Stat651 Mini Project 3

Arthur Lui

24 October 2014

1 Boostrap Importance Sampling

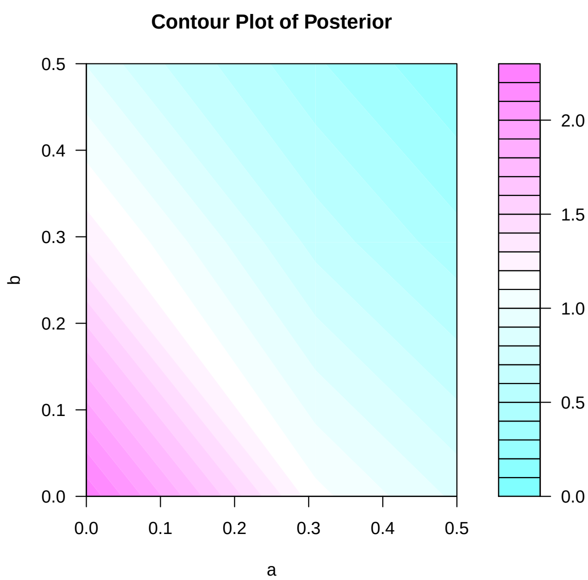
1.1 Importance Function

The importance function $I(\Theta)$ I chose was the envelop function:

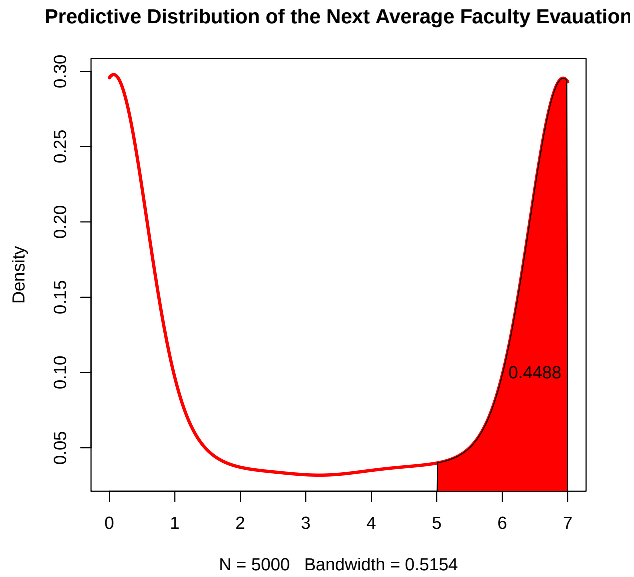
$$I(\Theta) = p(\theta_1, .5, 1)p(\theta_2, 5, 1)$$

where $p(x, a, b) = \frac{1}{\Gamma(a)\Gamma(b)} x^{a-1} e^{-x/b}$. The importance function I chose is easy to sample from and mirrors the sampling distribution well. It preserves the parameter space because a and b are positive. Also, the gamma distribution has long tails.

1.2 Posterior - Importance Sampling

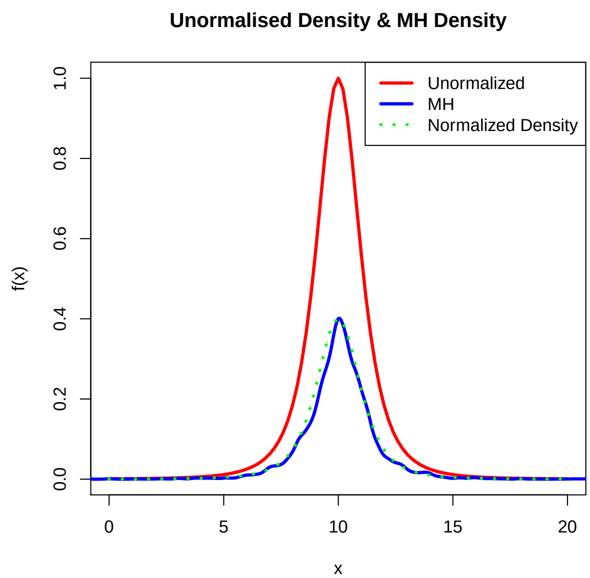


$$\begin{aligned}
E[\Theta|\boldsymbol{Y}] &= (0.5129, 0.5158) \\
V[\Theta|\boldsymbol{Y}] &= (0.5149, 0.5226) \\
SD[\Theta|\boldsymbol{Y}] &= (0.7176, 0.7229) \\
P[Y_{n+1} > 5|\boldsymbol{Y}_n] &= 0.4488
\end{aligned}$$



