Solution to computer exam in Bayesian learning

Per Siden

2019-10-31

First load all the data into memory by running the R-file given at the exam

```
rm(list=ls())
source("ExamData.R")
set.seed(1)
```

Problem 1

1a

```
theta = 0.6
```

The expected utility when buying the option is

```
EUbuy = theta*30 + (1-theta)*(-10)
print(EUbuy)
```

```
## [1] 14
```

The expected utility when not buying the option is

```
EUnotbuy = theta*90 + (1-theta)*(-120)
print(EUnotbuy)
```

```
## [1] 6
```

Since the expected utility when buying the option is higher (14 compared to 6), the bank should buy the option.

1b

On paper.

1c

Solution based on simulation:

```r
nSamples = 5000
alpha = 3
beta = 2
s = 62
f = 38
thetaSim = rbeta(nSamples,alpha+s,beta+f)
xSim = rbinom(nSamples,1,thetaSim)
EUbuyC1 = mean(xSim)*30 + (1-mean(xSim))*(-10)
EUnotbuyC1 = mean(xSim)*90 + (1-mean(xSim))*(-120)
print(EUbuyC1)
```

```
## [1] 14.608
```

```r
print(EUnotbuyC1)
```

```
## [1] 9.192
```

Solution based on result in (b):

```r
EUbuyC2 = (13/21)*30 + (1-(13/21))*(-10)
EUnotbuyC2 = (13/21)*90 + (1-(13/21))*(-120)
print(EUbuyC2)
```

```
## [1] 14.7619
```

```r
print(EUnotbuyC2)
```

```
## [1] 10
```

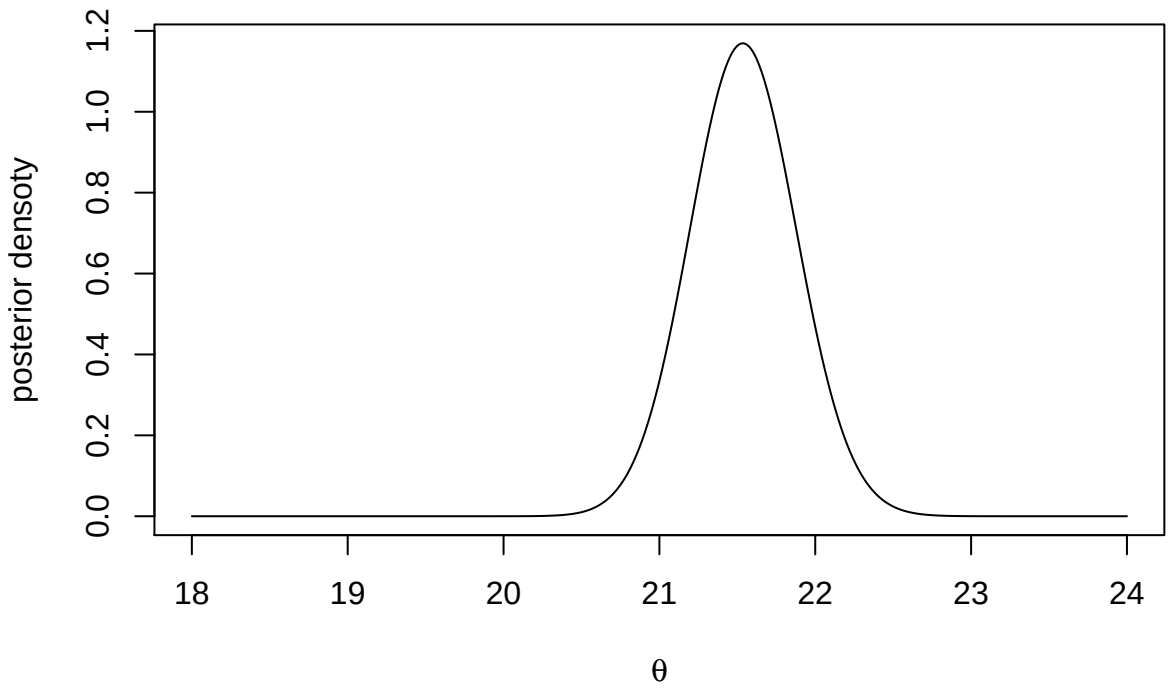
Since the expected utility when buying the option is higher (14.8 compared to 10), the bank should buy the option.

