MATH3871/MATH5960
Bayesian Inference and Computation

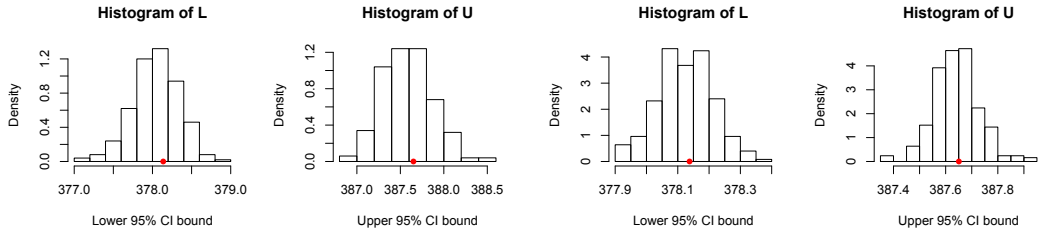
Lab 1 Exercises (Outline solutions)

1) Water consumption question from lectures

(a) The posterior mean is given by $(\alpha + \sum_i x_i)/(\beta + n) = 382.8796$. The lower $(x \times 100)\%$ of a Gamma(a, b) distribution can be computed by `qgamma(x,a,b)`. Hence, with $x = 0.025$ and $x = 0.975$, the following `R` code will a 95% central credible interval of (378.14, 387.65).

```
n=65; sumx=24890
alpha=1; beta=0.01
pmean=(alpha+sumx)/(beta+n)
L=qgamma(0.025,alpha+sumx,beta+n)
U=qgamma(0.975,alpha+sumx,beta+n)
cat("Posterior mean: ",pmean," (",L,",",U,")")
```

(b) To obtain a quantile estimate from a sample (`samp`), either we can sort the sample (`R` command `sort`) and look at the $q * N$-th element (e.g. `sort(samp)[0.025*N]`) or use the `quantile` function.



The two histograms with $N = 500$ (left plots) and $N = 5000$ (right plots) can be obtained with the following code:

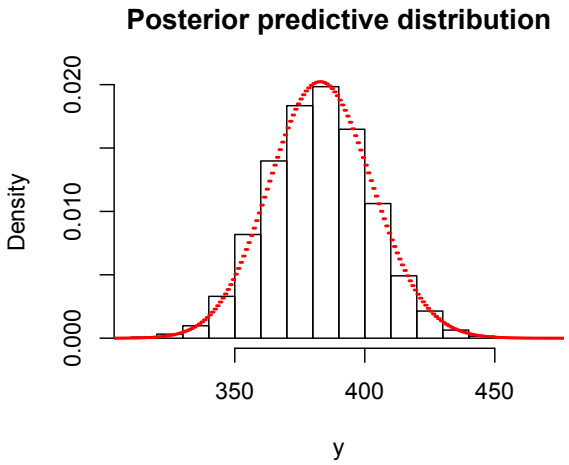
```
N=500 # or 500 or 5000
L=U=rep(NA,length=250)
for (i in 1:250) {
dat=sort(rgamma(N,alpha+sumx,beta+n))
L[i]=dat[0.025*N]
U[i]=dat[0.975*N]
}
widthL=max(L)-min(L)
widthU=max(U)-min(U)
par(mfrow=c(1,2))
hist(L,probability=T,xlab="Lower 95% CI bound")
```

```
points(qgamma(0.025,alpha+sumx,beta+n),0,pch=16,col=2)
hist(U,probability=T,xlab="Upper 95% CI bound")
points(qgamma(0.975,alpha+sumx,beta+n),0,pch=16,col=2)
cat("L interval variability (range):",widthL,"\n")
cat("U interval variability (range):",widthU,"\n")
```

The precision in the estimates is increased as the number of samples increases. In order to get the width of the histogram to be less than 0.15 for both lower and upper confidence limits, I had to increase N to around 70,000.

