Project 1

Kent Codding

2025-02-19

1) Specify two different Bayesian models for each biomarker expression

(two different models for CHD8, two different models for DNAJC6, for a total of four). Proceed by modeling conditional distributions for the biomarkers given the outcome, as done in class. Options include many of the distributions in Table 6.2 (pg. 34) of manualjags.pdf. combined with variable transformations of your choice. You are not constrained to using the same distribution for cases and controls. You may play with the choice of prior as well. Using visualizations of your choice, discuss the relative strengths and limitations of your two models for each expression.

```
# install.packages('knitr') # if not installed, run this line for knitr installation.
library(knitr)
```

```
## Warning: package 'knitr' was built under R version 4.4.2
```

```
library(curatedOvarianData)
library(pROC)
```

```
## Warning: package 'pROC' was built under R version 4.4.2
```

```
# packages for Bayes modeling
library(rjags)
```

```
## Warning: package 'rjags' was built under R version 4.4.2
```

```
## Warning: package 'coda' was built under R version 4.4.2
```

```
# library(R2jags)
library(pracma)
```

```
## Warning: package 'pracma' was built under R version 4.4.2
```

```
library(coda)
library(jagsUI)
```

```
## Warning: package 'jagsUI' was built under R version 4.4.2
```

```
data(GSE32063_eset)
# Refer to Section 1 material
```

CHD8

```
XX = as.matrix(cbind(exprs(GSE32063_eset)))
YY = 1 * as.vector(pData(GSE32063_eset)[,"debulking"]=="suboptimal")
XX = XX[,!is.na(YY)]
YY = YY[!is.na(YY)]
```

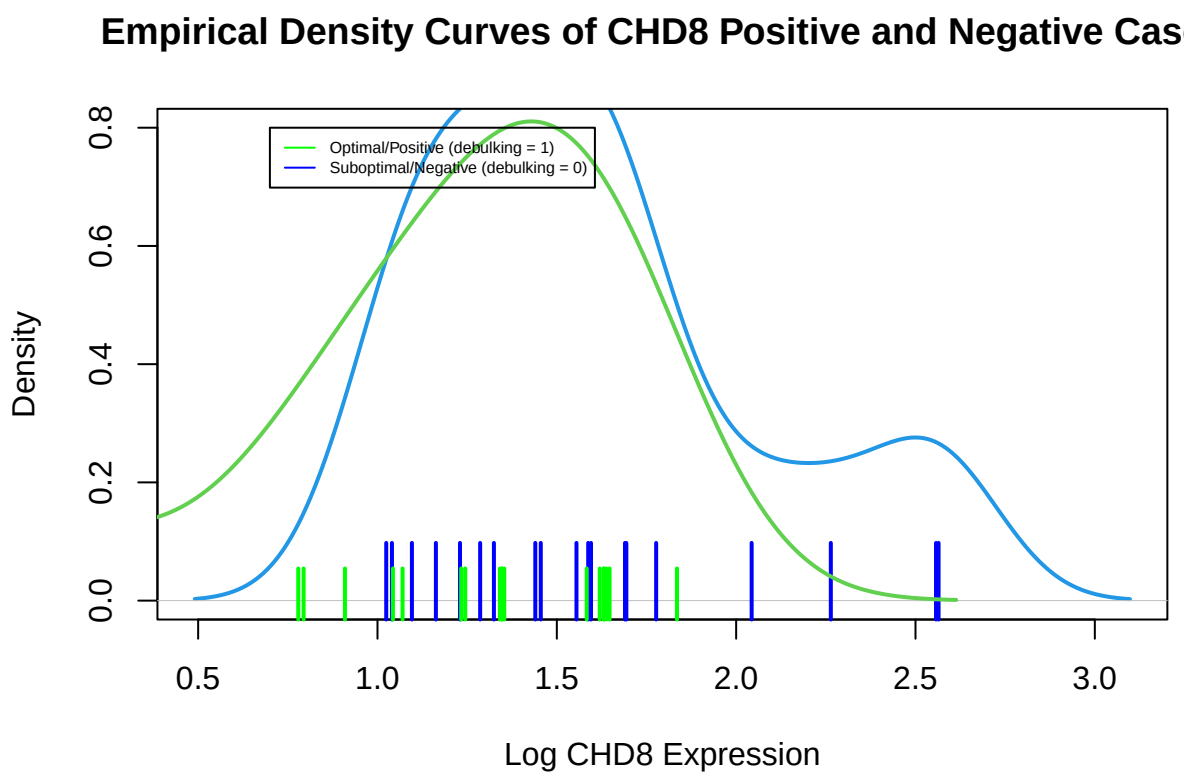
```
chd.neg <- exprs(GSE32063_eset)["CHD8",][which(YY == 0)]
chd.pos <- exprs(GSE32063_eset)["CHD8",][which(YY == 1)]
```

plot empirical density curves to determine prior

```
plot(density(chd.neg), xlab = "Log CHD8 Expression", col = 4, lwd = 2,
     main = "Empirical Density Curves of CHD8 Positive and Negative Cases",
     ylim = c(0, 0.8))
rug(chd.neg,ticksize = .15,lwd=2, col = "blue")
lines(density(chd.pos),xlab="Log CHD8 Expression",col=3,lwd=2,main="")
rug(chd.pos,ticksize = .1,lwd=2, col = "green")
```

```
## Warning in rug(chd.pos, ticksize = 0.1, lwd = 2, col = "green"): some values
## will be clipped
```

```
legend(0.7, 0.8, c("Optimal/Positive (debulking = 1)",
                   "Suboptimal/Negative (debulking = 0)"),
       lty=c(1,1), col=c("green", "blue"), cex = 0.52)
```



negative curve has a long right tail, so Gamma may be an appropriate prior choice. Positive curve has a longer left tail, so maybe increase the rate parameter to ensure that the right tail is weaker by exponential decay.

Modeling CHD8 Negative Arm using a Gamma Distribution

```
chd.neg.gamma <- "model{
  for (i in 1:N)
  {
  x[i] ~ dgamma(sh,ra)
  }

  sh ~ dgamma(0.5, 0.5)
  ra ~ dgamma(0.5, 0.5)
}"

num_chains <- 3
init_vals <- vector("list",length=num_chains)
for(i in 1:num_chains){
  init_vals[[i]]$.RNG.name="base::Mersenne-Twister"
  init_vals[[i]]$.RNG.seed=117+i
}



chd.neg.gamma <- jags(data = list(x = as.vector(exp(chd.neg)),N=length(chd.neg)), inits = init_vals, pa
```

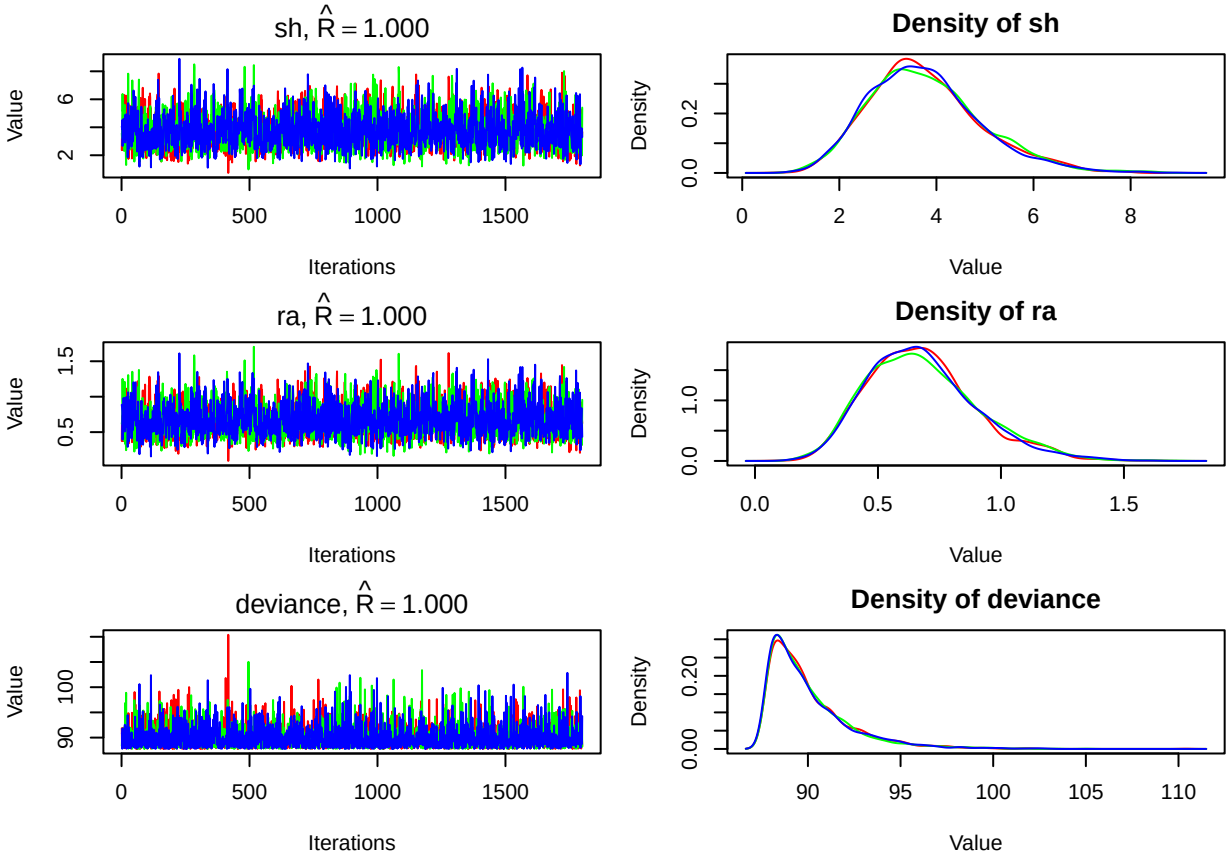
```
##
## Processing function input.......
##
## Done.
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 19
##    Unobserved stochastic nodes: 2
##    Total graph size: 23
##
## Initializing model
##
## Adaptive phase, 2000 iterations x 3 chains
## If no progress bar appears JAGS has decided not to adapt
##
##
##  Burn-in phase, 2000 iterations x 3 chains
##
##
## Sampling from joint posterior, 18000 iterations x 3 chains
##
##
## Calculating statistics.......
##
## Done.
```

```
print(chd.neg.gamma)
```

```
## JAGS output for model '4', generated by jagsUI.
```