Problem 2

2a

The conjugate prior is $Gamma(\alpha = 20, \beta = 1)$, see Lecture 2

```r
nSim = 5000
alphaPrior = 20
betaPrior  = 1
n = nrow(Traffic)
sumY = sum(Traffic$y)
grid = seq(18,24,.01)
plot(grid,dgamma(grid,shape = alphaPrior + sumY, rate = betaPrior + n),type="l",xlab=expression(theta),
```



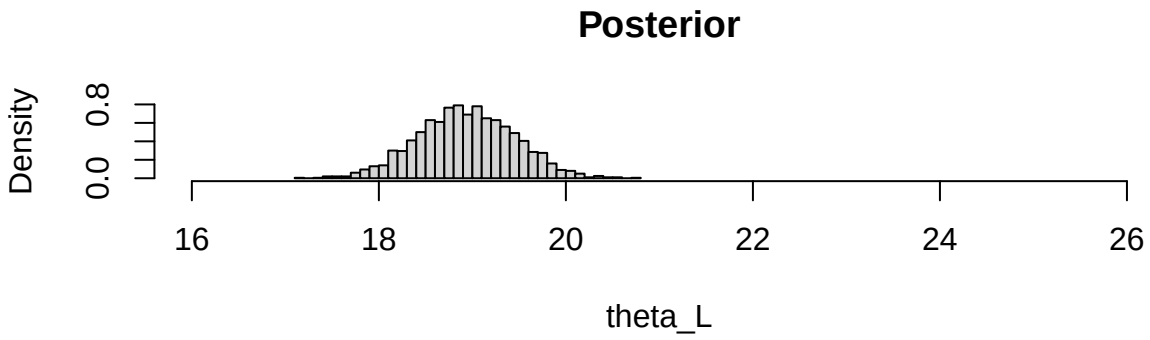

```
print(pgamma(21,shape = alphaPrior + sumY, rate = betaPrior + n))
```

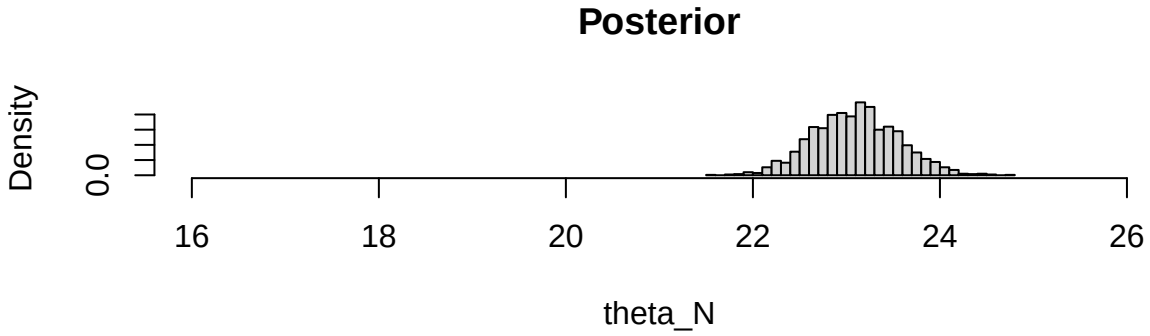
```
## [1] 0.05566221
```

The probability is 0.056

2b

```
nSamples = 2000
yL = Traffic$y[Traffic$limit=="yes"]
yN = Traffic$y[Traffic$limit=="no"]
simL = rgamma(nSamples,shape = alphaPrior + sum(yL), rate = betaPrior + length(yL))
simN = rgamma(nSamples,shape = alphaPrior + sum(yN), rate = betaPrior + length(yN))
par(mfrow=c(2,1))
hist(simL,30,prob=TRUE,main="Posterior",xlab = "theta_L",xlim=c(16,26))
hist(simN,30,prob=TRUE,main="Posterior",xlab = "theta_N",xlim=c(16,26))
```






```
mean(simN>simL)
```

```
## [1] 1
```

The posterior for θ_N is strictly to the right compared to that of θ_L, meaning that the average number of accidents with no speed limit is greater compared to when there is a limit, with posterior probability (close to) 1.