(c) The following `R` code will produce the below posterior predictive plot:

```
alpha=1; beta=0.01
sumx=24890; n=65
theta=rgamma(5000,alpha+sumx,beta+n)
y=rpois(5000,theta)
hist(y,probability=T,ylab="Density",main="Posterior predictive distribution")
xx=300:500
pr=dnbinom(xx,sumx+alpha,1-1/(beta+n+1))
#lines(xx,pr,col=2) # The (incorrect) continuous version - ok as an approx.
# The (correct) discrete version:
for (i in 1:length(xx)) lines(c(xx[i],xx[i+1]),rep(pr[i],2),col=2,lwd=2)
```

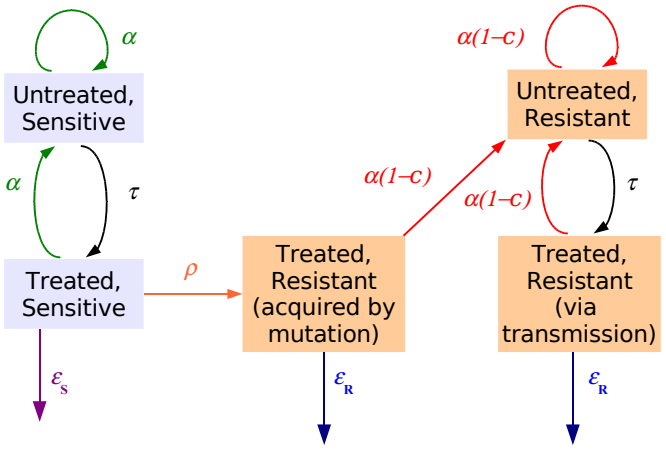


Clearly, the (more simply obtained) Monte Carlo samples are exact draws from the true (algebraically obtained) posterior distribution. This is very reassuring!

(g) Algebraically exact approach:
Pros: The answers will always be exact, with no simulation error.
Cons: It is often tedious to perform the calculations, and it is frequently not possible to perform them in closed form.

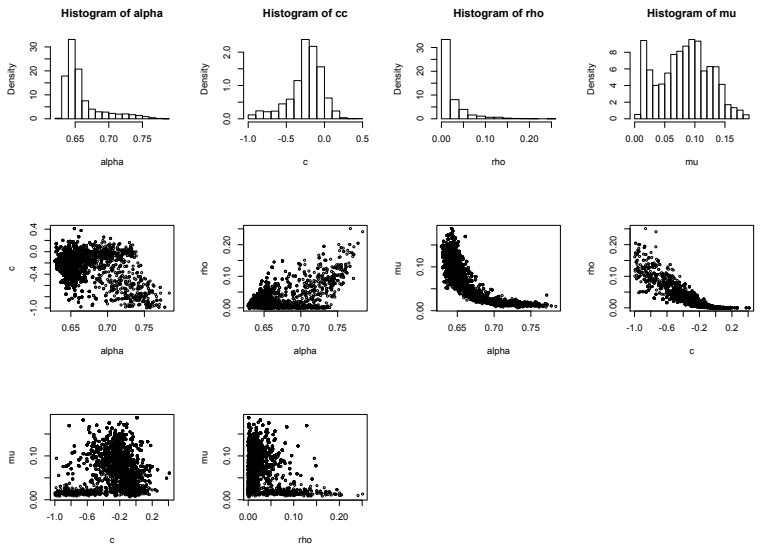
Simulation:
Pros: It is usually trivial to obtain estimates of whatever posterior quantities you need.
Cons: They are only estimates, and come with estimation/simulation error (that can be controlled).