2 Metropolis Hastings

2.1 Plot of Unormalized Density & MH Density



3 Metropolis Hastings & Gibbs

3.1 Posterior Mean

$E[\Theta|\boldsymbol{Y}] =$

	Gamma	Beta
1	1.65	3009.85

3.2 Posterior Variance

$V[\Theta|\boldsymbol{Y}] =$

	Gamma	Beta
Gamma	0.07	1076.94
Beta	1076.94	38780543.96

3.3 97% HPD

	HPD Lower	HPD Upper
gamma: 0%-97%	0.00	2.68
beta: 0.03%-97.03%	68.87	220511.16

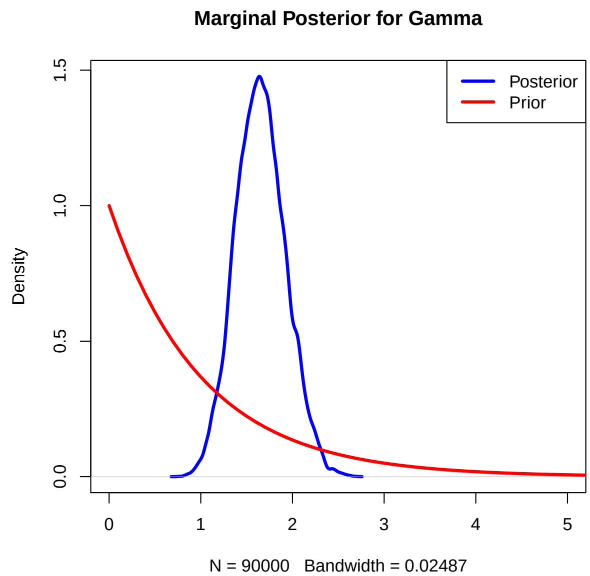
3.4 Posterior Probability that Next Ball Bearing will be Superior

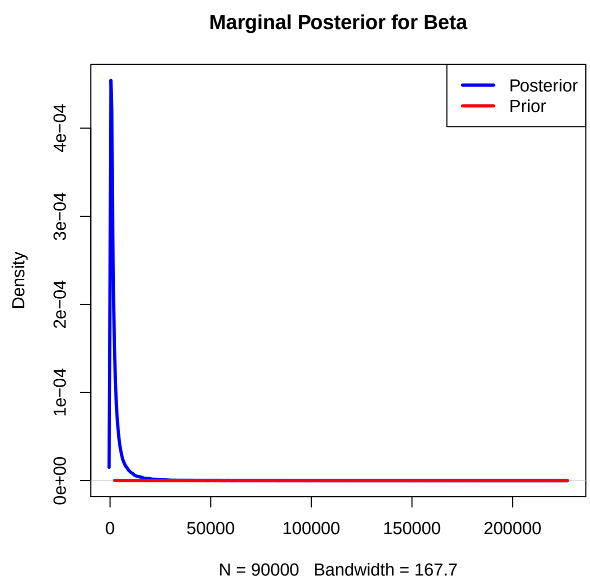
$P[Y_{n+1} > 100|\boldsymbol{Y}_n] = 0.2212$

3.5 MCMC Settings

Length $= 1e + 05$
Burn $= 10000$
$\sigma_{cand} = 0.1$
Candidate Density for x_i= Normal(x_{i-1},σ_{cand})