2c

```
print(mean(.85*simN>simL))
```

```
## [1] 0.8675
```

The posterior probability that $.85 * \theta_N > \theta_L$ is roughly 86%, which means that it is likely that the underlying average number of accidents decrease with more than 15% when having a speed limit. However, 86% is lower than the 95% normally used in statistical testing, so it is a bit strong to claim that the experiment proves this.

Problem 3

3a

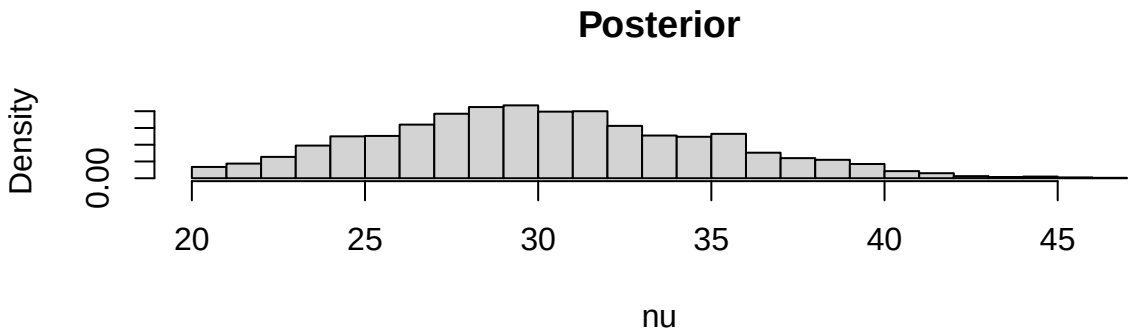
See solution on paper.

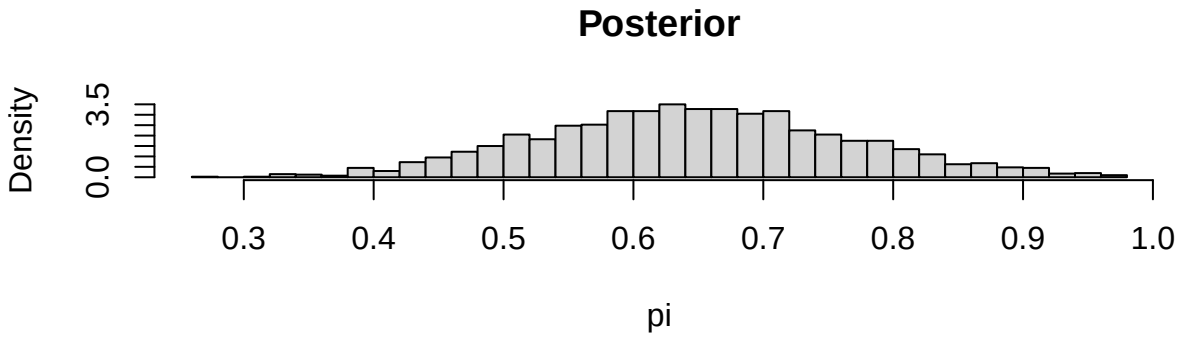
3b

See solution on paper.

