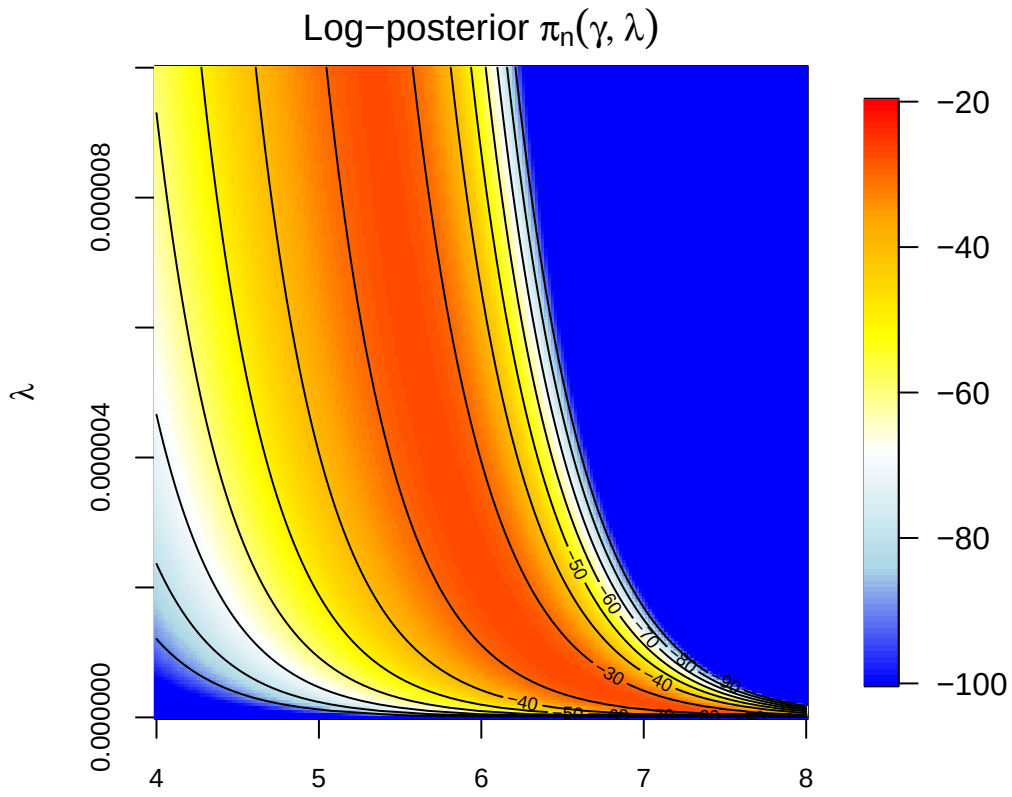
MATH 559: Bayesian Theory and Methods

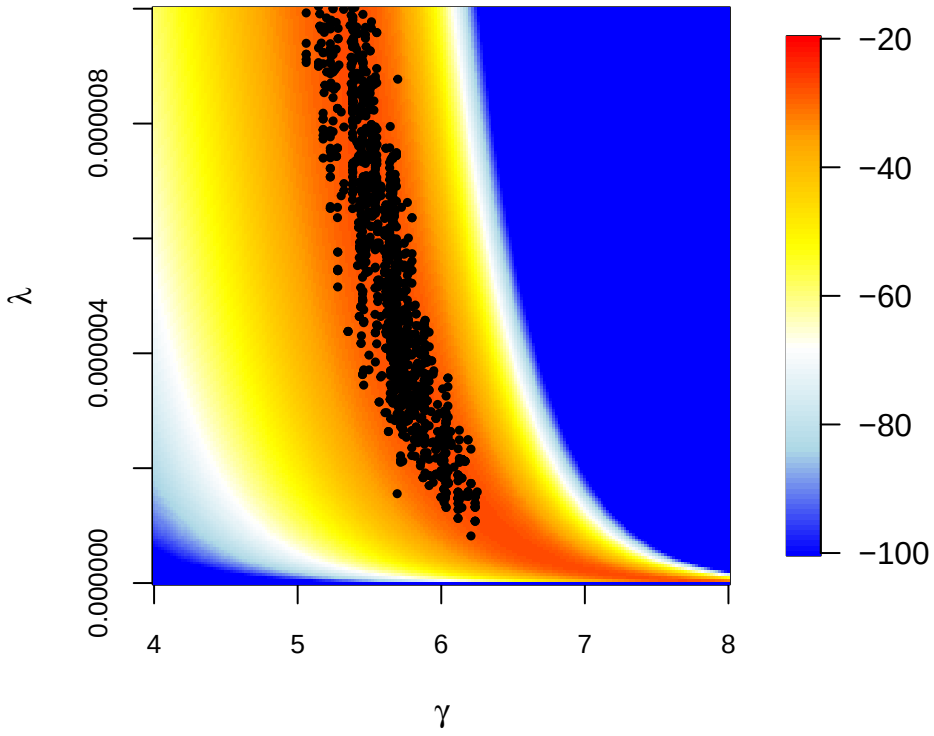
Bayesian inference for the Weibull distribution

```r
#simulate data
set.seed(323)
n<-15
Y<-rweibull(n,shape=5,scale=10)
##############################################################################

#Original parameterization
gpts<-seq(0,1,by=0.005)
g0<-gpts*4+4
l0<-gpts*1e-5
log.post.func0<-function(gv,lv,yv){
    logp<-sum(dweibull(yv,shape=gv,scale=(1/lv)^(1/gv),log=T))-0.01*(gv+lv)
    return(ifelse(logp > -100,logp,-100))
}
f <- Vectorize(log.post.func0,vectorize.args=c("gv","lv"))
log.p0<-outer(g0,l0,f,yv=Y)

#Plot
library(fields,quietly=TRUE)
par(pty='s',mar=c(2,3,2,2))
colfunc <- colorRampPalette(c("blue","lightblue","white","yellow","orange","red"))
image.plot(g0,l0,log.p0,col=colfunc(100),zlim=range(-100,-20),
           xlab=expression(gamma),ylab=expression(lambda),cex.axis=0.8)
contour(g0,l0,log.p0,add=T,levels=seq(-100,-20,by=10))
title(expression(paste('Log-posterior ',pi[n](gamma,lambda))))
```

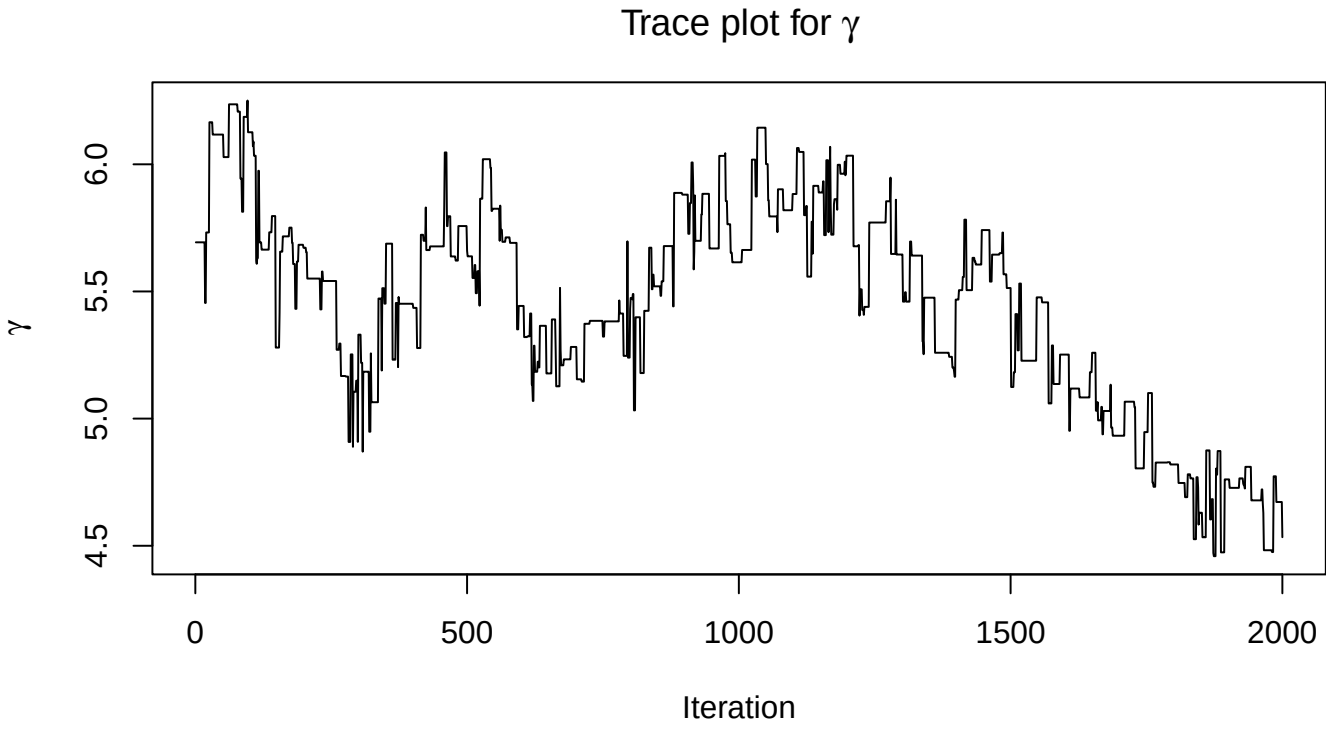


Gibbs sampler MH update for (γ, λ):

