc_run: 0% completed[1] "**"
## current sample:[100]   mcmc_run: 1% completed[1] "**"
```

```
## current sample:[500]   mcmc_run: 5% completed[1] "**"
## current sample:[1000]   mcmc_run: 10% completed[1] "**"
## current sample:[3000]   mcmc_run: 30% completed[1] "**"
## current sample:[5000]   mcmc_run: 50% completed[1] "**"
## current sample:[7000]   mcmc_run: 70% completed[1] "**"
## current sample:[9000]   mcmc_run: 90% completed[1] "**"
## current sample:[10000]   mcmc_run: 100% completed[1] "**"
## [1] "***"
## : Done :[1] "***"
##    user  system elapsed
##    6.83    0.00    6.86
```

```
#resss<- run(c(1.5,1.5),  q, 100000)
plot(ress_100[1001:10000,2], type="s", xpd=NA, ylab="Posterior (uniform prior)", xlab="Sample", las=1)
```

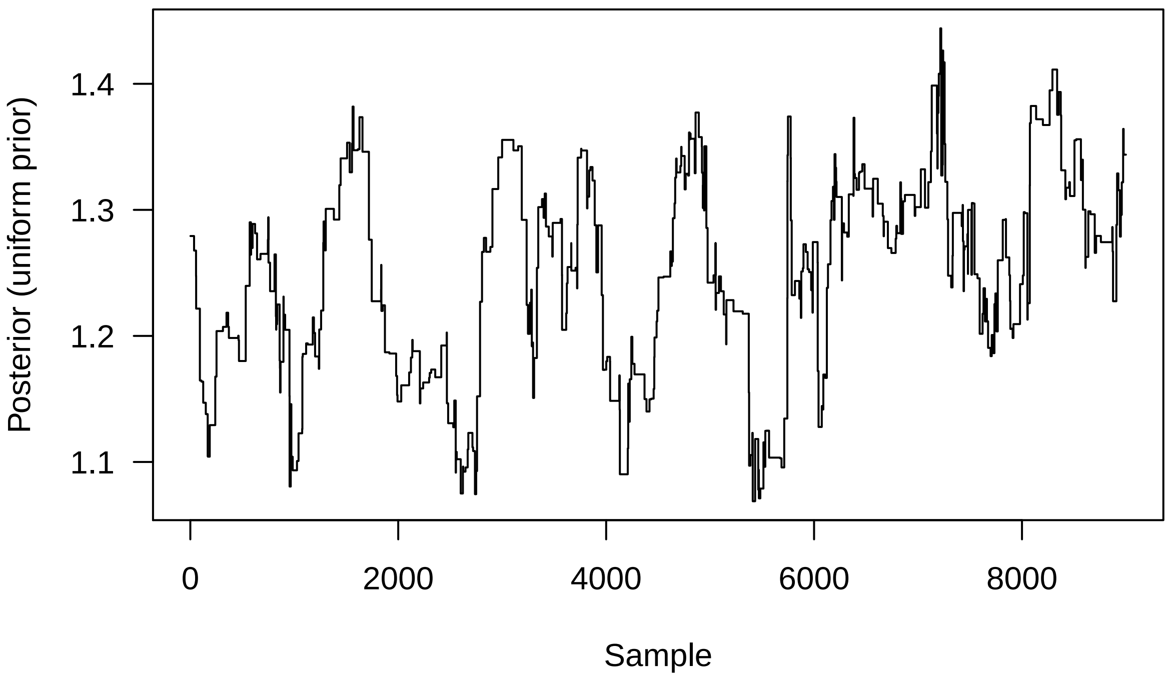



```
m=1000
n=1000
x=c()
for(i in 1:n)
{x[i]=X[i]
}
y = matrix(nrow=n,ncol=m);

for(i in 1:n)
{
  for(j in 1:m)
  {
```

```r
    y[i,j]=Y[i,j]
  }
}
k = 1
y_bar = mean(y[k,])
y_std = sd(y[k,])
x_k = sum(x)-x[k]

#a = sum(y)+1

ress_1000 <- run(c(1.2,1.2),  q, 10000)
```

```
## current sample:[10]   mcmc_run: 0% completed[1] "**"
## current sample:[100]   mcmc_run: 1% completed[1] "**"
## current sample:[500]   mcmc_run: 5% completed[1] "**"
## current sample:[1000]   mcmc_run: 10% completed[1] "**"
## current sample:[3000]   mcmc_run: 30% completed[1] "**"
## current sample:[5000]   mcmc_run: 50% completed[1] "**"
## current sample:[7000]   mcmc_run: 70% completed[1] "**"
## current sample:[9000]   mcmc_run: 90% completed[1] "**"
## current sample:[10000]   mcmc_run: 100% completed[1] "**"
## [1] "***"
## : Done :[1] "***"
##    user  system elapsed
##  146.52    0.05  146.94
```