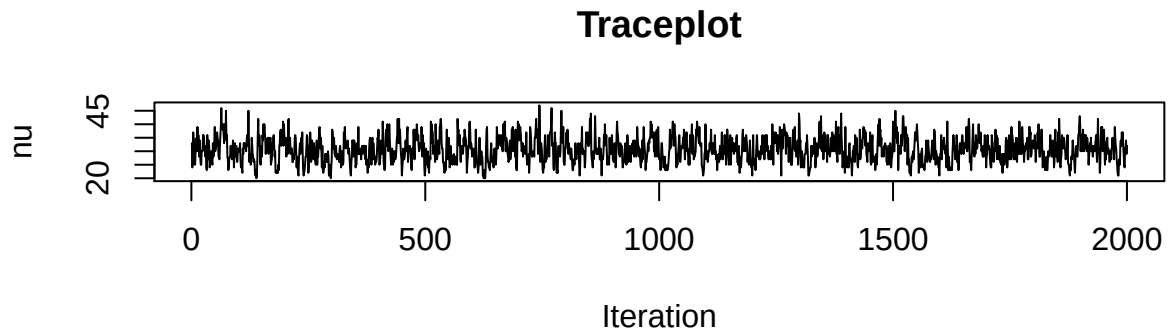
3c

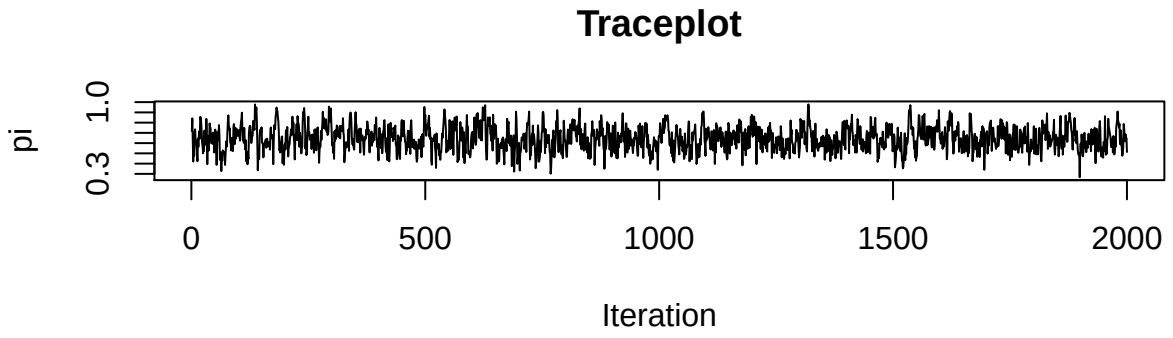
```
x = 20
lambda = 30
alpha = 2
beta = 2
# start values
burnin = 500
niter = 2000
nu <- 30
pi <- .5
nu_vec <- rep(0,burnin+niter)
pi_vec <- rep(0,burnin+niter)
nu_vec[1] <- nu
pi_vec[1] <- pi
for(i in 2:(burnin+niter)){
  z <- rpois(1,lambda*(1-pi))
  nu = z + x
  nu_vec[i] <- nu
  pi <- rbeta(1,alpha+x,beta+nu-x)
  pi_vec[i] <- pi
}
par(mfrow=c(2,1))
hist(nu_vec[(burnin+1):(burnin+niter)],30,prob=TRUE,main="Posterior",xlab = "nu")
hist(pi_vec[(burnin+1):(burnin+niter)],30,prob=TRUE,main="Posterior",xlab = "pi")
```






```
par(mfrow=c(2,1))
plot(nu_vec[(burnin+1):(burnin+niter)],type="l",main="Traceplot",ylab = "nu",xlab="Iteration")
plot(pi_vec[(burnin+1):(burnin+niter)],type="l",main="Traceplot",ylab = "pi",xlab="Iteration")
```





The Markov chain seems to have good mixing, since it rapidly explores the posterior, so the convergence is good.