2) Estimating evolutionary fitness of tuberculosis



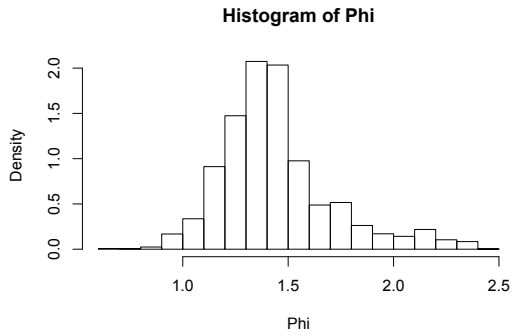
(a) The below `R` code will read in the data and produce the following plot.

```
dat=read.table("tuberculosis.txt")

alpha=dat[,1]
cc=dat[,2]  # Note if you use 'c=dat[,2]' you will overwrite the R command 'c'
rho=dat[,3] # with horrible consequences!
mu=dat[,4]

par(mfrow=c(3,4))
hist(alpha,probability=T,nclass=13,xlab="alpha")
hist(cc,   probability=T,nclass=13,xlab="c")
hist(rho,  probability=T,nclass=13,xlab="rho")
hist(mu,   probability=T,nclass=13,xlab="mu")

plot(alpha,cc ,pty=16,cex=0.5,xlab="alpha",ylab="c")
plot(alpha,rho,pty=16,cex=0.5,xlab="alpha",ylab="rho")
plot(alpha,mu, pty=16,cex=0.5,xlab="alpha",ylab="mu")
plot(cc,   rho,pty=16,cex=0.5,xlab="c",ylab="rho")
plot(cc,   mu,pty=16,cex=0.5,xlab="c",ylab="mu")
plot(rho,  mu,pty=16,cex=0.5,xlab="rho",ylab="mu")
```



(b) The posterior histogram can be produced by the below code:

```
par(mfrow=c(1,1))
delta=tau=eS=0.52
eR=0.202
Phi = (1-cc)*(1/tau + 1/(delta+eR))/(1/tau + 1/(delta+eS+rho))
hist(Phi,probability=T, nclass=13,xlab="Phi")
prob=sum(Phi<1)/length(Phi)
cat("Pr(Phi<1) =",prob,"\n")
```



The posterior probability that $\Phi < 1$ is 0.0206. Hence, there is strong evidence that $\Phi > 1$, and hence that the resistant strains are more evolutionarily fit than the susceptible strains.

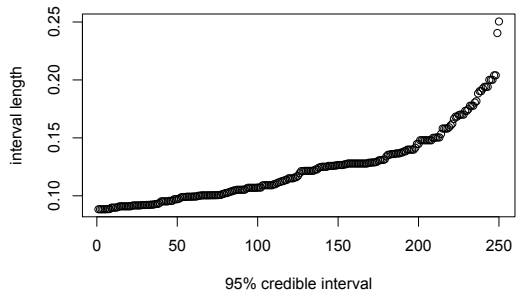
(c) The below code computes the central 95% credible interval for ρ

```
sortrho=sort(rho)
L=sortrho[0.025*5000]
U=sortrho[0.975*5000]
cat("95% central credible interval is (",L,",",U,")\n")
```

The interval is of length $U - L = 0.117176$. This interval is very wide as the posterior distribution for ρ is skewed.

(d) The below code evaluates and plots the length of the 250 95% credible intervals, and identifies the minimum length interval:

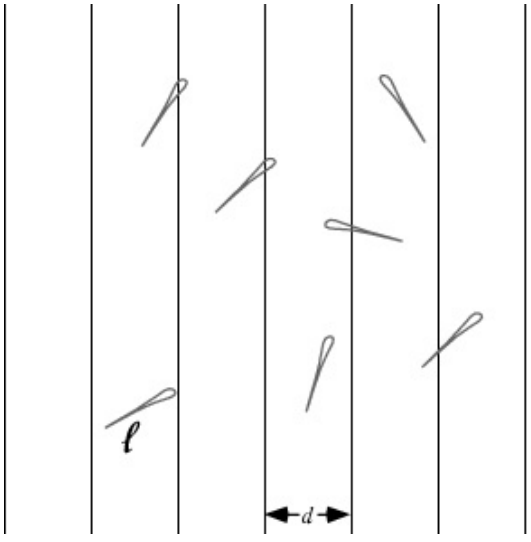
```
sortrho=sort(rho)
u=0.05*5000 # index of CI lower bound
len=rep(NA,u)
for (i in 1:u) len[i]=sortrho[5000*0.95+i]-sortrho[i]
plot(len,xlab="95% credible interval",ylab="interval length")
ind=which.min(len)
cat("Minimum interval length is ",len[ind], "(",sortrho[ind],",",
sortrho[5000*0.95+ind],")\n")
```



Because the posterior for ρ is skewed, the minimum length credible is the one obtained by removing the upper 5% of the posterior sample. The length of this interval is 0.088362, which is much less than the 0.117176 of the central credible interval.