```r
#resss<- run(c(1.5,1.5),  q, 100000)
plot(ress_1000[1001:10000,2], type="s", xpd=NA, ylab="Posterior (uniform prior)", xlab="Sample", las=1)
```




```r
plot(density(ress_10[1001:10000,2]), xlim=c(0,8),ylim=c(0,5),col="red",ylab="density (with vague prior)
#hist(resss[,1])
abline(v=X[1],add=T, col="black",lwd=2, lty=1)
```

```
## Warning in int_abline(a = a, b = b, h = h, v = v, untf = untf, ...): "add"
## is not a graphical parameter
```

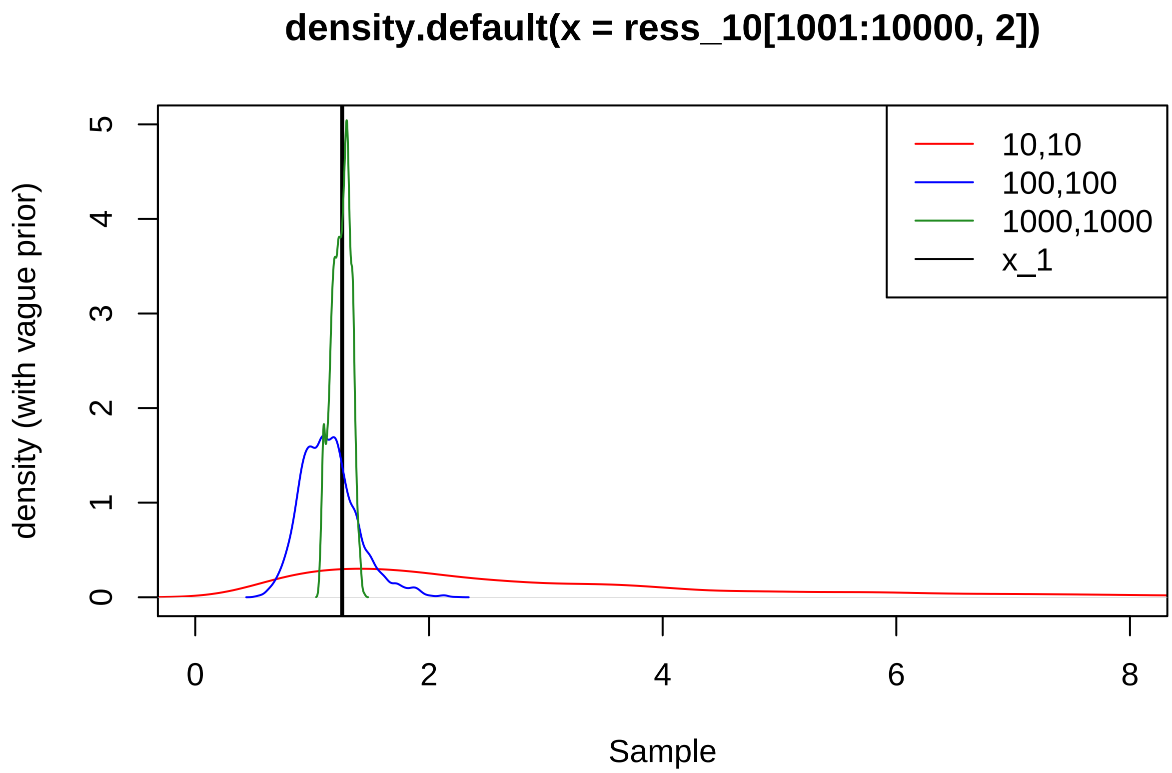
```r
lines(density(ress_100[1001:10000,2]), xlim=c(0,8),ylim=c(0,5),col="blue", add=T)
```

```
## Warning in plot.xy(xy.coords(x, y), type = type, ...): "add" is not a
## graphical parameter
```

```r
lines(density(ress_1000[1001:10000,2]), xlim=c(0,8),ylim=c(0,5),col="forestgreen", add=T)
```

```
## Warning in plot.xy(xy.coords(x, y), type = type, ...): "add" is not a
## graphical parameter
```

```r
legend("topright", c("10,10","100,100","1000,1000","x_1"), lty = c(1,1,1,1), col = c("red","blue","fores
```


density.default(x = ress_10[1001:10000, 2])
density (with vague prior)
Sample
0
1
2
3
4
5
0
2
4
6
8
10,10
100,100
1000,1000
x_1