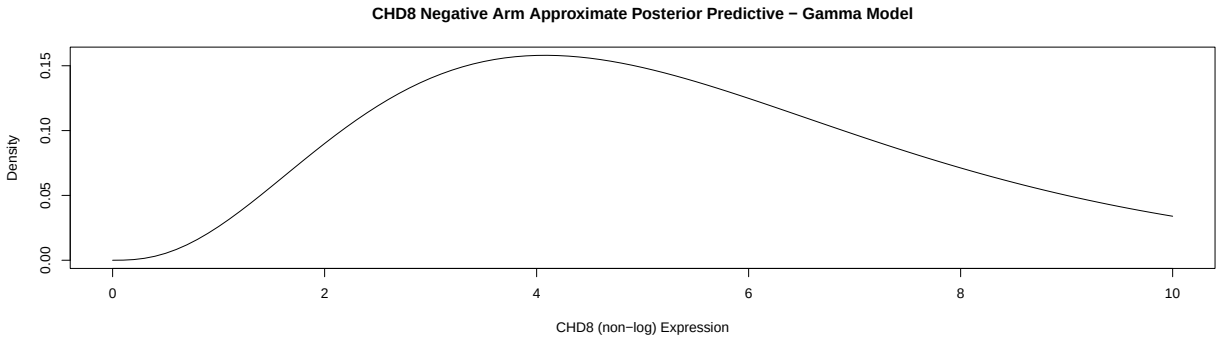
```
## Estimates based on 3 chains of 20000 iterations,
## adaptation = 2000 iterations (sufficient),
## burn-in = 2000 iterations and thin rate = 10,
## yielding 5400 total samples from the joint posterior.
## MCMC ran for 0.028 minutes at time 2025-02-24 17:52:01.772744.
##
##             mean    sd   2.5%    50%  97.5% overlap0 f Rhat n.eff
## sh         3.772 1.156  1.847  3.649  6.372    FALSE 1    1  5400
## ra         0.680 0.222  0.320  0.658  1.183    FALSE 1    1  5400
## deviance 90.022 2.221 87.807 89.386 96.075    FALSE 1    1  5400
##
## Successful convergence based on Rhat values (all < 1.1).
## Rhat is the potential scale reduction factor (at convergence, Rhat=1).
## For each parameter, n.eff is a crude measure of effective sample size.
##
## overlap0 checks if 0 falls in the parameter's 95% credible interval.
## f is the proportion of the posterior with the same sign as the mean;
## i.e., our confidence that the parameter is positive or negative.
##
## DIC info: (pD = var(deviance)/2)
## pD = 2.5 and DIC = 92.49
## DIC is an estimate of expected predictive error (lower is better).
```

```
plot(chd.neg.gamma)
```




```
# extract samples simulated from posterior for shape and rate
```

```
sh.chain.chd.neg <- chd.neg.gamma$sims.list$sh
ra.chain.chd.neg <- chd.neg.gamma$sims.list$ra

curve(dgamma(x, mean(sh.chain.chd.neg), mean(ra.chain.chd.neg)), from = 0, to = 10, xlab = "CHD8 (non-l
```



Modeling CHD8 Positive Arm using a Gamma Distribution

```
chd.pos.gamma <- "model{
  for (i in 1:N)
  {
  x[i] ~ dgamma(sh,ra)
  }

  sh ~ dgamma(0.5, 0.5)
  ra ~ dgamma(2, 0.5)
}"

num_chains <- 3
init_vals <- vector("list",length=num_chains)
for(i in 1:num_chains){
  init_vals[[i]]$.RNG.name="base::Mersenne-Twister"
  init_vals[[i]]$.RNG.seed=117+i
}

#exponentiate log-transformed data s.t. Gamma can handle all positive values
chd.pos.exp <- exp(chd.pos)

chd.pos.gamma <- jags(data = list(x = as.vector(chd.pos.exp),N=length(chd.pos.exp)), inits = init_vals,
```

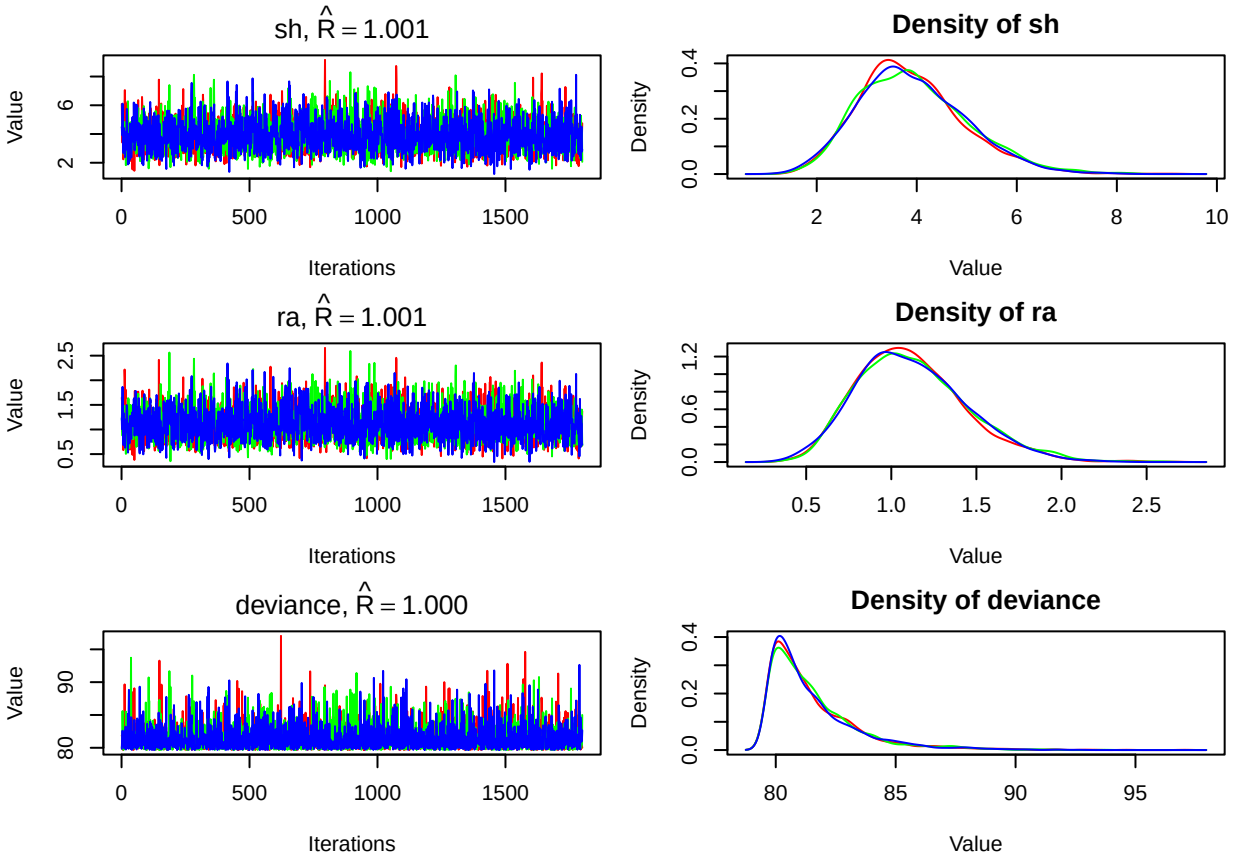
```
##
## Processing function input.......
##
## Done.
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 21
##    Unobserved stochastic nodes: 2
##    Total graph size: 26
```

```
##
## Initializing model
##
## Adaptive phase, 2000 iterations x 3 chains
## If no progress bar appears JAGS has decided not to adapt
##
##
##  Burn-in phase, 2000 iterations x 3 chains
##
##
## Sampling from joint posterior, 18000 iterations x 3 chains
##
##
## Calculating statistics.......
##
## Done.
```

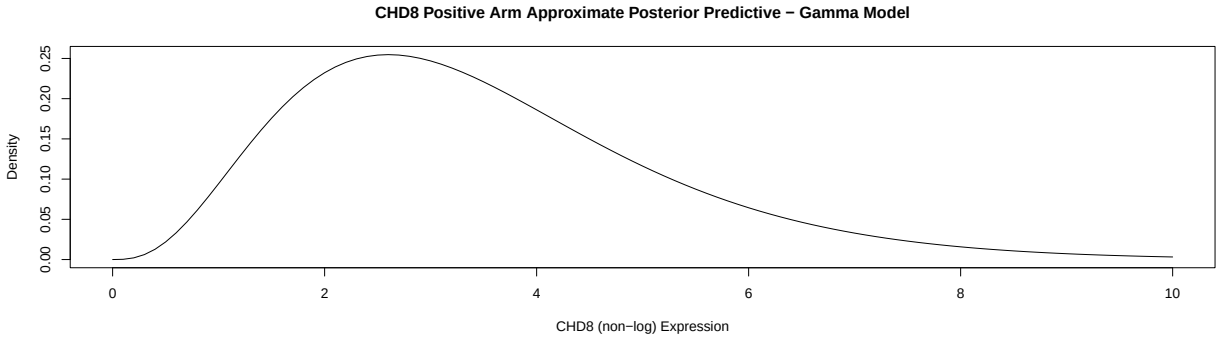
```
print(chd.pos.gamma)
```

```
## JAGS output for model '5', generated by jagsUI.
## Estimates based on 3 chains of 20000 iterations,
## adaptation = 2000 iterations (sufficient),
## burn-in = 2000 iterations and thin rate = 10,
## yielding 5400 total samples from the joint posterior.
## MCMC ran for 0.03 minutes at time 2025-02-24 17:52:03.521113.
##
##             mean    sd   2.5%    50%  97.5% overlap0 f  Rhat n.eff
## sh         3.919 1.066  2.137  3.803  6.258    FALSE 1 1.001  3355
## ra         1.122 0.321  0.592  1.091  1.839    FALSE 1 1.001  2894
## deviance 81.512 1.877 79.702 80.940 86.810    FALSE 1 1.000  5400
##
## Successful convergence based on Rhat values (all < 1.1).
## Rhat is the potential scale reduction factor (at convergence, Rhat=1).
## For each parameter, n.eff is a crude measure of effective sample size.
##
## overlap0 checks if 0 falls in the parameter's 95% credible interval.
## f is the proportion of the posterior with the same sign as the mean;
## i.e., our confidence that the parameter is positive or negative.
##
## DIC info: (pD = var(deviance)/2)
## pD = 1.8 and DIC = 83.275
## DIC is an estimate of expected predictive error (lower is better).
```

```
plot(chd.pos.gamma)
```




```
sh.chain.chd.pos <- chd.pos.gamma$sims.list$sh
ra.chain.chd.pos <- chd.pos.gamma$sims.list$ra

curve(dgamma(x, mean(sh.chain.chd.pos), mean(ra.chain.chd.pos)), from = 0, to = 10, xlab = "CHD8 (non-l
```



Modeling CHD8 Negative Arm using a Normal Distribution

```
chd.neg.normal <- "model {

  for (i in 1:N){
  x[i] ~ dnorm(mu, tau)
  }
```

```
mu ~ dnorm(0,1)
tau <- pow(sigma, -2)
sigma ~ dunif(0,2)
}
"

# Create reproducibility list - jags requires us to set the random number
#   generator type and seed for EACH chain. Mersenne-Twister is the default R RNG.
num_chains <- 3
init_vals <- vector("list",length=num_chains)
for(i in 1:num_chains){
  init_vals[[i]]$.RNG.name="base::Mersenne-Twister"
  init_vals[[i]]$.RNG.seed=117+i
}

chd.neg.Bayes <- jags(data = list(x = as.vector(chd.neg),N=length(chd.neg)),
                      inits = init_vals, parameters.to.save = c("mu", "sigma"),
                      model.file = textConnection(chd.neg.normal), n.chains=num_chains,
                      n.iter=20000, n.burnin = 2000, n.adapt = 2000, n.thin=10)
```