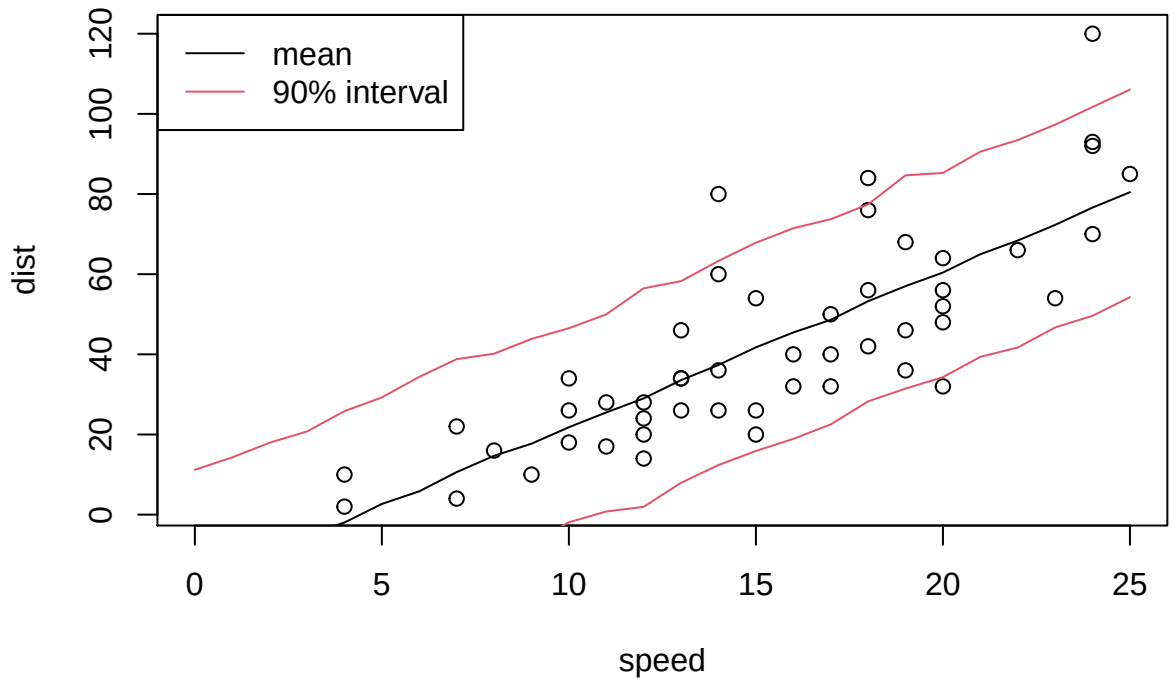
Problem 4

4a

```
data = list(N=nrow(cars),y=cars$dist,x=cars$speed)
burnin = 500
niter = 2000
fit = stan(model_code=LinRegModel,data=data,
           warmup=burnin,iter=niter,chains=1)
postDraws <- extract(fit)
```

```
xgrid = 0:25
ysim = matrix(0,26,niter-burnin)
ymean = matrix(0,26,1)
ybands = matrix(0,26,2)
for(i in 1:26){
  ysim[i,] = postDraws$alpha + xgrid[i] * postDraws$beta + sqrt(postDraws$sigma2) * rnorm(niter-burnin,
  ymean[i] = mean(ysim[i,])
  ybands[i,] = quantile(ysim[i,],probs=c(.05,.95))
}
plot(cars,xlim=c(0,25))
```

```
lines(xgrid,ymean)
lines(xgrid,ybands[,1],col=2)
lines(xgrid,ybands[,2],col=2)
legend("topleft",c("mean","90% interval"),col=c(1,2),lty=1)
```