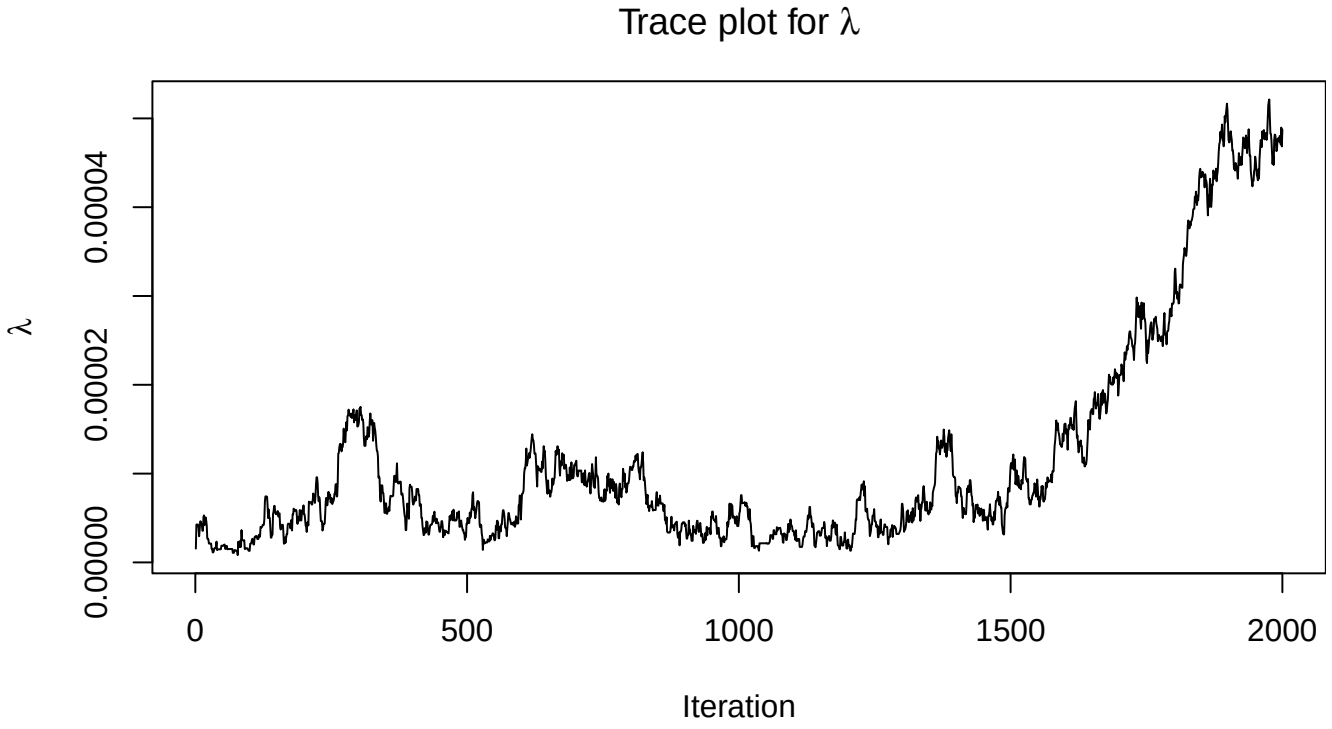
```
#Metropolis-Hastings within Gibbs sampler in original parameterization
set.seed(23)
nits<-2000
old.gam<-5.5; old.lam<-1e-6    #Starting values
sig.gam<-1;sig.lam<-1e-6       #Proposal standard deviations

old.like<-sum(log(old.gam*old.lam*Y^(old.gam-1))-old.lam*Y^old.gam)
old.prior<--0.01*(old.gam+old.lam)
old.post<-old.like+old.prior

post.samp0<-matrix(0,nrow=nits,2)
for(iter in 1:nits){
    #Gibbs sampler
    #Update gamma parameter
        new.gam<-abs(rnorm(1,old.gam,sig.gam))          #'Reflected' normal proposal
        new.lam<-old.lam
        new.like<-sum(log(new.gam*new.lam*Y^(new.gam-1))-new.lam*Y^new.gam)
        new.prior<--0.01*(new.gam+new.lam)
        new.post<-new.like+new.prior
        if(log(runif(1)) < new.post-old.post){
        #Accept the move and update the current vales
                old.gam<-new.gam; old.lam<-new.lam
                old.like<-new.like; old.prior<-new.prior; old.post<-new.post
        }
        post.samp0[iter,1]<-old.gam

    #Update lambda parameter
        new.gam<-old.gam
        new.lam<-abs(rnorm(1,old.lam,sig.lam))
        new.like<-sum(log(new.gam*new.lam*Y^(new.gam-1))-new.lam*Y^new.gam)
        new.prior<--0.01*(new.gam+new.lam)
        new.post<-new.like+new.prior
        if(log(runif(1)) < new.post-old.post){
        #Accept the move and update the current vales
                old.gam<-new.gam; old.lam<-new.lam
                old.like<-new.like; old.prior<-new.prior; old.post<-new.post
        }

    #Store the value of the chain
        post.samp0[iter,2]<-old.lam
}

par(pty="s",mar=c(4,4,2,2))
image.plot(g0,l0,log.p0,col=colfunc(100),zlim=range(-100,-20),
           xlab=expression(gamma),ylab=expression(lambda),cex.axis=0.8)
points(post.samp0,pch=19,cex=0.5)
```


λ
0.000000
0.000004
0.000008
4
5
6
7
8
γ
−20
−40
−60
−80
−100

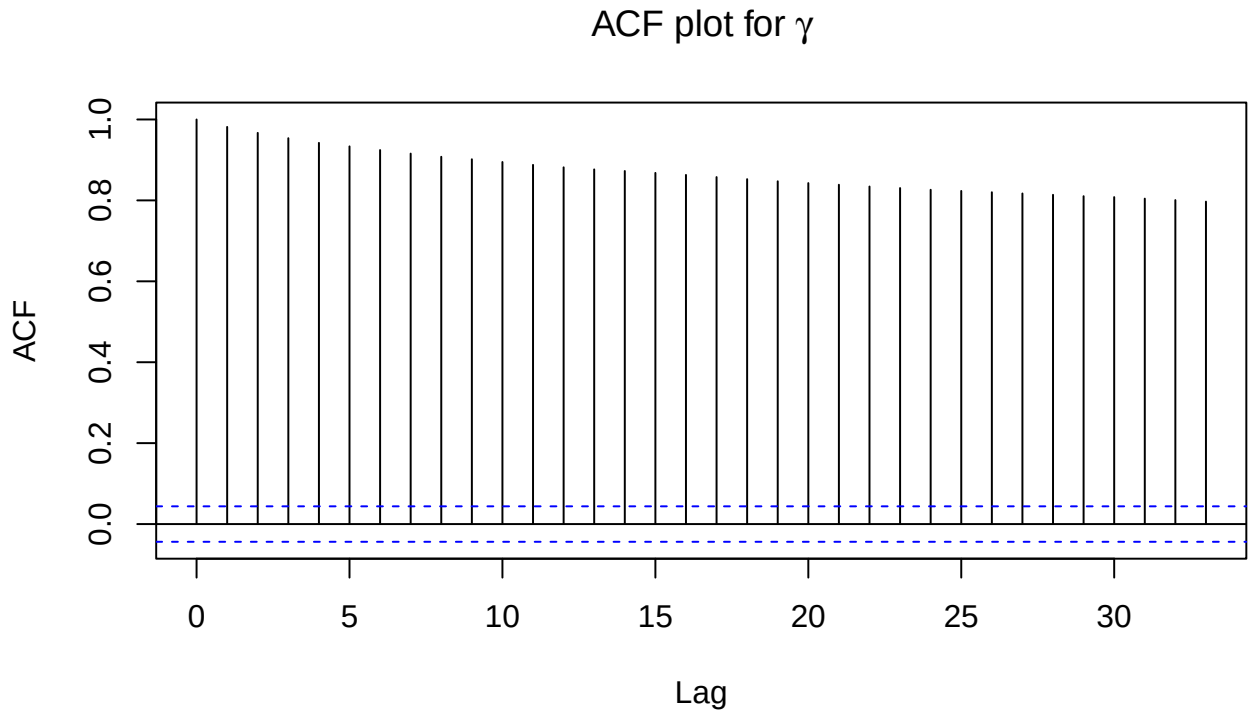
```
par(mar=c(4,4,3,0))
plot(1:nits,post.samp0[,1],type="l",xlab="Iteration",ylab=expression(gamma))
title(expression(paste('Trace plot for ',gamma)))
```