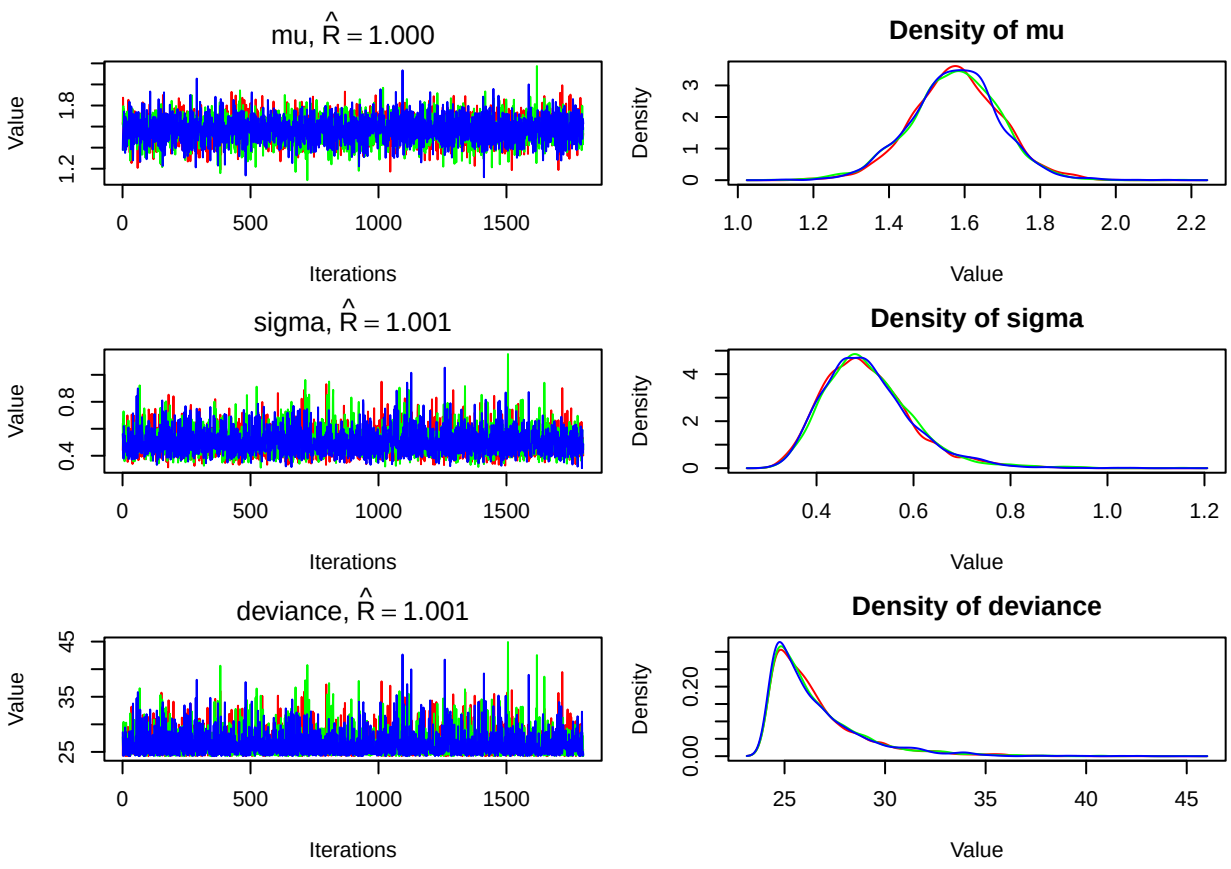
```
##
## Processing function input.......
##
## Done.
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 19
##    Unobserved stochastic nodes: 2
##    Total graph size: 27
##
## Initializing model
##
## Adaptive phase, 2000 iterations x 3 chains
## If no progress bar appears JAGS has decided not to adapt
##
##
##  Burn-in phase, 2000 iterations x 3 chains
##
##
## Sampling from joint posterior, 18000 iterations x 3 chains
##
##
## Calculating statistics.......
##
## Done.
```

```
print(chd.neg.Bayes)
```

```
## JAGS output for model '6', generated by jagsUI.
## Estimates based on 3 chains of 20000 iterations,
## adaptation = 2000 iterations (sufficient),
```

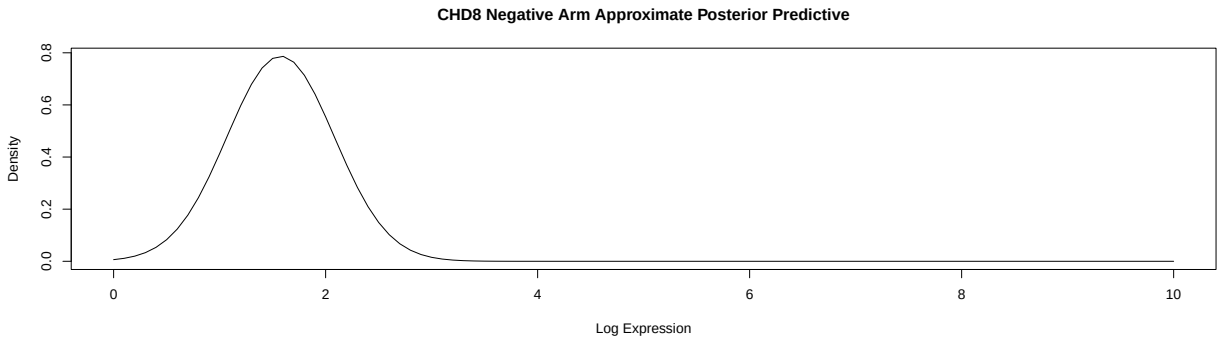
```
## burn-in = 2000 iterations and thin rate = 10,
## yielding 5400 total samples from the joint posterior.
## MCMC ran for 0.006 minutes at time 2025-02-24 17:52:05.414584.
##
##             mean    sd   2.5%    50%  97.5% overlap0 f  Rhat n.eff
## mu         1.575 0.117  1.344  1.577  1.803    FALSE 1 1.000  4366
## sigma      0.507 0.093  0.363  0.494  0.728    FALSE 1 1.001  2955
## deviance 26.501 2.289 24.288 25.794 32.783    FALSE 1 1.001  5400
##
## Successful convergence based on Rhat values (all < 1.1).
## Rhat is the potential scale reduction factor (at convergence, Rhat=1).
## For each parameter, n.eff is a crude measure of effective sample size.
##
## overlap0 checks if 0 falls in the parameter's 95% credible interval.
## f is the proportion of the posterior with the same sign as the mean;
## i.e., our confidence that the parameter is positive or negative.
##
## DIC info: (pD = var(deviance)/2)
## pD = 2.6 and DIC = 29.121
## DIC is an estimate of expected predictive error (lower is better).
```

```
plot(chd.neg.Bayes)
```




```
# extract samples simulated from posterior for mu and sigma

mu.chain.chd.neg <- chd.neg.Bayes$sims.list$mu
sigma.chain.chd.neg <- chd.neg.Bayes$sims.list$sigma
```

```
curve(dnorm(x, mean(mu.chain.chd.neg), mean(sigma.chain.chd.neg)), from = 0, to = 10, xlab = "Log Expres
```



Modeling CHD8 Positive Arm using a Normal Distribution

```
chd.pos.normal <- "model {

  for (i in 1:N){
  x[i] ~ dnorm(mu, tau)
  }

mu ~ dnorm(0,1)
tau <- pow(sigma, -2)
sigma ~ dunif(0,2)
}
"


chd.pos.Bayes <- jags(data = list(x = as.vector(chd.pos),N=length(chd.pos)),
                      inits = init_vals, parameters.to.save = c("mu", "sigma"),
                      model.file = textConnection(chd.pos.normal), n.chains=num_chains,
                      n.iter=20000, n.burnin = 2000, n.adapt = 2000, n.thin=10)
```

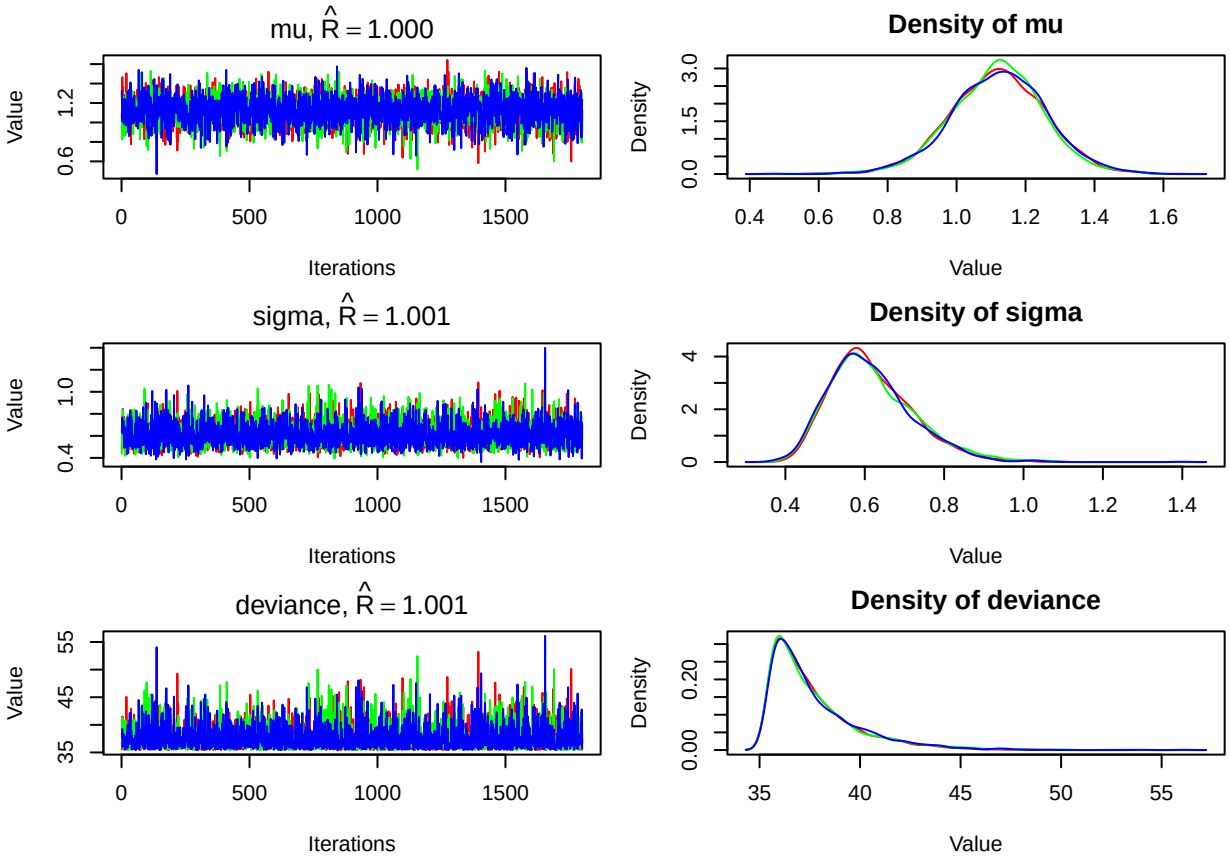
```
##
## Processing function input.......
##
## Done.
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 21
##    Unobserved stochastic nodes: 2
##    Total graph size: 29
##
## Initializing model
##
## Adaptive phase, 2000 iterations x 3 chains
## If no progress bar appears JAGS has decided not to adapt
##
```

```
##
##  Burn-in phase, 2000 iterations x 3 chains
##
##
## Sampling from joint posterior, 18000 iterations x 3 chains
##
##
## Calculating statistics.......
##
## Done.
```