4b

```
print(fit,digits_summary=3)
```

```
## Inference for Stan model: 67ae848567b5f9c776dd4cd088e4d5ad.
## 1 chains, each with iter=2000; warmup=500; thin=1;
## post-warmup draws per chain=1500, total post-warmup draws=1500.
##
##            mean se_mean     sd     2.5%      25%      50%      75%    97.5%
## alpha   -17.141   0.258  6.660  -29.503  -21.637  -17.443  -13.007   -3.458
## beta      3.905   0.016  0.413    3.072    3.650    3.927    4.174    4.664
## sigma2  234.331   1.811 48.553  162.592  200.310  226.215  260.114  352.363
## lp__   -176.828   0.056  1.307 -180.254 -177.454 -176.466 -175.880 -175.364
##        n_eff  Rhat
## alpha    668 0.999
## beta     656 0.999
## sigma2   719 1.000
## lp__     548 1.002
```

```
##
## Samples were drawn using NUTS(diag_e) at Sun May 24 14:15:18 2020.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

The 95% credible interval for α is roughly [-32,-5]. A real-world interpretation of this interval is that at speed 0 mph the average stopping distance of cars between -32 and -5 ft, with 95% probability. This is not very realistic, as the car is already standing still, and it is not reasonable that it would move backwards.

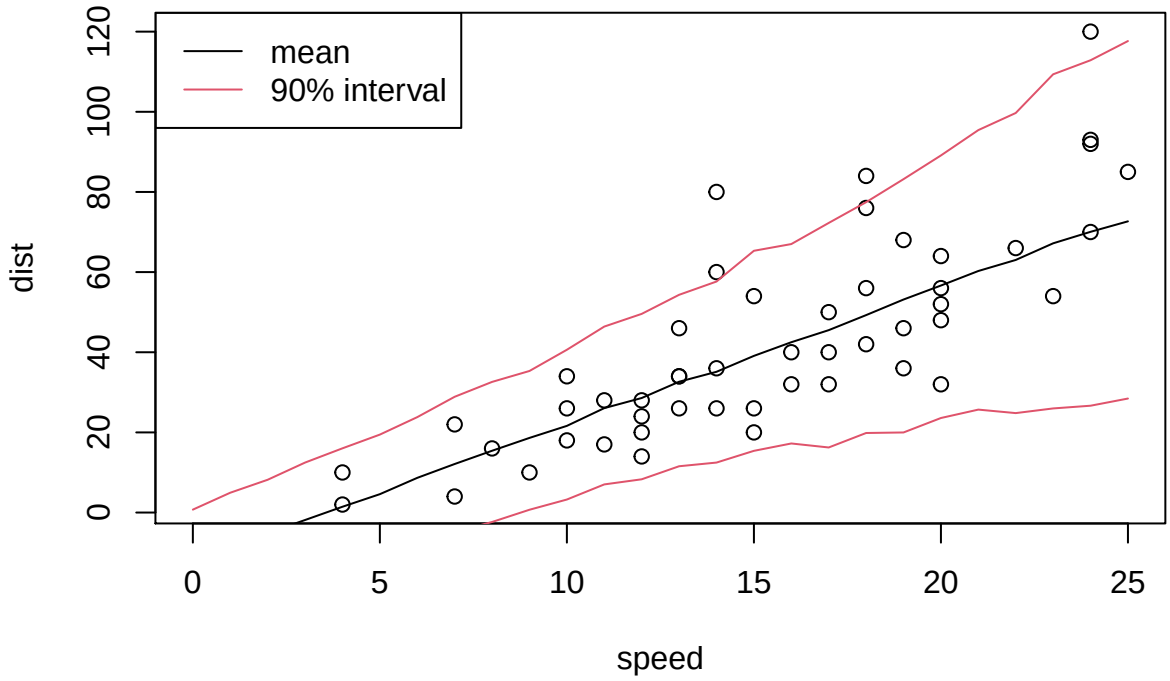
One possible change to the model would be to fix $\alpha = 0$, but this would make the linear fit to the data worse. A possiby better solution would be to use a lognormal distribution instead of a normal which guarantees that y is always positive.

4c

```
HeteroModel <- '
data {
  int<lower=0> N;
  vector[N] x;
  vector[N] y;
}
parameters {
  real alpha;
  real beta;
  real gamma;
  real phi;
  real<lower=0> sigma2[N];
}
model {
  for (n in 1:N){
    sigma2[n] ~ scaled_inv_chi_square(5,exp(gamma + phi * x[n]));
    y[n] ~ normal(alpha + beta * x[n], sqrt(sigma2[n]));
  }
}
'
fit2 = stan(model_code=HeteroModel,data=data,
            warmup=burnin,iter=niter,chains=1)
postDraws2 <- extract(fit2)
```

```
xgrid = 0:25
sigma2sim = matrix(0,26,niter-burnin)
ysim = matrix(0,26,niter-burnin)
ymean = matrix(0,26,1)
ybands = matrix(0,26,2)
for(i in 1:26){
  sim0 <- rchisq(niter-burnin,5)
  sigma2sim[i,] <- 5*exp(postDraws2$gamma + xgrid[i] * postDraws2$phi)^2/sim0
  ysim[i,] = postDraws2$alpha + xgrid[i] * postDraws2$beta + sqrt(sigma2sim[i,]) * rnorm(niter-burnin,0
  ymean[i] = mean(ysim[i,])
  ybands[i,] = quantile(ysim[i,],probs=c(.05,.95))
}
```

```
plot(cars,xlim=c(0,25))
lines(xgrid,ymean)
lines(xgrid,ybands[,1],col=2)
lines(xgrid,ybands[,2],col=2)
legend("topleft",c("mean","90% interval"),col=c(1,2),lty=1)
```



The heteroscedastic model captures the increasing variance with speed in a better way.