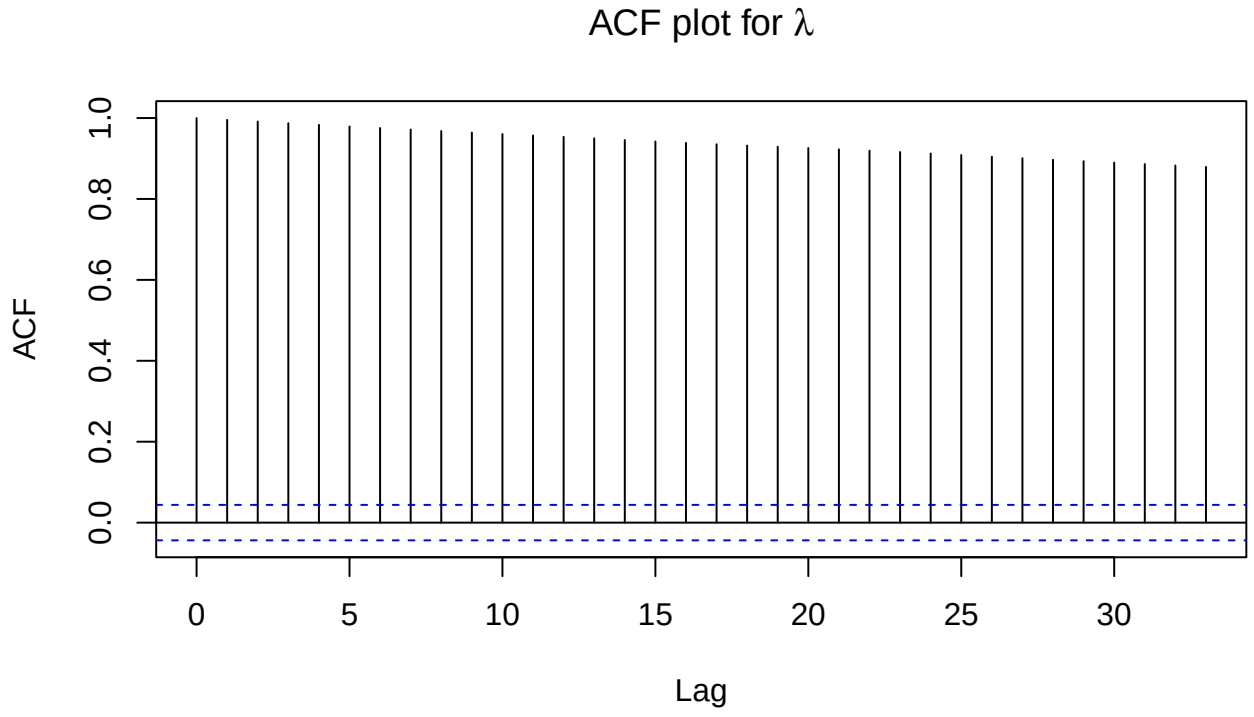

```
plot(1:nits,post.samp0[,2],type="l",xlab="Iteration",ylab=expression(lambda))
title(expression(paste('Trace plot for ',lambda)))
```



```r
acf(post.samp0[,1],main=expression(paste('ACF plot for ',gamma)))
```

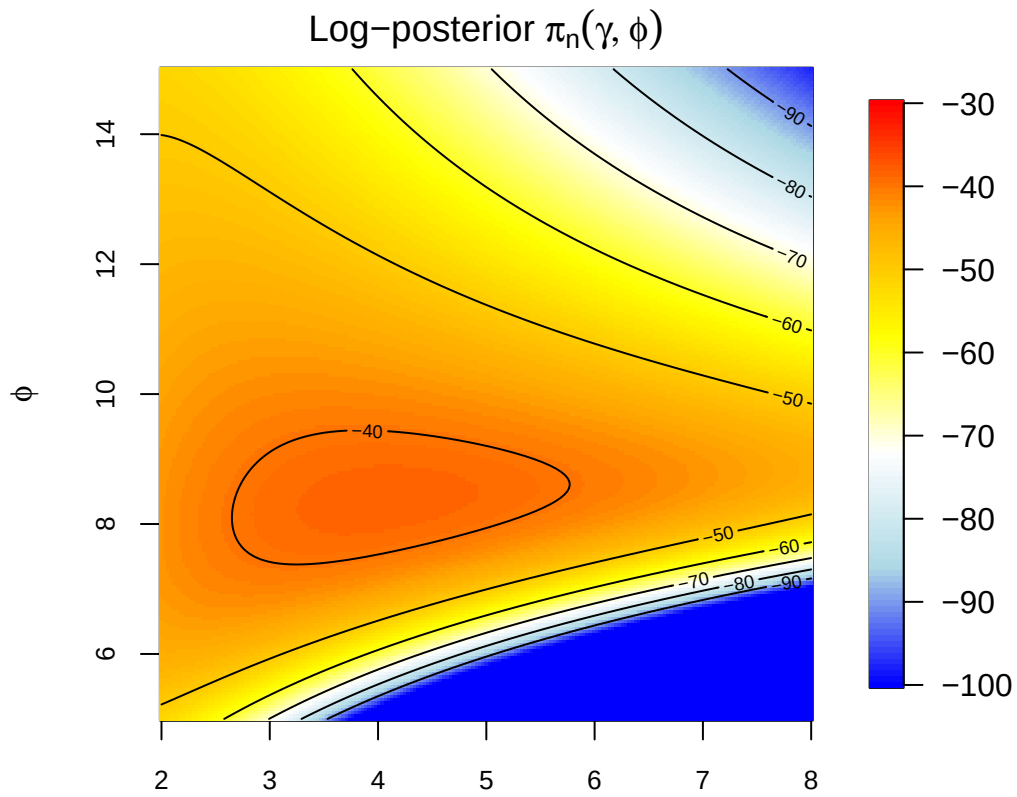



```r
acf(post.samp0[,2],main=expression(paste('ACF plot for ',lambda)))
```



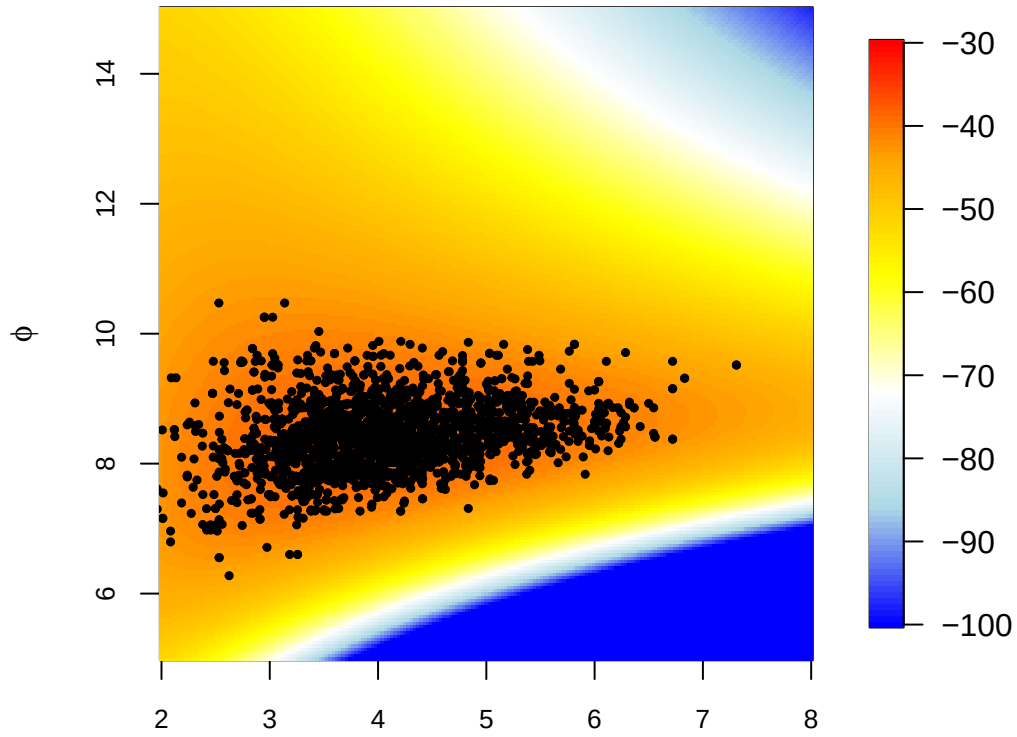
Gibbs sampler MH update for (γ, ϕ):

```
#Metropolis-Hastings within Gibbs sampler in second parameterization
g1<-gpts*6+2
ph1<-gpts*10+5

log.post.func1<-function(gv,phv,yv){
    logp<-sum(dweibull(yv,shape=gv,scale=phv,log=T))-0.01*gv-0.1*(1/phv)^gv+log(gv)-(gv+1)*log(phv)
    return(ifelse(logp > -100,logp,-100))
}
f <- Vectorize(log.post.func1,vectorize.args=c("gv","phv"))
log.p1<-outer(g1,ph1,f,yv=Y)

#Plot
library(fields,quietly=TRUE)
par(pty='s',mar=c(2,3,2,2))
colfunc <- colorRampPalette(c("blue","lightblue","white","yellow","orange","red"))
image.plot(g1,ph1,log.p1,col=colfunc(100),zlim=range(-100,-30),
           xlab=expression(gamma),ylab=expression(phi),cex.axis=0.8)
contour(g1,ph1,log.p1,add=T,levels=seq(-100,-30,by=10))
title(expression(paste('Log-posterior ',pi[n](gamma,phi))))
```