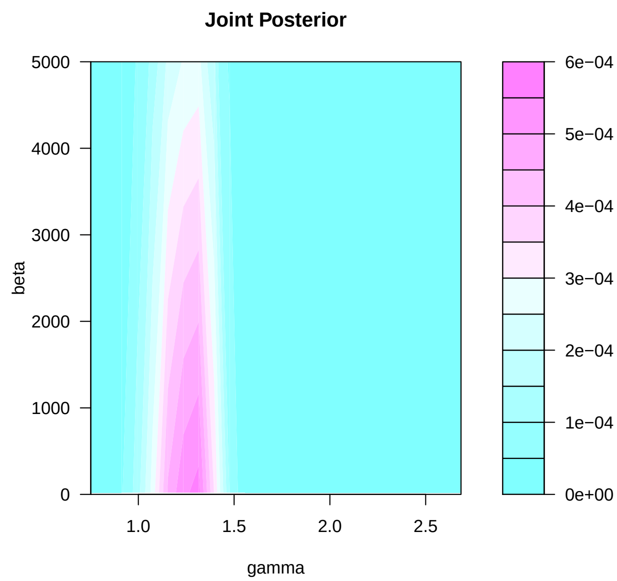
3.6 Plots of Posteriors

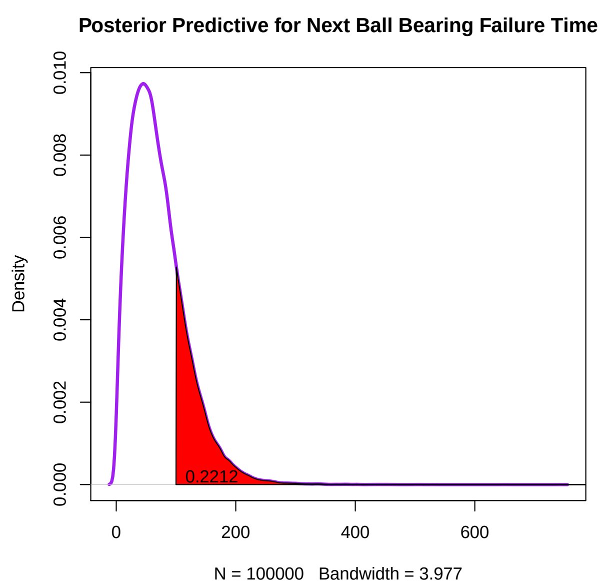




4 Appendix - Code

```
library(xtable)
pb <- function(i,n) {
```


Joint Posterior
5000
4000
3000
2000
1000
0
beta
1.0
1.5
2.0
2.5
gamma
6e−04
5e−04
4e−04
3e−04
2e−04
1e−04
0e+00