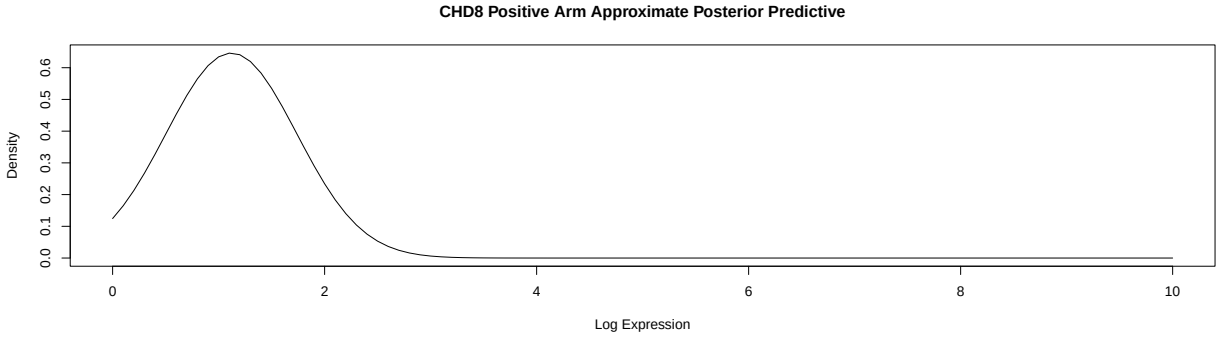
```
print(chd.pos.Bayes)
```

```
## JAGS output for model '7', generated by jagsUI.
## Estimates based on 3 chains of 20000 iterations,
## adaptation = 2000 iterations (sufficient),
## burn-in = 2000 iterations and thin rate = 10,
## yielding 5400 total samples from the joint posterior.
## MCMC ran for 0.007 minutes at time 2025-02-24 17:52:05.879236.
##
##            mean    sd   2.5%    50%  97.5% overlap0 f  Rhat n.eff
## mu        1.120 0.136  0.843  1.123  1.384    FALSE 1 1.000  5045
## sigma     0.617 0.106  0.452  0.601  0.859    FALSE 1 1.001  2498
## deviance 37.679 2.226 35.488 36.993 43.618    FALSE 1 1.001  5400
##
## Successful convergence based on Rhat values (all < 1.1).
## Rhat is the potential scale reduction factor (at convergence, Rhat=1).
## For each parameter, n.eff is a crude measure of effective sample size.
##
## overlap0 checks if 0 falls in the parameter's 95% credible interval.
## f is the proportion of the posterior with the same sign as the mean;
## i.e., our confidence that the parameter is positive or negative.
##
## DIC info: (pD = var(deviance)/2)
## pD = 2.5 and DIC = 40.156
## DIC is an estimate of expected predictive error (lower is better).
```

```
plot(chd.pos.Bayes)
```




```
# extract samples simulated from posterior for mu and sigma

mu.chain.chd.pos <- chd.pos.Bayes$sims.list$mu
sigma.chain.chd.pos <- chd.pos.Bayes$sims.list$sigma

curve(dnorm(x, mean(mu.chain.chd.pos), mean(sigma.chain.chd.pos)), from = 0, to = 10, xlab = "Log Expres
```



DNAJC6

```
dnajc.neg <- exprs(GSE32063_eset)["DNAJC6",][which(YY == 0)]
dnajc.pos <- exprs(GSE32063_eset)["DNAJC6",][which(YY == 1)]
```

plot empirical density curves to determine prior

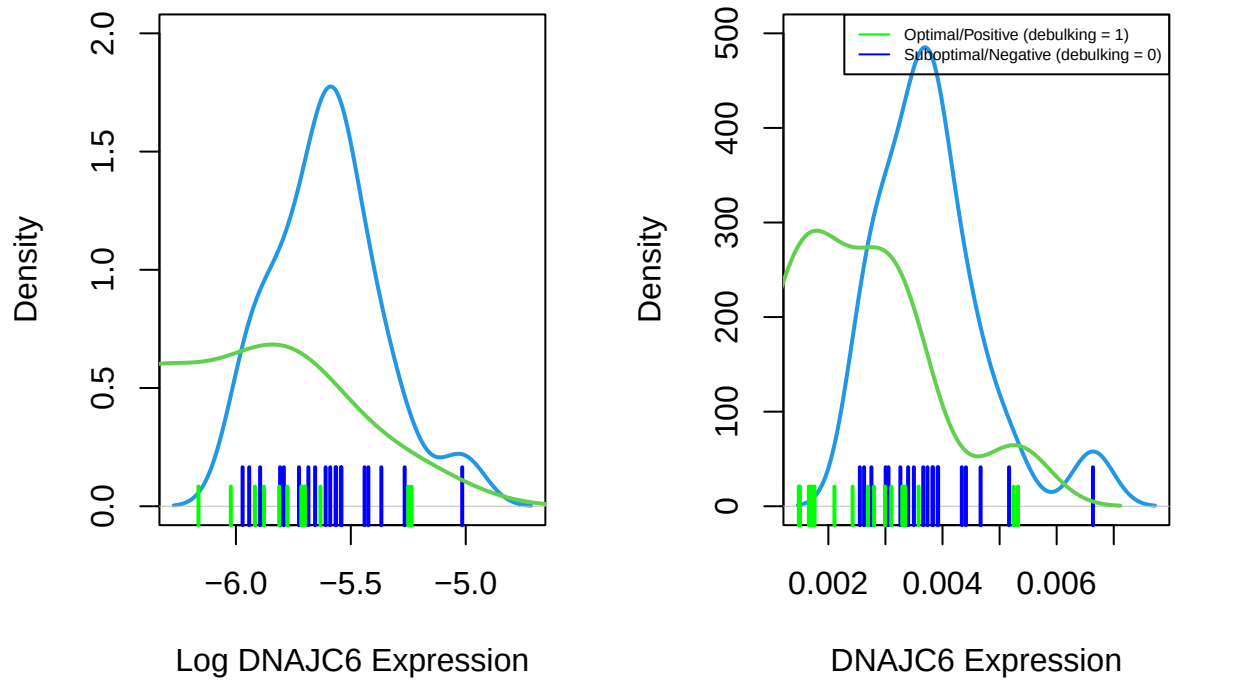
```
par(mfrow = c(1,2))
plot(density(dnajc.neg), xlab = "Log DNAJC6 Expression", col = 4, lwd = 2,
     main = "Empirical Density Curves of CHD8 Positive and Negative Cases",
     ylim = c(0, 2))
rug(dnajc.neg,ticksize = .15,lwd=2, col = "blue")
lines(density(dnajc.pos),xlab="Log DNAJC6 Expression",col=3,lwd=2,main="")
rug(dnajc.pos,ticksize = .1,lwd=2, col = "green")
```

```
## Warning in rug(dnajc.pos, ticksize = 0.1, lwd = 2, col = "green"): some values
## will be clipped
```

```
legend(0.7, 0.8, c("Optimal/Positive (debulking = 1)",
                   "Suboptimal/Negative (debulking = 0)"),
       lty=c(1,1), col=c("green", "blue"), cex = 0.52)

# again with exp - transformed
plot(density(exp(dnajc.neg)),
     xlab = "DNAJC6 Expression",
     col = 4, lwd = 2,
     main = "Empirical Density Curves of DNAJC6 Cases (Exp-transformed)",
     ylim = c(0, 500))
rug(exp(dnajc.neg), ticksize = 0.15, lwd = 2, col = "blue")
lines(density(exp(dnajc.pos)),
      col = 3, lwd = 2)
rug(exp(dnajc.pos), ticksize = 0.1, lwd = 2, col = "green")
```

```
## Warning in rug(exp(dnajc.pos), ticksize = 0.1, lwd = 2, col = "green"): some
## values will be clipped
```

```
legend("topright",
       legend = c("Optimal/Positive (debulking = 1)",
                  "Suboptimal/Negative (debulking = 0)"),
       lty = 1, col = c("green", "blue"), cex = 0.52)
```



Like CHD8, the suboptimal/negative curve has a long right tail. Moreover, the positive curve is much more flat, also with a right tail. Observing the original data, non-log transformed, both curves have a long right tail, which could be modeled with a Gamma distribution. Since Gamma is only defined for positive values, as I learned with CHD8, I must use the non-log transformed values for `DNAJC6` expression.

Modeling DNAJC Negative Arm using a Gamma Distribution

```
dnajc.neg.gamma <- "model{
  for (i in 1:N)
  {
  x[i] ~ dgamma(sh,ra)
  }

  #sh ~ dgamma(4, 1000) # so prior on X has E[X] shape/rate 0.004
                        # make more strongly informative to narrow posterior
  sh ~ dgamma(1, 1)   # try more uninformative to widen posterior
  ra ~ dgamma(1, 1)
}"

# use specified num_chains and init_vals as in CHD models

dnajc.neg.gamma <- jags(data = list(x = as.vector(exp(dnajc.neg)),N=length(dnajc.neg)), inits = init_va
```

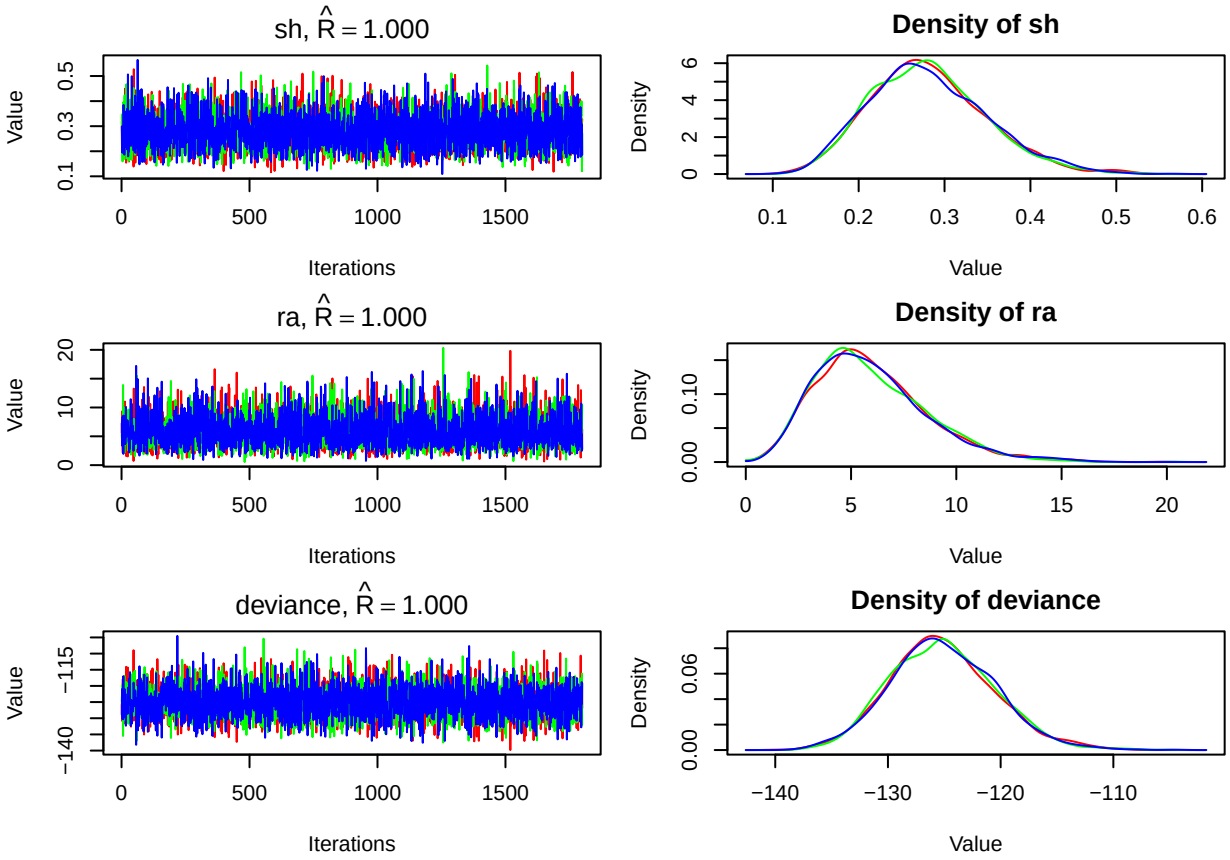
```
##
## Processing function input.......
##
```

```
## Done.
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 19
##    Unobserved stochastic nodes: 2
##    Total graph size: 23
##
## Initializing model
##
## Adaptive phase, 2000 iterations x 3 chains
## If no progress bar appears JAGS has decided not to adapt
##
##
##  Burn-in phase, 2000 iterations x 3 chains
##
##
## Sampling from joint posterior, 18000 iterations x 3 chains
##
##
## Calculating statistics.......
##
## Done.
```