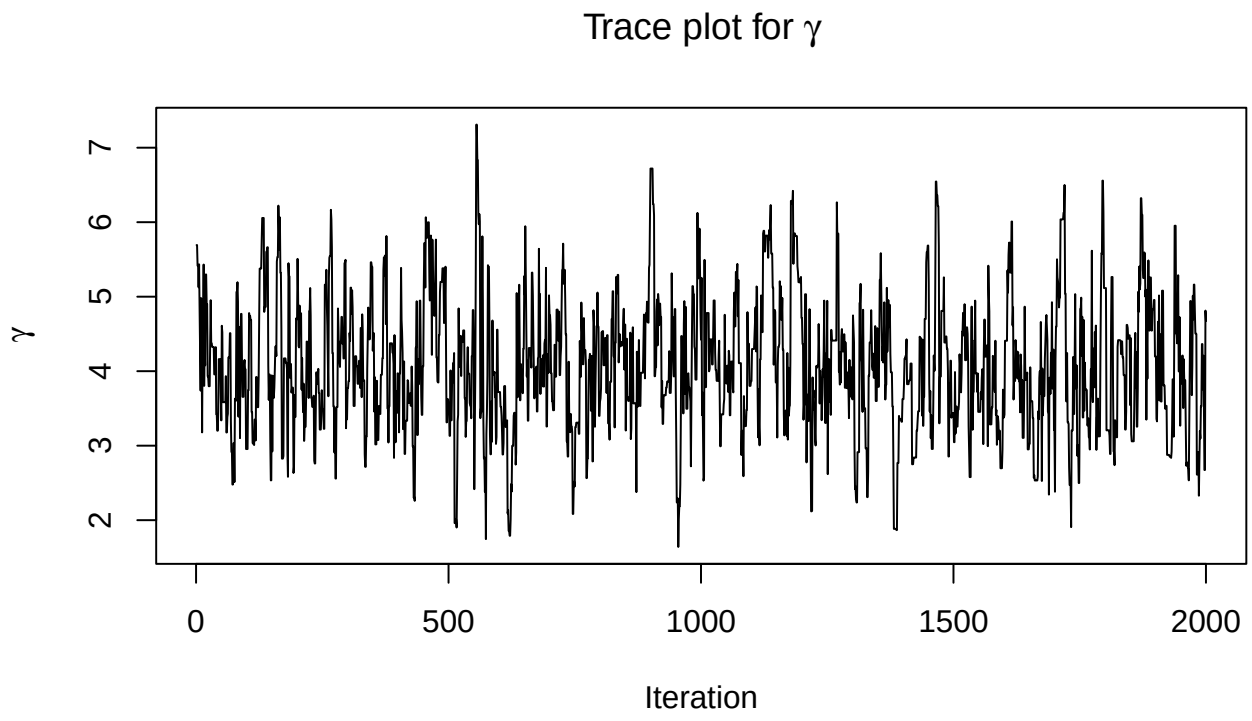



```
set.seed(23)
nits<-2000
old.gam<-5.5
old.phi<-8
sig.gam<-sig.phi<-1

old.like<-sum(dweibull(Y,shape=old.gam,scale=old.phi,log=T))
old.prior<--0.01*old.gam-0.01*(1/old.phi)^old.gam+log(old.gam)-(old.gam+1)*log(old.phi)
old.post<-old.like+old.prior

post.samp1<-matrix(0,nrow=nits,2)
```

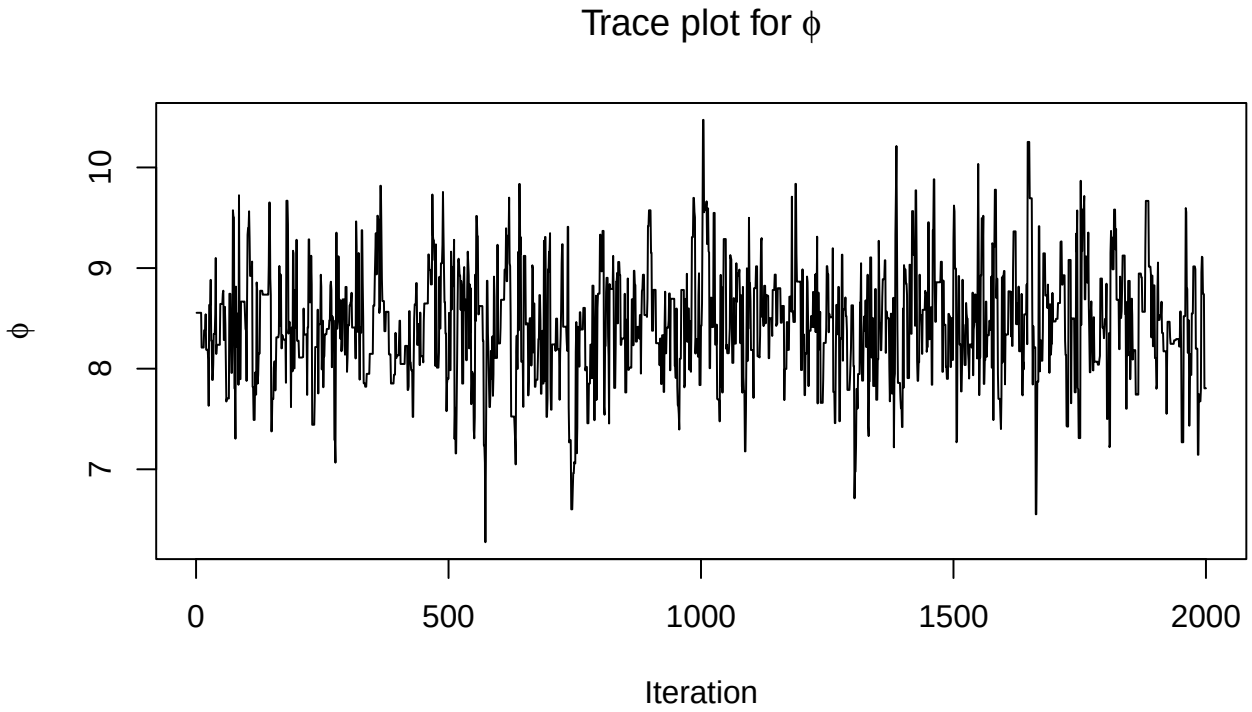
```
for(iter in 1:nits){

        new.gam<-abs(rnorm(1,old.gam,sig.gam))
        new.phi<-old.phi
        new.like<-sum(dweibull(Y,shape=new.gam,scale=new.phi,log=T))
        new.prior<--0.01*new.gam-0.01*(1/new.phi)^new.gam+log(new.gam)-(new.gam+1)*log(new.phi)
        new.post<-new.like+new.prior
        if(log(runif(1)) < new.post-old.post){
                old.gam<-new.gam; old.phi<-new.phi
                old.like<-new.like;     old.prior<-new.prior; old.post<-new.post
        }
        post.samp1[iter,1]<-old.gam

        new.gam<-old.gam
        new.phi<-abs(rnorm(1,old.phi,sig.phi))
        new.like<-sum(dweibull(Y,shape=new.gam,scale=new.phi,log=T))
        new.prior<--0.01*new.gam-0.01*(1/new.phi)^new.gam+log(new.gam)-(new.gam+1)*log(new.phi)
        new.post<-new.like+new.prior
        if(log(runif(1)) < new.post-old.post){
                old.gam<-new.gam; old.phi<-new.phi;     old.like<-new.like
                old.prior<-new.prior; old.post<-new.post
        }
        post.samp1[iter,2]<-old.phi
}
par(pty='s',mar=c(2,3,2,2))
colfunc <- colorRampPalette(c("blue","lightblue","white","yellow","orange","red"))
image.plot(g1,ph1,log.p1,col=colfunc(100),zlim=range(-100,-30),
           xlab=expression(gamma),ylab=expression(phi),cex.axis=0.8)
points(post.samp1,pch=19,cex=0.5)
```