The central credible interval is likely to be near the minimum length of all credible intervals when the posterior is unimodal and symmetric.

3) Buffon's Needle

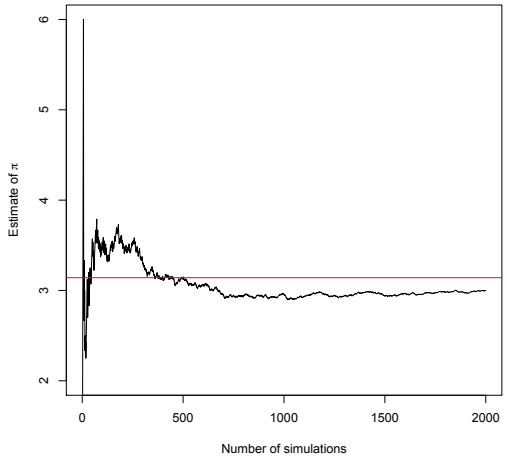


a) The following function will simulate the experiment for any values of ℓ and d.

```
buffon=function(n,d=1,ell=1, make.plot=TRUE) {
x=runif(n,0,d)
theta=runif(n, 0, pi)
L=cos(theta)
```

```
int=(ell*L)>x
C=cumsum(int)
if (make.plot) {
plot(1:n, (1:n)*ell/(d*C), type="l", xlab="Number of simulations",
ylab=expression(paste("Estimate of ",pi)), ylim=c(2,6))
abline(pi,0,col=2)
}
return(n*ell/(d*C[n]))
}
buffon(2000)
```

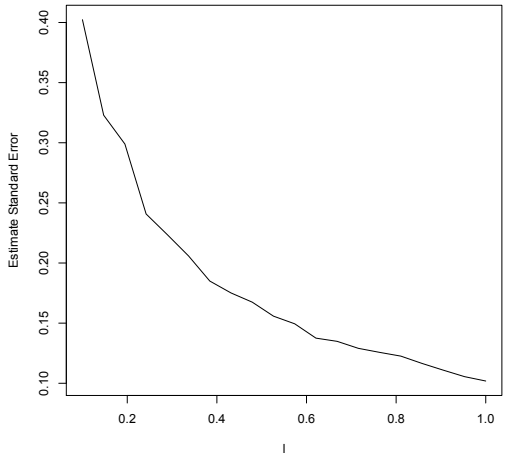
In this example (plot below) I got an estimate of 2.998501.



b) $\mathrm{Var}(1/\hat{\pi}) = \mathrm{Var}(\hat{p}d/2\ell) = \frac{1}{4\rho^2}p(1-p)/n = \frac{1}{4\rho^2}\frac{2\rho}{\pi}(1-\frac{2\rho}{\pi})/n = \frac{1}{n\pi^2}(\frac{\pi}{2\rho}-1)$.
This is minimised when $\rho \leq 1$ is as large as possible. Hence $\rho = 1$.

c) Using the above function, we can use the following code:

```
out=rep(NA,1000)
n=2000
ell.seq=seq(0.1,1,length=20)
sd.out=rep(NA,length(ell.seq))
for (j in 1:length(ell.seq)) {
for (i in 1:1000) out[i]=buffon(n,ell=ell.seq[j],make.plot=F)
sd.out[j]=sd(out)
}
plot(ell.seq,sd.out,xlab="l",ylab="Estimate Standard Error",type='l')
```


Estimate Standard Error
0.10
0.15
0.20
0.25
0.30
0.35
0.40
0.2
0.4
0.6
0.8
1.0
l