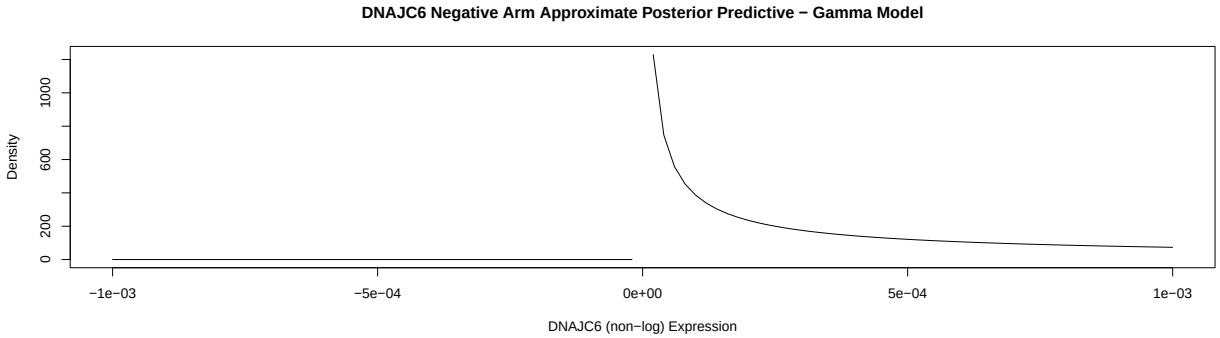
```
print(dnajc.neg.gamma)
```

```
## JAGS output for model '8', generated by jagsUI.
## Estimates based on 3 chains of 20000 iterations,
## adaptation = 2000 iterations (sufficient),
## burn-in = 2000 iterations and thin rate = 10,
## yielding 5400 total samples from the joint posterior.
## MCMC ran for 0.041 minutes at time 2025-02-24 17:52:06.413053.
##
##               mean    sd     2.5%      50%    97.5% overlap0 f Rhat n.eff
## sh           0.281 0.067    0.165    0.276    0.428    FALSE 1    1  5400
## ra           5.928 2.640    1.903    5.542   11.977    FALSE 1    1  5400
## deviance -125.127 4.632 -133.657 -125.339 -115.569    FALSE 1    1  5400
##
## Successful convergence based on Rhat values (all < 1.1).
## Rhat is the potential scale reduction factor (at convergence, Rhat=1).
## For each parameter, n.eff is a crude measure of effective sample size.
##
## overlap0 checks if 0 falls in the parameter's 95% credible interval.
## f is the proportion of the posterior with the same sign as the mean;
## i.e., our confidence that the parameter is positive or negative.
##
## DIC info: (pD = var(deviance)/2)
## pD = 10.7 and DIC = -114.395
## DIC is an estimate of expected predictive error (lower is better).
```

```
plot(dnajc.neg.gamma)
```




```
sh.chain.dnajc.neg <- dnajc.neg.gamma$sims.list$sh
ra.chain.dnajc.neg <- dnajc.neg.gamma$sims.list$ra

curve(dgamma(x, mean(sh.chain.dnajc.neg), mean(ra.chain.dnajc.neg)), from = -0.001, to = 0.001, xlab =
```



Modeling DNAJC Positive Arm using a Gamma Distribution

```
dnajc.pos.gamma <- "model{
  for (i in 1:N)
  {
  x[i] ~ dgamma(sh,ra)
  }
```

```
  sh ~ dgamma(1, 500) # so prior on X has E[X] shape/rate 0.002
  ra ~ dgamma(1, 1)
}"

# use specified num_chains and init_vals as in CHD models

dnajc.pos.gamma <- jags(data = list(x = as.vector(exp(dnajc.pos)),N=length(dnajc.pos)), inits = init_va
```

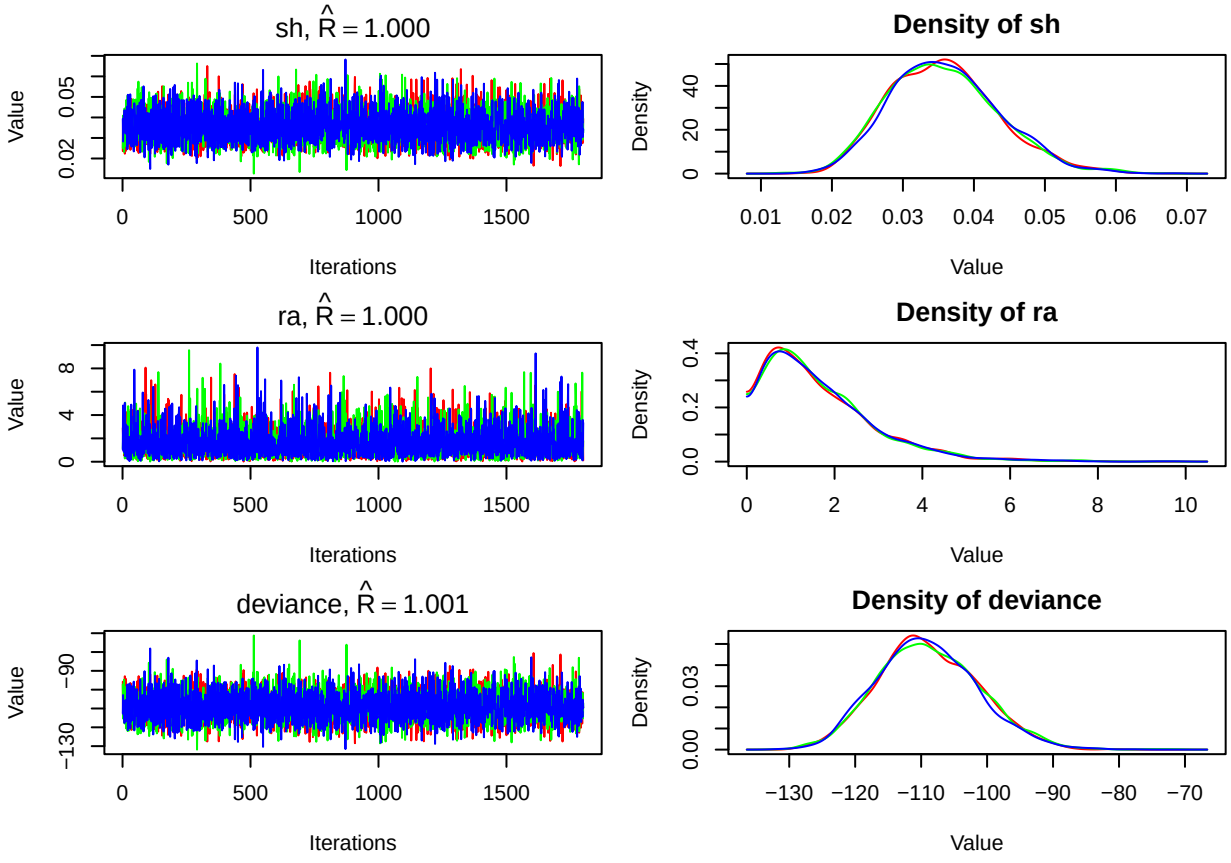
```
##
## Processing function input.......
##
## Done.
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 21
##    Unobserved stochastic nodes: 2
##    Total graph size: 26
##
## Initializing model
##
## Adaptive phase, 2000 iterations x 3 chains
## If no progress bar appears JAGS has decided not to adapt
##
##
##  Burn-in phase, 2000 iterations x 3 chains
##
##
## Sampling from joint posterior, 18000 iterations x 3 chains
##
##
## Calculating statistics.......
##
## Done.
```

```
print(dnajc.pos.gamma)
```

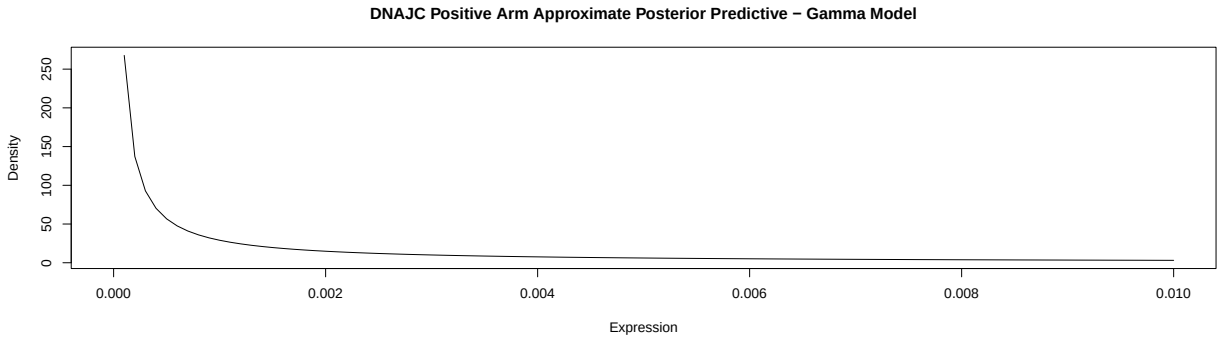
```
## JAGS output for model '9', generated by jagsUI.
## Estimates based on 3 chains of 20000 iterations,
## adaptation = 2000 iterations (sufficient),
## burn-in = 2000 iterations and thin rate = 10,
## yielding 5400 total samples from the joint posterior.
## MCMC ran for 0.045 minutes at time 2025-02-24 17:52:08.947475.
##
##               mean    sd     2.5%      50%   97.5% overlap0 f  Rhat n.eff
## sh           0.036 0.008    0.023    0.035   0.052    FALSE 1 1.000  5400
## ra           1.673 1.258    0.154    1.370   4.769    FALSE 1 1.000  5400
## deviance -108.914 7.603 -122.593 -109.377 -93.255    FALSE 1 1.001  4108
##
## Successful convergence based on Rhat values (all < 1.1).
## Rhat is the potential scale reduction factor (at convergence, Rhat=1).
## For each parameter, n.eff is a crude measure of effective sample size.
```

```
##
## overlap0 checks if 0 falls in the parameter's 95% credible interval.
## f is the proportion of the posterior with the same sign as the mean;
## i.e., our confidence that the parameter is positive or negative.
##
## DIC info: (pD = var(deviance)/2)
## pD = 28.9 and DIC = -80.016
## DIC is an estimate of expected predictive error (lower is better).
```

```
plot(dnajc.pos.gamma)
```




```
sh.chain.dnajc.pos <- dnajc.pos.gamma$sims.list$sh
ra.chain.dnajc.pos <- dnajc.pos.gamma$sims.list$ra

curve(dgamma(x, mean(sh.chain.dnajc.pos), mean(ra.chain.dnajc.pos)), from = 0, to = 0.01, xlab = "Expres
```



The posterior seems uninformative even after specifying a more informative prior for both positive and negative arms. It may be better to work with log-transformed data in a Normal Distribution. Since the shape parameter is less than 1, the gamma function behaves as $x^{\text{shape}-1}$, squishing the density near 0.