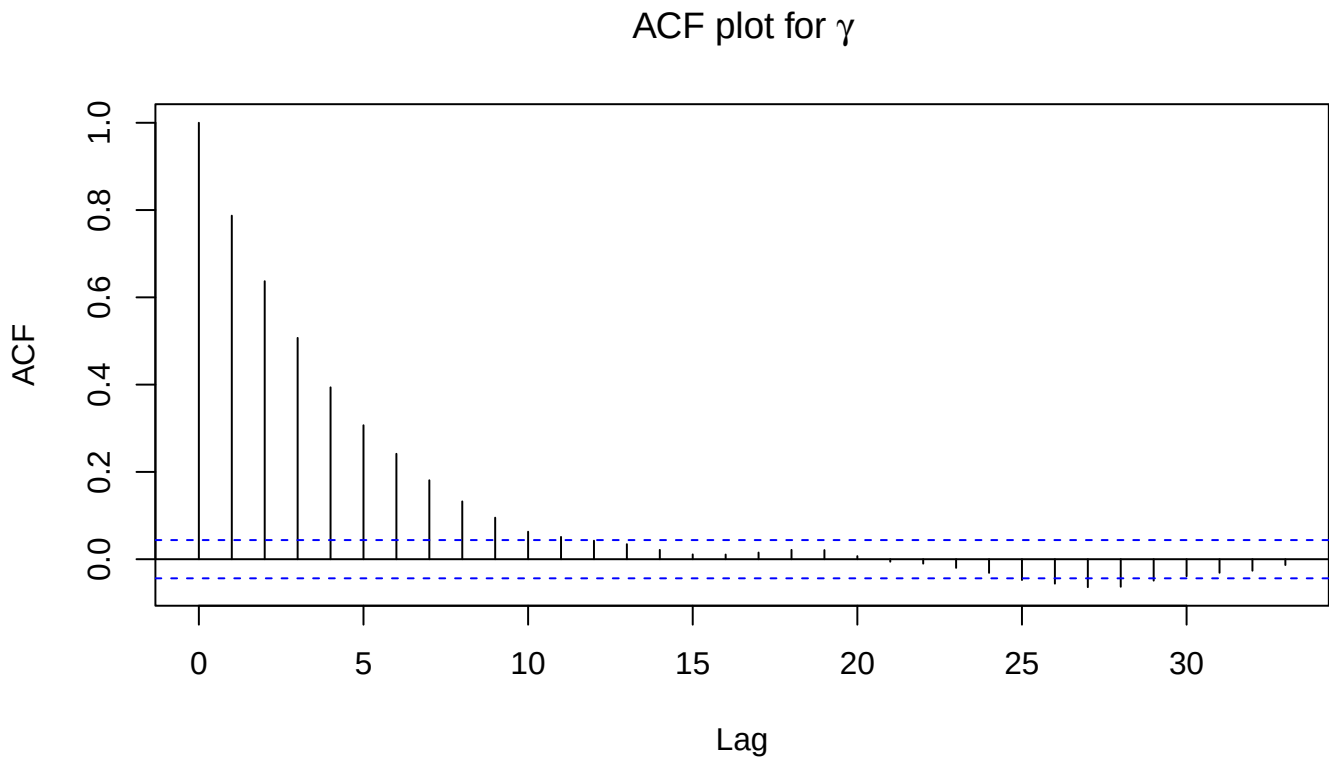

```
plot(1:nits,post.samp1[,1],type="l",xlab="Iteration",ylab=expression(gamma))
title(expression(paste('Trace plot for ',gamma)))
```



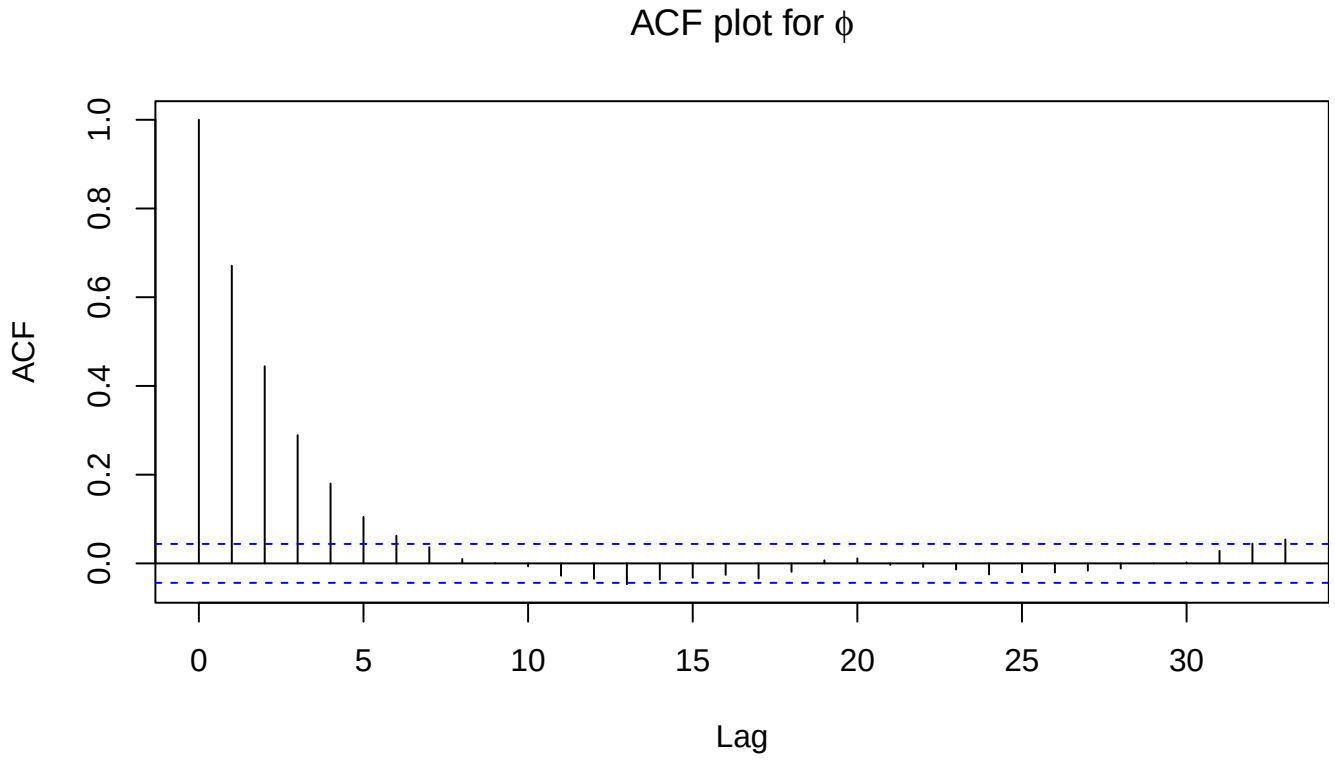

```
plot(1:nits,post.samp1[,2],type="l",xlab="Iteration",ylab=expression(phi))
title(expression(paste('Trace plot for ',phi)))
```