Posterior Predictive for Next Ball Bearing Failure Time
0.010
0.008
0.006
0.004
0.002
0.000
Density
0.2212
0
200
400
600
N = 100000 Bandwidth = 3.977

```
  cat(paste0("\rProgress: ",round(i*1000/n)/10,"%"))
  if (i==n) cat("\n")
}

#1: Do last
fac <- as.vector(as.matrix(read.table("faculty.dat")))
y <- fac/7

l <- function(x=y,a=.5,b=.5,log=F) {
  if (!log) {
    prod( dbeta(x,a,b) )
  } else {
    sum( dbeta(x,a,b,log=T) )
  }
}

g <- function(x,log=F) {
  a <- x[1]
  b <- x[2]
  if (!log) {
    l(y,a,b) * dgamma(a,.5,1) * dgamma(b,.5,1)
  } else {
    l(y,a,b,log=T) + dgamma(a,.5,1,log=T) + dgamma(b,.5,1,log=T)
  }
}


imp.boot <- function(B=5000) {
  x <- rgamma(2*B,.5,1)
  X <- matrix(x,B,2)

  I <- function(x) {
    a <- x[1]; b <- x[2]
    dgamma(a,.5,1) * dgamma(b,.5,1)
  }

  num <- apply(X,1,function(x) g(x) / I(x))
  denom <- sum(num)

  q <- num/denom

  ind <- sample(1:B,B,replace=T)

  M <- apply(matrix(ind),1,function(i) X[i,])
  t(M)
}

W <- imp.boot()

library(MASS)
plot.1b <- function() {
```

```
  J <- kde2d(W[,1],W[,2])
  filled.contour(J,ylim=c(0,.5),xlim=c(0,.5),main="Contour Plot of Posterior",
                 xlab="a",ylab="b")
}

#plot(density(M[,1]),col="blue",lwd=3)
#lines(density(M[,2]),col="red",lwd=3,lty=3)

imp.mean <- apply(W,2,mean)
imp.var <- apply(W,2,var)
imp.sd <- apply(W,2,sd)

source("color.R")
post.pred.1 <- 7*rbeta(nrow(W),W[,1],W[,2])
prob.1 <- mean(post.pred.1>5)

#pdf("latex/postpred1.pdf")
plot.pred.1 <- function() {
  den.1 <- density(post.pred.1)
  plot(density(post.pred.1,from=0,to=7),col="red",lwd=3,main="Predictive Distribution of the Next Averag
  color.den(den.1,5,7)
  #abline(v=mean(post.pred),col="orange",lwd=3)
  text(6.5,.1,prob.1)
}
#plot.pred.1()
#dev.off()

#2
find.mode.den <- function(samp) {
  d <- density(samp)

  x <- d$x[which.max(d$y)]
  y <- max(d$y)

  c(x,y)
}

find.den.height <- function(x,den) {
  den$y[which.min(abs(den$x-x))]
}

f <- function(x,log=F) { # -Inf < x < Inf
  if (!log) {
    (1+(x-10)^2/3)^-2
  } else {
    -2 * log(1+(x-10)^2/3)
  }
}

mh.2 <- function(cs=1,M=1e4,burn=M*.1) {
  out <- NULL
```

```
  out[1] <- 0
  log.u <- log(runif(M))
  acc <- 0

  for (i in 2:M) {
    #pb(i,M)
    out[i] <- out[i-1]
    cand <- rnorm(1,out[i],cs)

    q <- f(cand,log=T)  - f(out[i],log=T)
    if (q > log.u[i]) {
      out[i] <- cand
      if (i > burn) acc <- acc + 1
    }
  }

  #print(acc/(M-burn))
  out
}

x <- mh.2(5,M=1e4)
plot.2 <- function(){
  curve(f(x),0,20,col="red",lwd=3,main="Unormalised Density & MH Density")
  lines(density(x),col="blue",lwd=3)
  mode <- find.mode.den(x)
  c.hat <- mode[2] / f(10)
  f2 <- function(x,...) f(x,...) * c.hat
  curve(f2(x),0,20,col="green",lwd=3,add=T,lty=3)
  legend("topright",legend=c("Unormalized","MH","Normalized Density"),col=c("red","blue","green"),lwd=3,
}

# To get c.hat:
#den <- density(x)
#c.hats <- apply(matrix(seq(5,15,length=1e4)),1,function (x) find.den.height(x,den))
#mean.c.hat <- mean(c.hats)


#3
# average ~ (50,70)
# bb > 10
bb <- read.table("ballbearing2.dat")
bb <- as.vector(as.matrix(bb))

# Priors parameters
a <- 1
b <- 1
c <- 1
d <- 1
n <- length(bb)

ding <- function(x,a,b) { # density of inverese gamma
```

```
  b^a / gamma(a) * x^(-a-1) * exp(-b/x)
}