Modeling DNAJC6 Negative Arm using a Normal Distribution

```
dnajc.neg.normal <- "model {

  for (i in 1:N){
  x[i] ~ dnorm(mu, tau)
  }

mu ~ dnorm(0,1)
tau <- pow(sigma, -2)
sigma ~ dunif(0,2)
}
"


dnajc.neg.Bayes <- jags(data = list(x = as.vector(dnajc.neg),N=length(dnajc.neg)),
                        inits = init_vals, parameters.to.save = c("mu", "sigma"),
                        model.file = textConnection(dnajc.neg.normal), n.chains=num_chains,
                        n.iter=20000, n.burnin = 2000, n.adapt = 2000, n.thin=10)
```

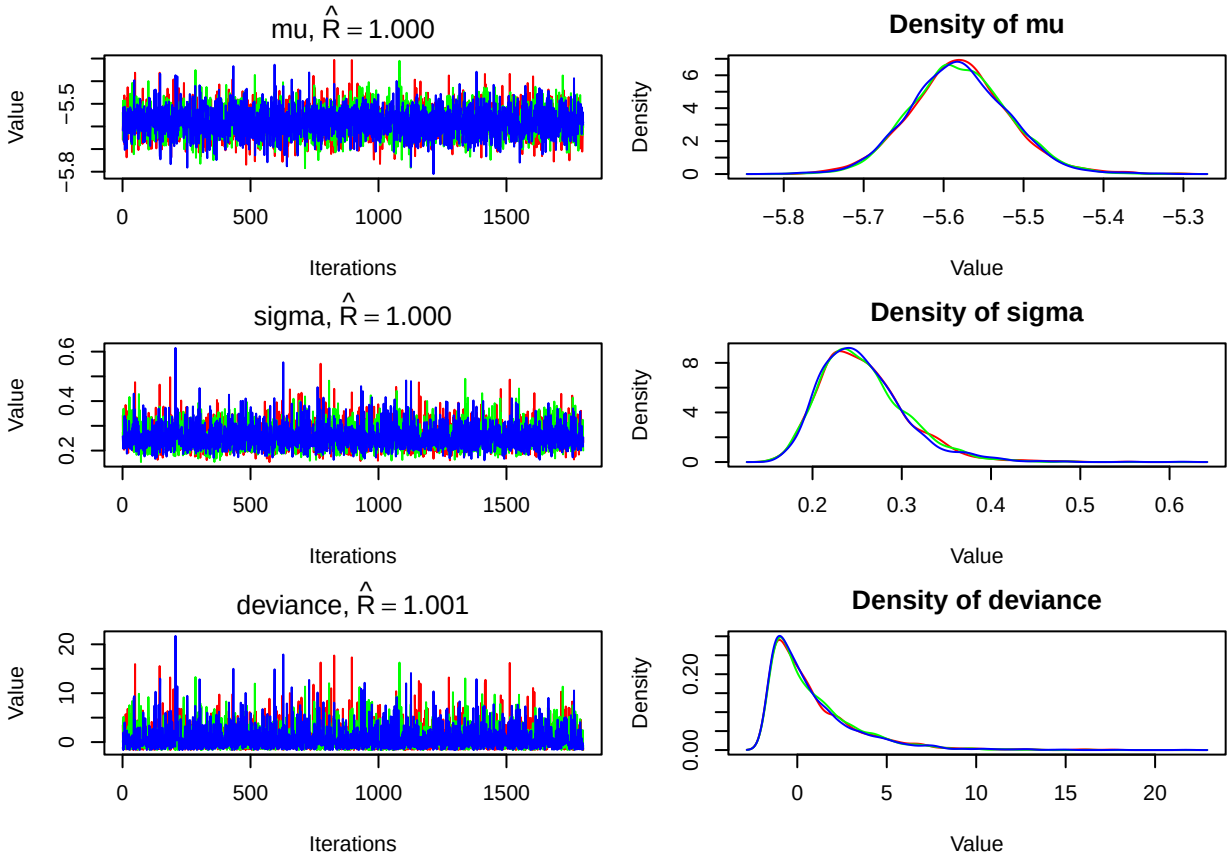
```
##
## Processing function input.......
##
## Done.
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 19
##    Unobserved stochastic nodes: 2
##    Total graph size: 27
##
## Initializing model
##
## Adaptive phase, 2000 iterations x 3 chains
## If no progress bar appears JAGS has decided not to adapt
##
```

```
##
##  Burn-in phase, 2000 iterations x 3 chains
##
##
## Sampling from joint posterior, 18000 iterations x 3 chains
##
##
## Calculating statistics.......
##
## Done.
```

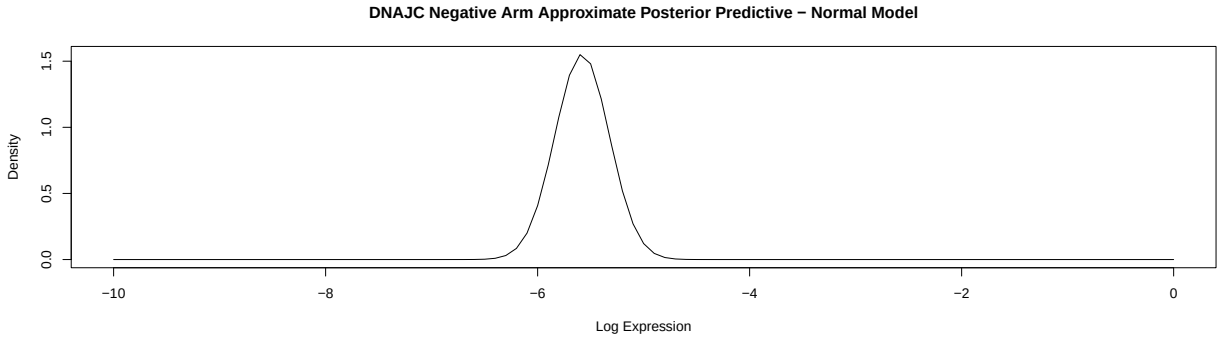
```
print(dnajc.neg.Bayes)
```

```
## JAGS output for model '10', generated by jagsUI.
## Estimates based on 3 chains of 20000 iterations,
## adaptation = 2000 iterations (sufficient),
## burn-in = 2000 iterations and thin rate = 10,
## yielding 5400 total samples from the joint posterior.
## MCMC ran for 0.006 minutes at time 2025-02-24 17:52:11.734437.
##
##             mean    sd   2.5%    50%  97.5% overlap0     f  Rhat n.eff
## mu       -5.580 0.061 -5.698 -5.582 -5.456    FALSE 1.000 1.000  5400
## sigma     0.257 0.048  0.183  0.250  0.370    FALSE 1.000 1.000  5400
## deviance  0.814 2.441 -1.558  0.063  7.215     TRUE 0.511 1.001  5400
##
## Successful convergence based on Rhat values (all < 1.1).
## Rhat is the potential scale reduction factor (at convergence, Rhat=1).
## For each parameter, n.eff is a crude measure of effective sample size.
##
## overlap0 checks if 0 falls in the parameter's 95% credible interval.
## f is the proportion of the posterior with the same sign as the mean;
## i.e., our confidence that the parameter is positive or negative.
##
## DIC info: (pD = var(deviance)/2)
## pD = 3 and DIC = 3.794
## DIC is an estimate of expected predictive error (lower is better).
```

```
plot(dnajc.neg.Bayes)
```




```
mu.chain.dnajc.neg <- dnajc.neg.Bayes$sims.list$mu
sigma.chain.dnajc.neg <- dnajc.neg.Bayes$sims.list$sigma

curve(dnorm(x, mean(mu.chain.dnajc.neg), mean(sigma.chain.dnajc.neg)), from = -10, to = 0, xlab = "Log
```



Modeling DNAJC6 Positive Arm using a Normal Distribution

```
dnajc.pos.normal <- "model {

  for (i in 1:N){
  x[i] ~ dnorm(mu, tau)
  }
```

```
mu ~ dnorm(0,1)
tau <- pow(sigma, -2)
sigma ~ dunif(0,2)
}
"


dnajc.pos.Bayes <- jags(data = list(x = as.vector(dnajc.pos),N=length(dnajc.pos)),
                        inits = init_vals, parameters.to.save = c("mu", "sigma"),
                        model.file = textConnection(dnajc.pos.normal), n.chains=num_chains,
                        n.iter=20000, n.burnin = 2000, n.adapt = 2000, n.thin=10)
```

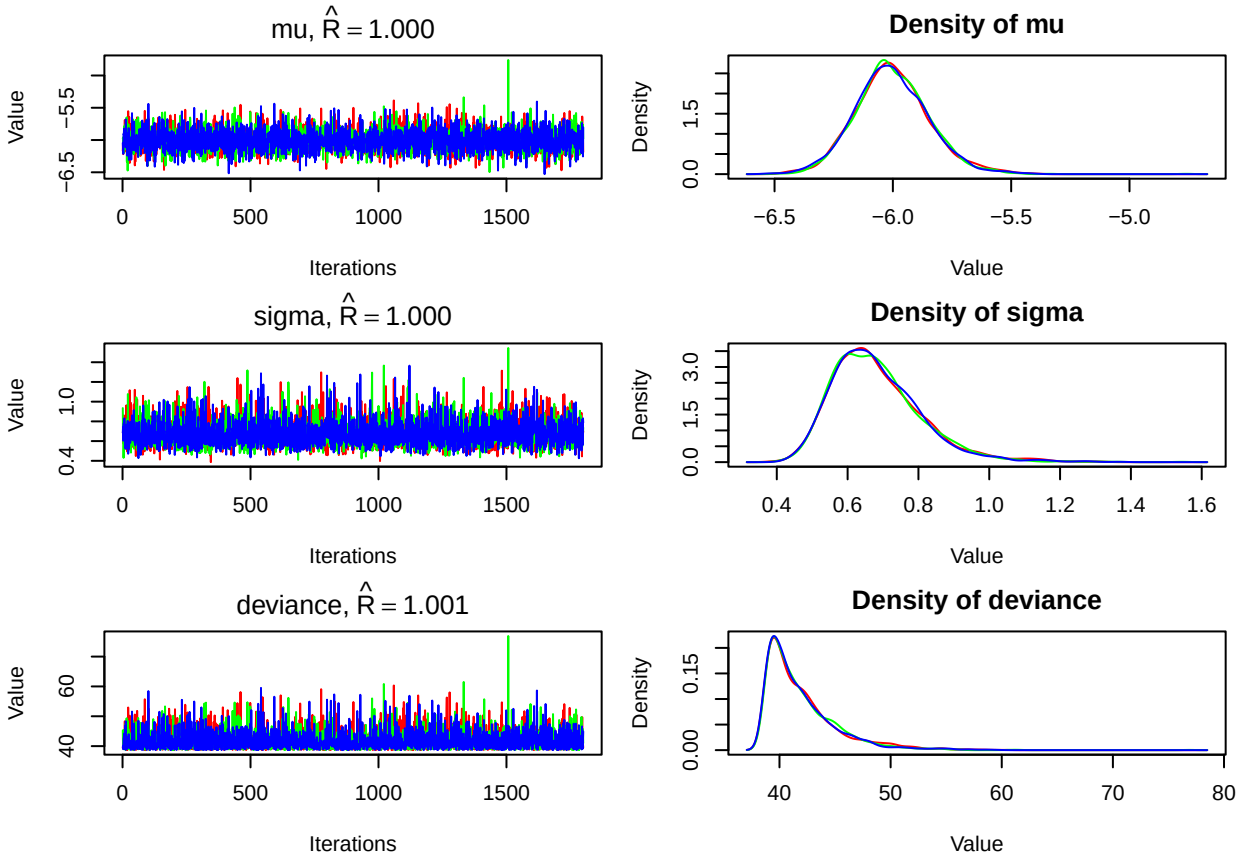
```
##
## Processing function input.......
##
## Done.
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 21
##    Unobserved stochastic nodes: 2
##    Total graph size: 29
##
## Initializing model
##
## Adaptive phase, 2000 iterations x 3 chains
## If no progress bar appears JAGS has decided not to adapt
##
##
##  Burn-in phase, 2000 iterations x 3 chains
##
##
## Sampling from joint posterior, 18000 iterations x 3 chains
##
##
## Calculating statistics.......
##
## Done.
```

```
print(dnajc.pos.Bayes)
```