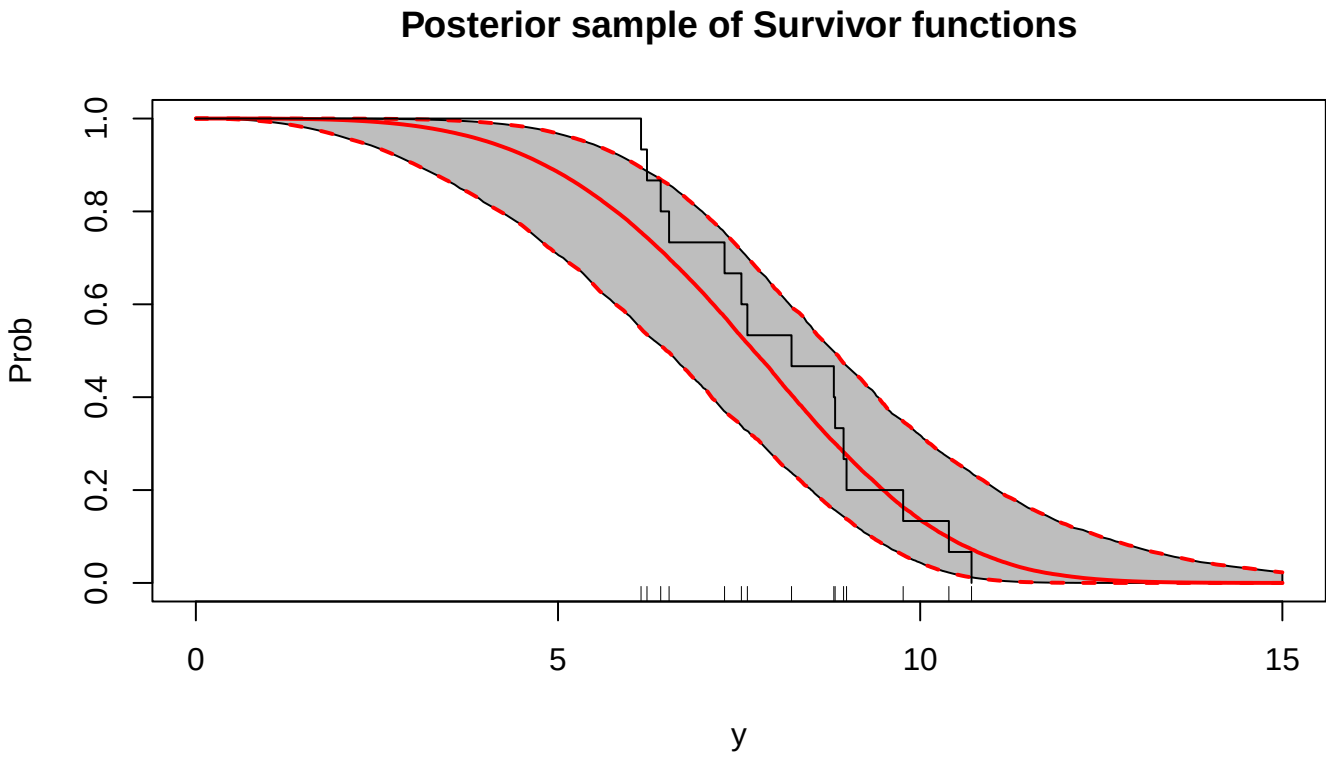
```
par(mar=c(4,4,4,0))
acf(post.samp1[,1],main=expression(paste('ACF plot for ',gamma)))
```




```
acf(post.samp1[,2],main=expression(paste('ACF plot for ',phi)))
```

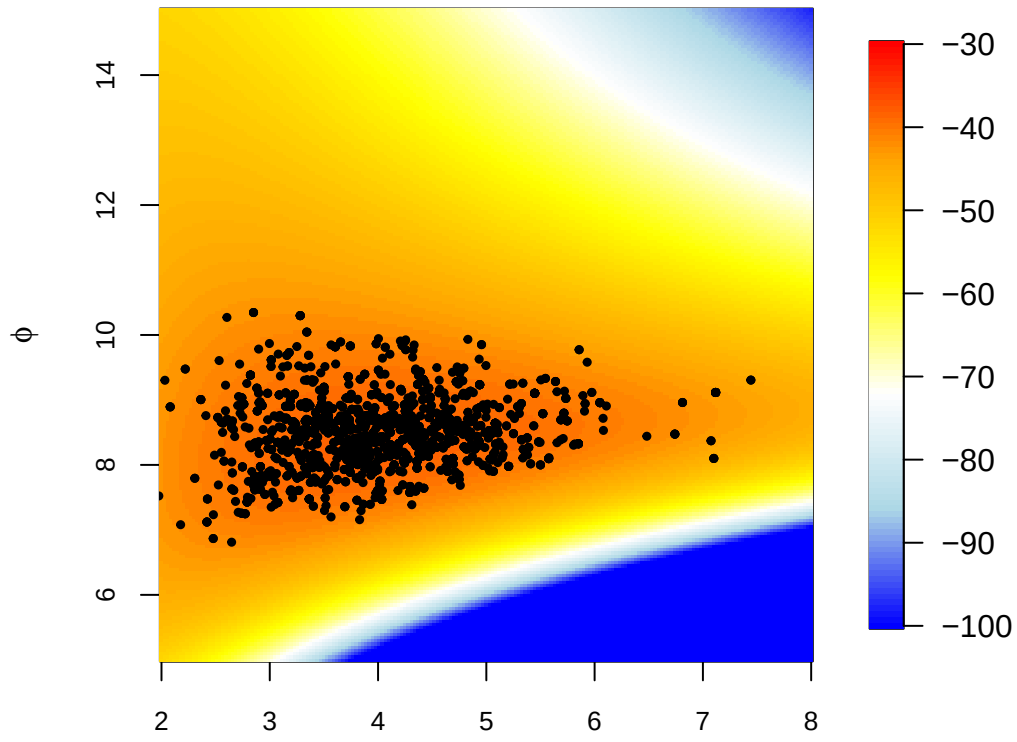


```
#Plot the posterior for the survivor function
yvec<-seq(0,15,by=0.01)
Smat<-matrix(0,nrow=nits,ncol=length(yvec))
for(i in 1:nits){
        Smat[i,]<-exp(-(yvec/post.samp1[i,2])^post.samp1[i,1])
}
Post.survivor<-apply(Smat,2,quantile,probs=c(0.025,0.5,0.975))
par(mar=c(4,4,4,0))
plot(yvec,Post.survivor[2,],type="n",xlab="y",ylab="Prob")
polygon(c(yvec,rev(yvec)),c(Post.survivor[1,],rev(Post.survivor[3,])),col="gray")
lines(yvec,Post.survivor[2,],col="red",lwd=2)
lines(yvec,Post.survivor[1,],col="red",lwd=2,lty=2)
lines(yvec,Post.survivor[3,],col="red",lwd=2,lty=2)
rug(Y)
lines(c(0,sort(Y)),1-c(0:n)/n,type="s")
title('Posterior sample of Survivor functions')
```

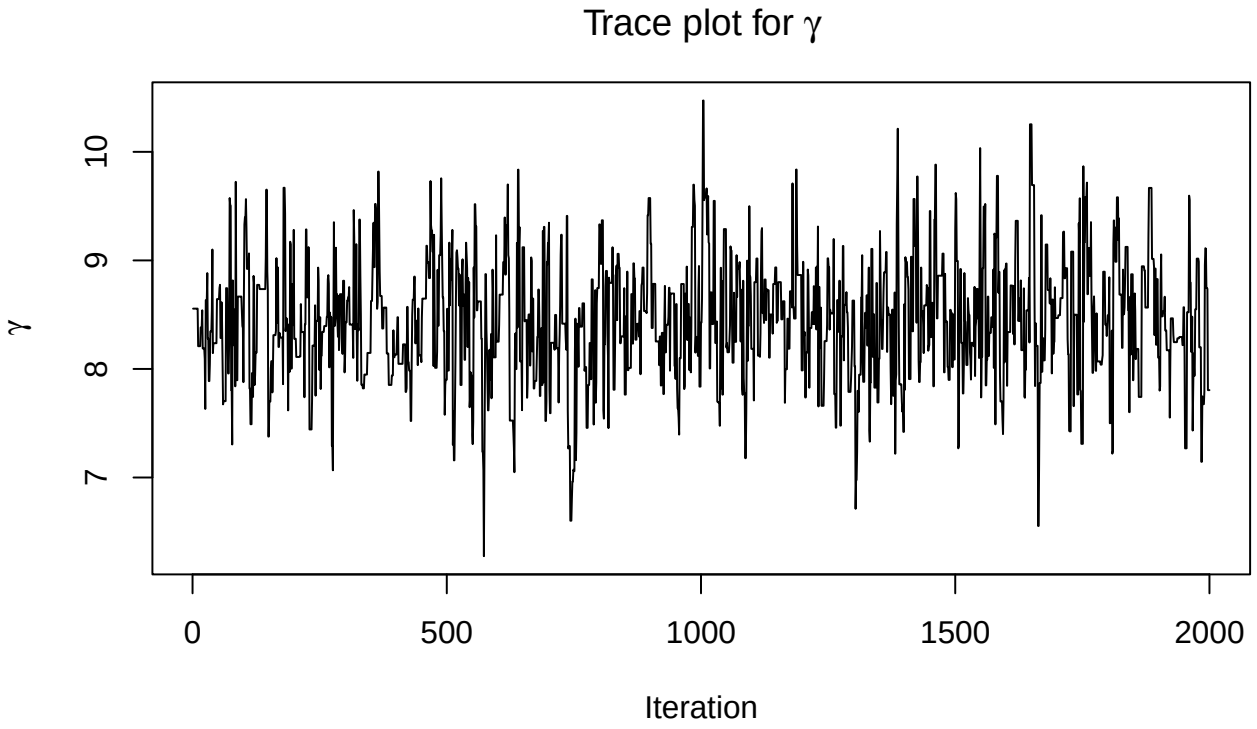


Metropolis-Hastings joint update for (γ, ϕ):