ring <- function(n,a,b) { # random draws from inverse gamma
  1/rgamma(n,a,b)
}

#curve(ding(x,4,7),from=0,to=20,ylim=c(0,2))
#lines(density(ring(1e4,4,7)),col="blue")

po.be <- function(be,ga,log=T) {
  if (!log) {
    ding(be,n+c,sum(bb^ga)+d)
  } else {
    A <- n+c
    B <- sum(bb^ga)+d

    A*log(B) - lgamma(A) -(A+1)*sum(log(bb))
  }
}

update.be <- function(ga) {
  p <- c(n+c,sum(bb^ga)+d)
  ring(1,p[1],p[2])
}

lg.ga <- function(be,ga) {
  (a+n-1)  * log(ga) + (ga-1) * sum(log(bb)) - ga/b - sum(bb^ga) / be
}

mh.gibbs <- function(csig=2,B=1e4,burn=.1*B) {
  acc <- 0
  M <- matrix(0,B,2)
  M[1,] <- c(1,1)
  lu <- log(runif(B))

  for (i in 2:B) {
    #pb(i,B)
    M[i,] <- M[i-1,] # Gamma, Beta

    # Update Gamma (MH)
    #cand <- rgamma(1,x/csig,scale=csig)
    #rat <- lg.ga(M[i,2],cand) - lg.ga(M[i,2],x) + dgamma(x,cand/csig,scale=csig)- dgamma(cand,x/csig,sc
    x <- M[i,1]
    cand <- rnorm(1,x,csig)
    if (cand>0) {
      rat <- lg.ga(M[i,2],cand) - lg.ga(M[i,2],x)

      if (rat>lu[i]) {
        M[i,1] <- cand
        acc <- acc + 1
      }
```

```
    }

    # Update Beta (Gibbs)
    M[i,2] <- update.be(M[i,1])
  }

  #print(acc/B)
  M[-c(1:burn),]
}

B <- 1e5
M <- mh.gibbs(csig=.1,B=B)

# Trace Plots
plot(M[,1],type="l")
plot(M[,2],type="l")

#3a:
post.mean.3 <- matrix(apply(M,2,mean),1)
post.var.3 <- var(M)

colnames(post.mean.3) <-  c("Gamma","Beta")
rownames(post.mean.3) <- ""
colnames(post.var.3) <- rownames(post.var.3) <- c("Gamma","Beta")


#3b:
# Marginal Densities
plot.ga <- function() {
  plot(density(M[,1]),main="Marginal Posterior for Gamma",col="blue",lwd=3,xlim=c(0,5))
  curve(dgamma(x,a,scale=b),from=0,to=10,add=T,col="red",lwd=3)
  legend("topright",legend=c("Posterior","Prior"),col=c("blue","red"),lwd=3)
}

plot.be <- function() {
  plot(density(M[,2]),col="blue",lwd=3,main="Marginal Posterior for Beta")
  curve(ding(x,c,d),from=0,add=T,col="red",lwd=3)
  legend("topright",legend=c("Posterior","Prior"),col=c("blue","red"),lwd=3)
}

K <- kde2d(M[,1],M[,2]) # Gamma, Beta
#library(rgl)
#persp3d(K,col="yellow") #rgl
plot.joint <- function() {
  filled.contour(K,ylim=c(0,5000),main="Joint Posterior",xlab="gamma",ylab="beta")
}

#3c:
post.pred <- rweibull(B,M[,1],M[,2]^(1/M[,1]))
prob.3e <- mean(post.pred>100)
```

```
plot.post.pred <- function() {
  den.3 <- density(post.pred)
  plot(den.3,col="purple",lwd=3,
               main="Posterior Predictive for Next Ball Bearing Failure Time")
  color.den(den.3,from=100,to=1074)
  text(160,.0002,prob.3e)
}

#3d:
find.hpd <- function(den,name="",cover=.97,prec=10000) {

  n <- length(den$x)
  find.dis <- function(p) {
    qq <- quantile(den$x,c(p,p+cover))
    d <- qq[2] - qq[1]
    c(d,qq)
  }

  xx <- seq(0,1-cover,length=prec)
  D <- t(apply(matrix(xx),1,find.dis))
  ind <- which.min(D[,1])
  p <- xx[ind]

  lo.perc <- p
  up.perc <- p + cover

  out <- matrix(c(D[ind,2:3]),1)
  rownames(out) <- paste0(name,": ",round(lo.perc*100,2),"%","-",round(up.perc*100,2),"%")
  colnames(out) <- c("HPD Lower","HPD Upper")
  out
}

ga.hpd <- find.hpd(density(M[,1],from=0),name="gamma",prec=100)
be.hpd <- find.hpd(density(M[,2],from=0),name="beta",prec=100)
A3d <- rbind(ga.hpd,be.hpd)

#3f:
# Burn = B*.1
# B = B*(1-.1)
# csig = .1
# cand.den = Normal

#3g:
#Code
```