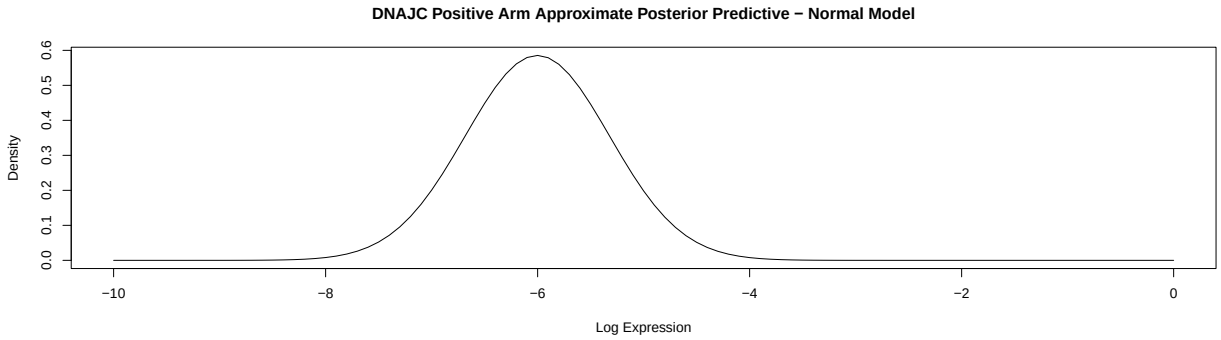
```
## JAGS output for model '11', generated by jagsUI.
## Estimates based on 3 chains of 20000 iterations,
## adaptation = 2000 iterations (sufficient),
## burn-in = 2000 iterations and thin rate = 10,
## yielding 5400 total samples from the joint posterior.
## MCMC ran for 0.007 minutes at time 2025-02-24 17:52:12.195621.
##
##              mean    sd   2.5%    50%  97.5% overlap0 f  Rhat n.eff
## mu         -6.003 0.155 -6.292 -6.011 -5.678    FALSE 1 1.000  5115
## sigma       0.681 0.126  0.493  0.663  0.978    FALSE 1 1.000  5400
## deviance 41.892 3.156 38.752 40.960 50.306    FALSE 1 1.001  4886
```

```
##
## Successful convergence based on Rhat values (all < 1.1).
## Rhat is the potential scale reduction factor (at convergence, Rhat=1).
## For each parameter, n.eff is a crude measure of effective sample size.
##
## overlap0 checks if 0 falls in the parameter's 95% credible interval.
## f is the proportion of the posterior with the same sign as the mean;
## i.e., our confidence that the parameter is positive or negative.
##
## DIC info: (pD = var(deviance)/2)
## pD = 5 and DIC = 46.872
## DIC is an estimate of expected predictive error (lower is better).
```

```
plot(dnajc.pos.Bayes)
```




```
mu.chain.dnajc.pos <- dnajc.pos.Bayes$sims.list$mu
sigma.chain.dnajc.pos <- dnajc.pos.Bayes$sims.list$sigma

curve(dnorm(x, mean(mu.chain.dnajc.pos), mean(sigma.chain.dnajc.pos)), from = -10, to = 0, xlab = "Log
```



because of the extremely low shape paremeter and long right tail of the posterior distribution in the Gamma DNAJC, I will attempt another model with the aim of better capturing low values of gene expression with a Weibull Distribution

Modeling DNAJC6 Negative Arm using a Weibull Distribution (FAILED)

It seems that a Weibull distribution faced similar problems to the Gamma with high density around 0. Will try shifting the gamma distribution to maintain larger spread from log-transformed data for lower average expression values.

Modeling DNAJC6 Negative Arm using a SHIFTED Gamma Distribution

```
dnajc.neg.gamma2 <- "model{
  for (i in 1:N)
  {
  x[i] ~ dgamma(sh,ra)
  }

  #sh ~ dgamma(4, 1000) # so prior on X has E[X] shape/rate 0.004
                        # make more strongly informative to narrow posterior
  sh ~ dgamma(1, 1)   # try more uninformative to widen posterior
  ra ~ dgamma(1, 1)
}"

# use specified num_chains and init_vals as in CHD models
# add shift plus perturbance 0.5 s.t. values are all > 0

dnajc.neg.gamma2 <- jags(data = list(x = as.vector((dnajc.neg - min(dnajc.neg) +  0.5)),N=length(dnajc.
```

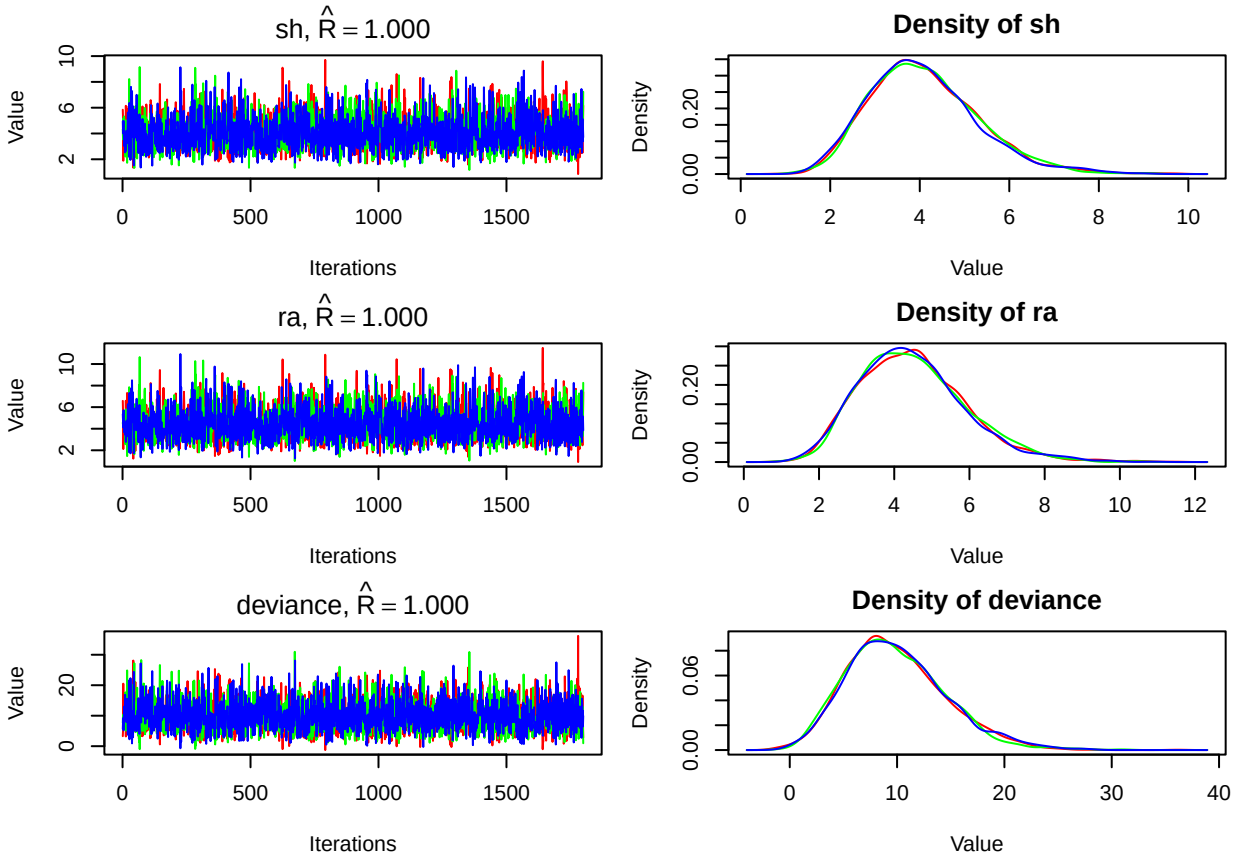
```
##
## Processing function input.......
##
## Done.
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 19
##    Unobserved stochastic nodes: 2
##    Total graph size: 23
##
```

```
## Initializing model
##
## Adaptive phase, 2000 iterations x 3 chains
## If no progress bar appears JAGS has decided not to adapt
##
##
##  Burn-in phase, 2000 iterations x 3 chains
##
##
## Sampling from joint posterior, 18000 iterations x 3 chains
##
##
## Calculating statistics.......
##
## Done.
```

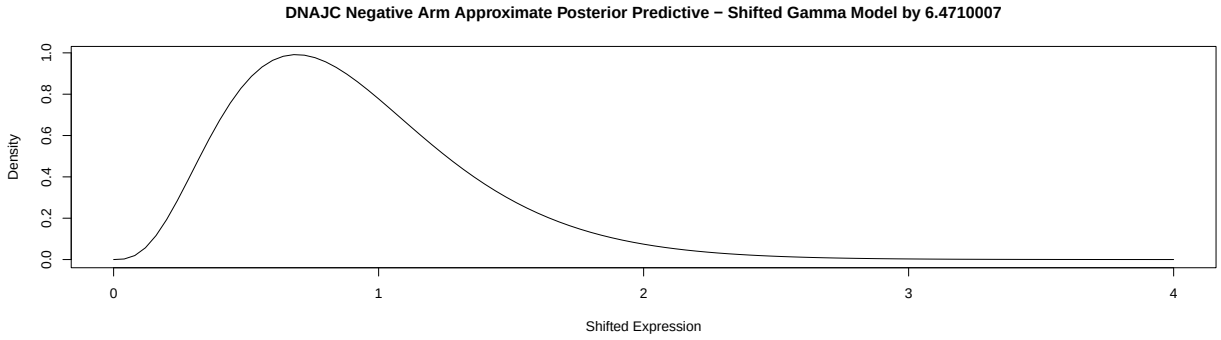
```
print(dnajc.neg.gamma2)
```

```
## JAGS output for model '12', generated by jagsUI.
## Estimates based on 3 chains of 20000 iterations,
## adaptation = 2000 iterations (sufficient),
## burn-in = 2000 iterations and thin rate = 10,
## yielding 5400 total samples from the joint posterior.
## MCMC ran for 0.026 minutes at time 2025-02-24 17:52:12.703173.
##
##           mean    sd  2.5%   50%  97.5% overlap0     f Rhat n.eff
## sh       4.099 1.204 2.122 3.981  6.814    FALSE 1.000    1  3499
## ra       4.495 1.400 2.219 4.374  7.622    FALSE 1.000    1  5190
## deviance 9.927 4.584 2.417 9.448 20.202    FALSE 0.998    1  3717
##
## Successful convergence based on Rhat values (all < 1.1).
## Rhat is the potential scale reduction factor (at convergence, Rhat=1).
## For each parameter, n.eff is a crude measure of effective sample size.
##
## overlap0 checks if 0 falls in the parameter's 95% credible interval.
## f is the proportion of the posterior with the same sign as the mean;
## i.e., our confidence that the parameter is positive or negative.
##
## DIC info: (pD = var(deviance)/2)
## pD = 10.5 and DIC = 20.433
## DIC is an estimate of expected predictive error (lower is better).
```

```
plot(dnajc.neg.gamma2)
```




```
sh.chain.dnajc.neg2 <- dnajc.neg.gamma2$sims.list$sh
ra.chain.dnajc.neg2 <- dnajc.neg.gamma2$sims.list$ra

curve(dgamma(x, mean(sh.chain.dnajc.neg2), scale = 1 / mean(ra.chain.dnajc.neg2)), from = 0, to = 4, xla
```



Modeling DNAJC6 Positive Arm using a SHIFTED Gamma Distribution

```
dnajc.pos.gamma2 <- "model{
  for (i in 1:N)
  {
  x[i] ~ dgamma(sh,ra)
  }
```

```
  #sh ~ dgamma(4, 1000) # so prior on X has E[X] shape/rate 0.004
                       # make more strongly informative to narrow posterior
  sh ~ dgamma(1, 1)   # try more uninformative to widen posterior
  ra ~ dgamma(1, 1)
}"

# use specified num_chains and init_vals as in CHD models
# add shift plus perturbance 0.5 s.t. values are all > 0

dnajc.pos.gamma2 <- jags(data = list(x = as.vector((dnajc.pos - min(dnajc.pos) +  0.5)),N=length(dnajc.
```