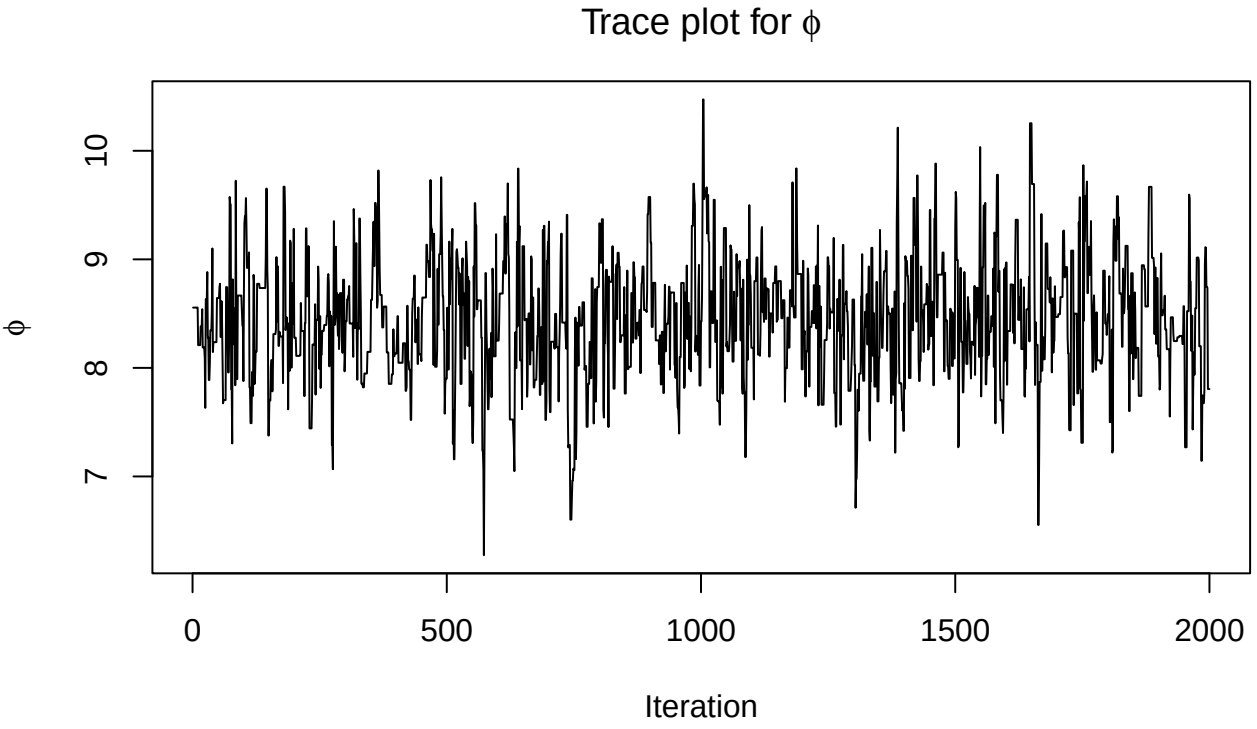
```
#Metropolis-Hastings in both parameters simultaneously in second parameterization
set.seed(23)
nits<-2000
old.gam<-5.5
old.phi<-8
old.th<-c(old.gam,old.phi)
sig.gam<-sig.phi<-1
old.like<-sum(dweibull(Y,shape=old.gam,scale=old.phi,log=T))
old.prior<--0.01*old.gam-0.01*(1/old.phi)^old.gam+log(old.gam)-(old.gam+1)*log(old.phi)
old.post<-old.like+old.prior
post.samp2<-matrix(0,nrow=nits,2)

for(iter in 1:nits){

    new.th<-rep(0,2); while(min(new.th)<=0){new.th<-old.th+rnorm(2)}  #Only accept positive values

        new.gam<-new.th[1];     new.phi<-new.th[2]
        new.like<-sum(dweibull(Y,shape=new.gam,scale=new.phi,log=T))
        new.prior<--0.01*new.gam-0.01*(1/new.phi)^new.gam+log(new.gam)-(new.gam+1)*log(new.phi)
        new.post<-new.like+new.prior
        if(log(runif(1)) < new.post-old.post){
        old.th<-new.th; old.gam<-new.gam; old.phi<-new.phi;
        old.like<-new.like;     old.prior<-new.prior; old.post<-new.post
        }
        post.samp2[iter,]<-old.th
}
par(pty='s',mar=c(2,3,2,2))
colfunc <- colorRampPalette(c("blue","lightblue","white","yellow","orange","red"))
image.plot(g1,ph1,log.p1,col=colfunc(100),zlim=range(-100,-30),
           xlab=expression(gamma),ylab=expression(phi),cex.axis=0.8)
points(post.samp2,pch=19,cex=0.5)
```



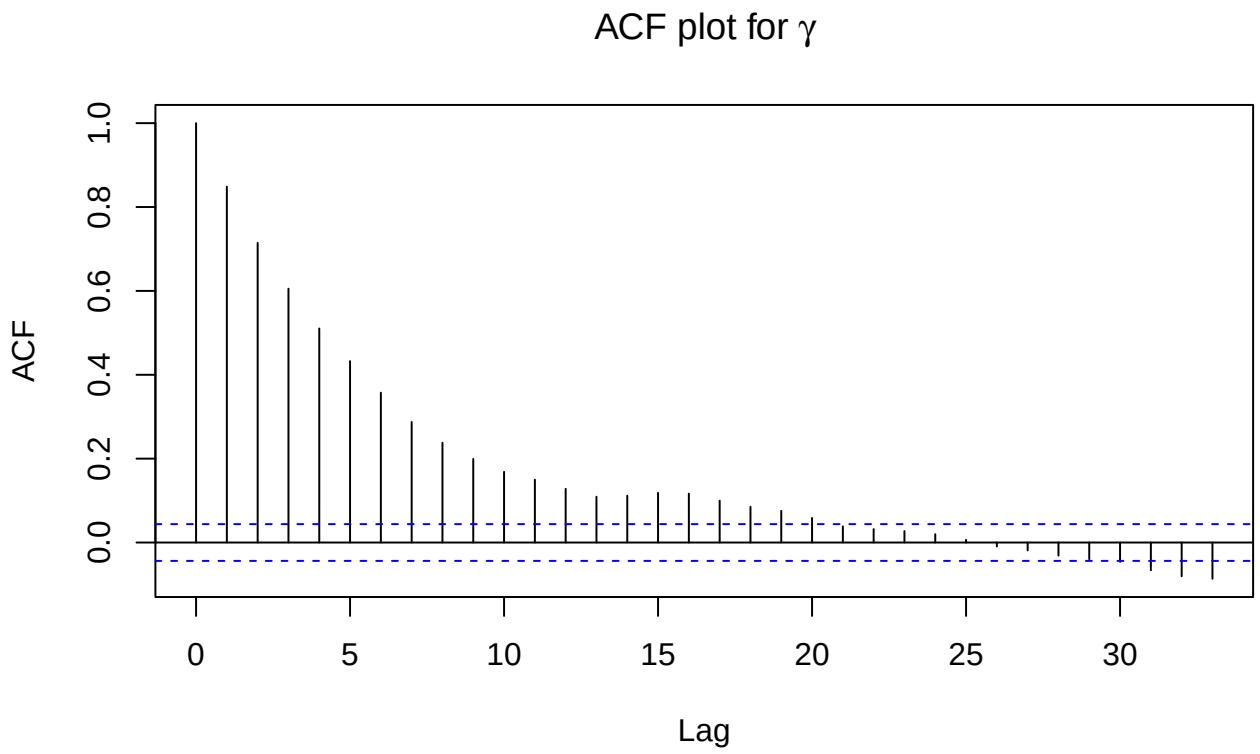
```
par(mar=c(4,4,3,2))
plot(1:nits,post.samp1[,2],type="l",xlab="Iteration",ylab=expression(gamma))
title(expression(paste('Trace plot for ',gamma)))
```



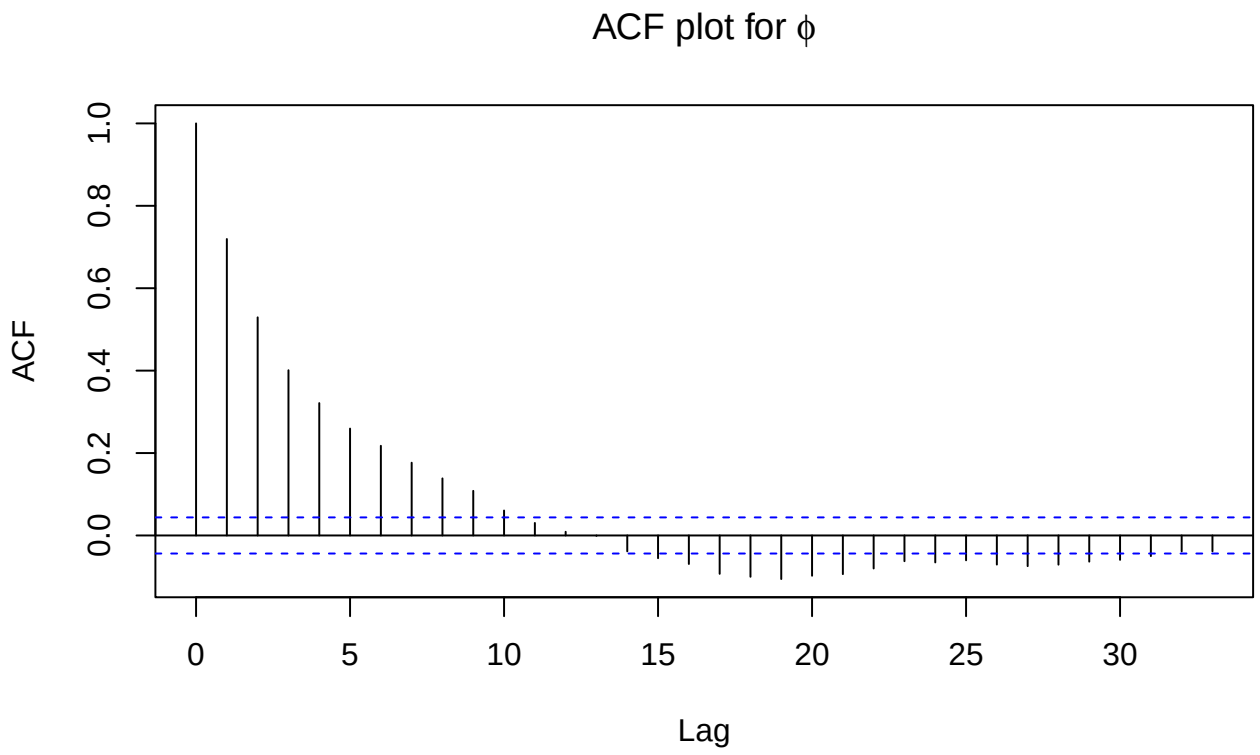

```
plot(1:nits,post.samp1[,2],type="l",xlab="Iteration",ylab=expression(phi))
title(expression(paste('Trace plot for ',phi)))
```



```
acf(post.samp2[,1],main=expression(paste('ACF plot for ',gamma)))
```

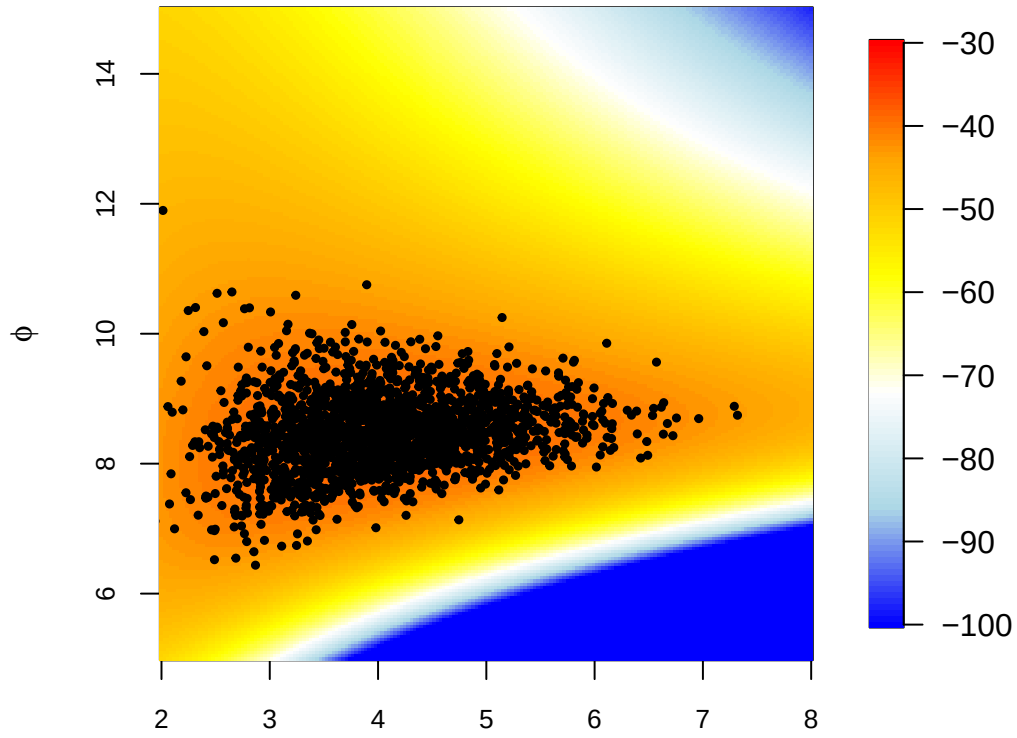



```
acf(post.samp2[,2],main=expression(paste('ACF plot for ',phi)))
```



Rejection sampling:

```
#Rejection sampling
g.shape<-2;g.scale<-2  #Gamma parameter proposal
p.shape<-4;p.scale<-2  #Phi parameter proposal
g1<-gpts*6+2; ph1<-gpts*10+5 #Grids

#Compute the f/f0 ratio on the log scale
log.rej.func1<-function(gv,phv,yv){
    logp<-sum(dweibull(yv,shape=gv,scale=phv,log=T))-0.01*gv-0.1*(1/phv)^gv+log(gv)-
    (gv+1)*log(phv)-dgamma(gv,shape=g.shape,scale=g.scale,log=T)-
                    dgamma(phv,shape=p.shape,scale=p.scale,log=T)
    return(ifelse(logp > -100,logp,-100))
}
f <- Vectorize(log.rej.func1,vectorize.args=c("gv","phv"))
log.rej<-outer(g1,ph1,f,yv=Y)
logM<-max(log.rej)

nsamp<-2000
isamp<-0; ico<-0
gam.samp<-phi.samp<-numeric()
while(isamp < nsamp){

    #Propose from independent gammas
        gam<-rgamma(1,shape=g.shape,scale=g.scale)
        phi<-rgamma(1,shape=p.shape,scale=p.scale)

        fval<-sum(dweibull(Y,shape=gam,scale=phi,log=T))-
                0.01*gam-0.01*(1/phi)^gam+log(gam)-(gam+1)*log(phi)
        gval<-dgamma(gam,shape=g.shape,scale=g.scale,log=T)+
            dgamma(phi,shape=p.shape,scale=p.scale,log=T)

        ico<-ico+1

        if(log(runif(1)) <  fval - logM - gval){
                isamp<-isamp+1
                gam.samp<-c(gam.samp,gam)
                phi.samp<-c(phi.samp,phi)
        }

}
par(pty='s',mar=c(2,3,2,2))
colfunc <- colorRampPalette(c("blue","lightblue","white","yellow","orange","red"))
image.plot(g1,ph1,log.p1,col=colfunc(100),zlim=range(-100,-30),
           xlab=expression(gamma),ylab=expression(phi),cex.axis=0.8)
points(gam.samp,phi.samp,pch=19,cex=0.5)
```


ϕ
14
12
10
8
6
2
3
4
5
6
7
8
−30
−40
−50
−60
−70
−80
−90
−100

Effective sample sizes/efficieny of sampling:

```
#Compute Effective sample sizes for MH algorithms
library(coda)
effectiveSize(post.samp0[,1]);effectiveSize(post.samp0[,2])

+     var1
+ 8.463257
+     var1
+ 4.140942

effectiveSize(post.samp1[,1]);effectiveSize(post.samp1[,2])

+    var1
+ 217.201
+    var1
+ 393.687

effectiveSize(post.samp2[,1]);effectiveSize(post.samp2[,2])

+     var1
+ 163.5654
+     var1
+ 326.1397

#Acceptance Rate for Rejection sampler
nsamp/ico

+ [1] 0.03460746

#Efficiencies
c(effectiveSize(post.samp0[,1]),effectiveSize(post.samp0[,2]))/nsamp

+        var1        var1
+ 0.004231628 0.002070471

c(effectiveSize(post.samp1[,1]),effectiveSize(post.samp1[,2]))/nsamp

+      var1      var1
+ 0.1086005 0.1968435

c(effectiveSize(post.samp2[,1]),effectiveSize(post.samp2[,2]))/nsamp

+       var1       var1
+ 0.08178268 0.16306983

nsamp/ico

+ [1] 0.03460746
```