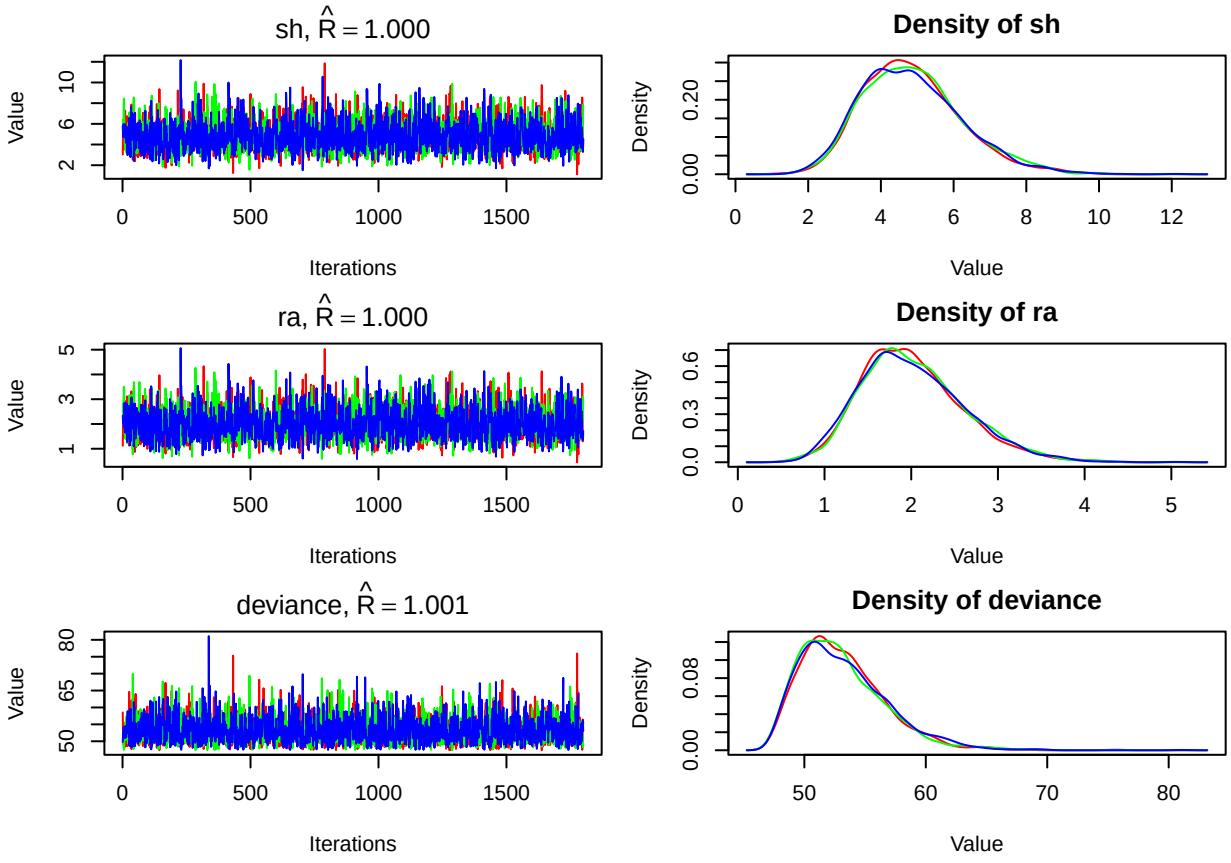
```
##
## Processing function input.......
##
## Done.
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 21
##    Unobserved stochastic nodes: 2
##    Total graph size: 25
##
## Initializing model
##
## Adaptive phase, 2000 iterations x 3 chains
## If no progress bar appears JAGS has decided not to adapt
##
##
##  Burn-in phase, 2000 iterations x 3 chains
##
##
## Sampling from joint posterior, 18000 iterations x 3 chains
##
##
## Calculating statistics.......
##
## Done.
```

```
print(dnajc.pos.gamma2)
```

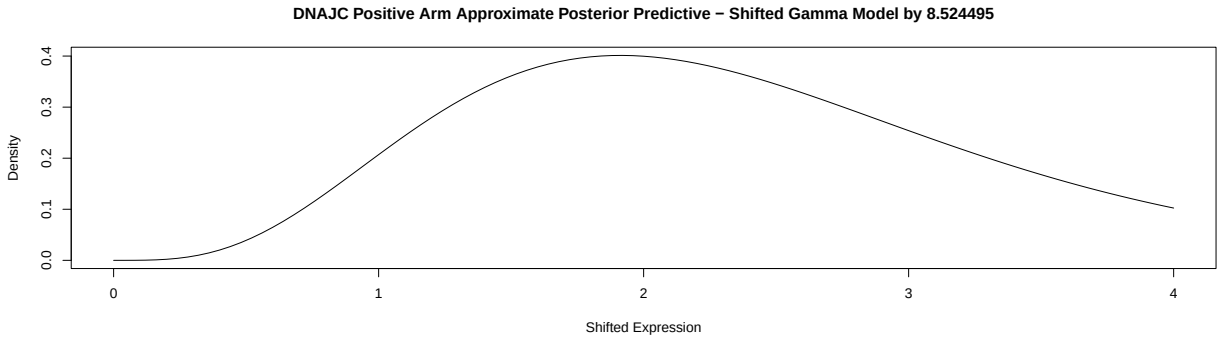
```
## JAGS output for model '13', generated by jagsUI.
## Estimates based on 3 chains of 20000 iterations,
## adaptation = 2000 iterations (sufficient),
## burn-in = 2000 iterations and thin rate = 10,
## yielding 5400 total samples from the joint posterior.
## MCMC ran for 0.03 minutes at time 2025-02-24 17:52:14.388975.
##
##             mean    sd   2.5%    50%  97.5% overlap0 f  Rhat n.eff
## sh         4.864 1.364  2.594  4.742  7.945    FALSE 1 1.000  5098
## ra         2.020 0.593  1.038  1.951  3.338    FALSE 1 1.000  4199
## deviance 53.132 3.558 48.052 52.585 61.448    FALSE 1 1.001  2458
##
```

```
## Successful convergence based on Rhat values (all < 1.1).
## Rhat is the potential scale reduction factor (at convergence, Rhat=1).
## For each parameter, n.eff is a crude measure of effective sample size.
##
## overlap0 checks if 0 falls in the parameter's 95% credible interval.
## f is the proportion of the posterior with the same sign as the mean;
## i.e., our confidence that the parameter is positive or negative.
##
## DIC info: (pD = var(deviance)/2)
## pD = 6.3 and DIC = 59.459
## DIC is an estimate of expected predictive error (lower is better).
```

```
plot(dnajc.pos.gamma2)
```



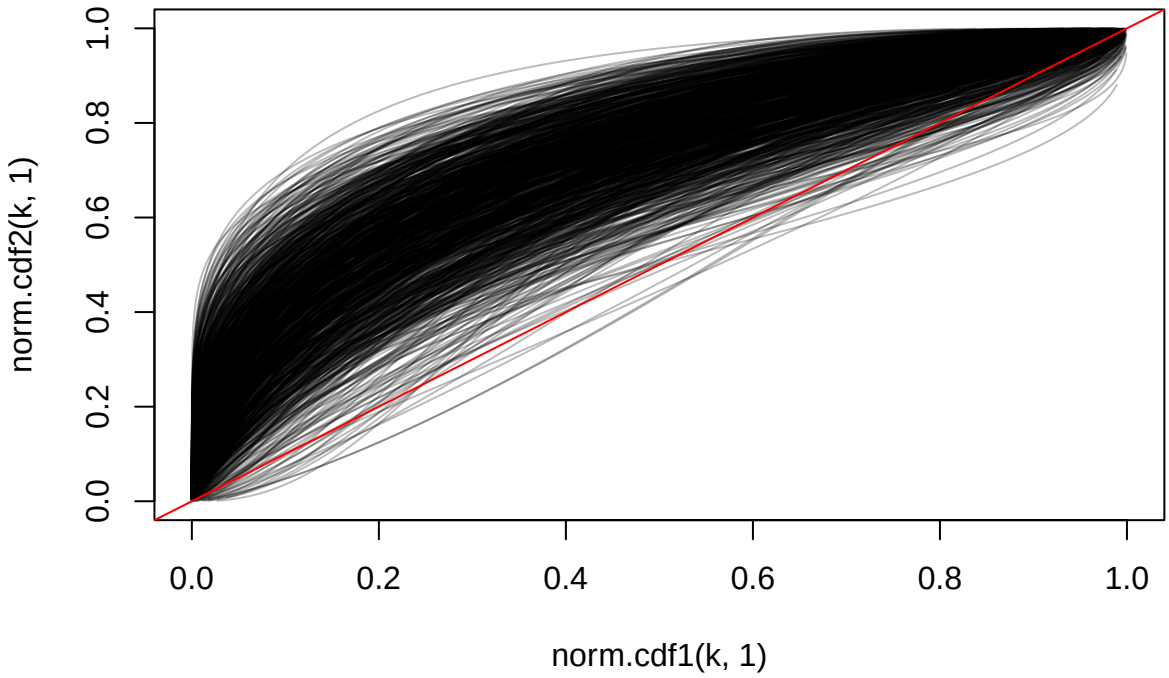

```
sh.chain.dnajc.pos2 <- dnajc.pos.gamma2$sims.list$sh
ra.chain.dnajc.pos2 <- dnajc.pos.gamma2$sims.list$ra

curve(dgamma(x, mean(sh.chain.dnajc.pos2), scale = 1 / mean(ra.chain.dnajc.pos2)), from = 0, to = 4, xla
```



2) Using an MCMC, generate a sample from the posterior distribution of the ROC curve for each biomarker, and plot it.

Then compute the probability that the area under the ROC curve (AUC) for CHD8 is higher than that for DNAJC6. Perform this analysis using each of your two alternative models (for a total of four comparisons), and give a discussion of your results.

```r
# Create functions for both CDFs
# use Normal for more informative posterior
norm.cdf1 <- function(x,n) pnorm(x, mu.chain.chd.neg[n], sd = sigma.chain.chd.neg[n])
# CDF of negative arm

norm.cdf2 <- function(x,n) pnorm(x, mu.chain.chd.pos[n], sd = sigma.chain.chd.pos[n])
# CDF of positive arm

#For ROC curves, one can take threshold values (they should correspond to min/max of the empirical dens

# Sequence of thresholds - BE CAREFUL TO USE THE RIGHT ONES ON THE PROJECT, SINCE THEY CONSIDER DIFFERE
# max chd.pos ~ 1.84
# min chd.pos ~ -0.2
# max chd.neg ~ 2.56
# min chd.neg ~ 1.02

k = seq(-0.2, 2.57, 0.01)
N.draws = 1000 # Number of parameter draws to look at
plot(norm.cdf1(k, 1), norm.cdf2(k,1), type = "l", xlim=c(0,1), ylim=c(0,1)) # Plot first ROC Curve

# cannot use plot(add = T) here since above is plot.default method.
for(i in 2:N.draws){
  lines(norm.cdf1(k,i), norm.cdf2(k,i), type = "l", col="#00000044")
}
abline(a = 0, b = 1, col = "red") # Add 45 degree line
```




```
AUC.chd <- rep(0, N.draws)
for(i in 1:N.draws){
  AUC.chd[i] <- trapz(sort(norm.cdf1(k,i)), sort(norm.cdf2(k,i)))
  # trapz calls out an approxmation using the trapezoid rule. We have to
  # sort the cdf values first!
}
mean(AUC.chd)
```

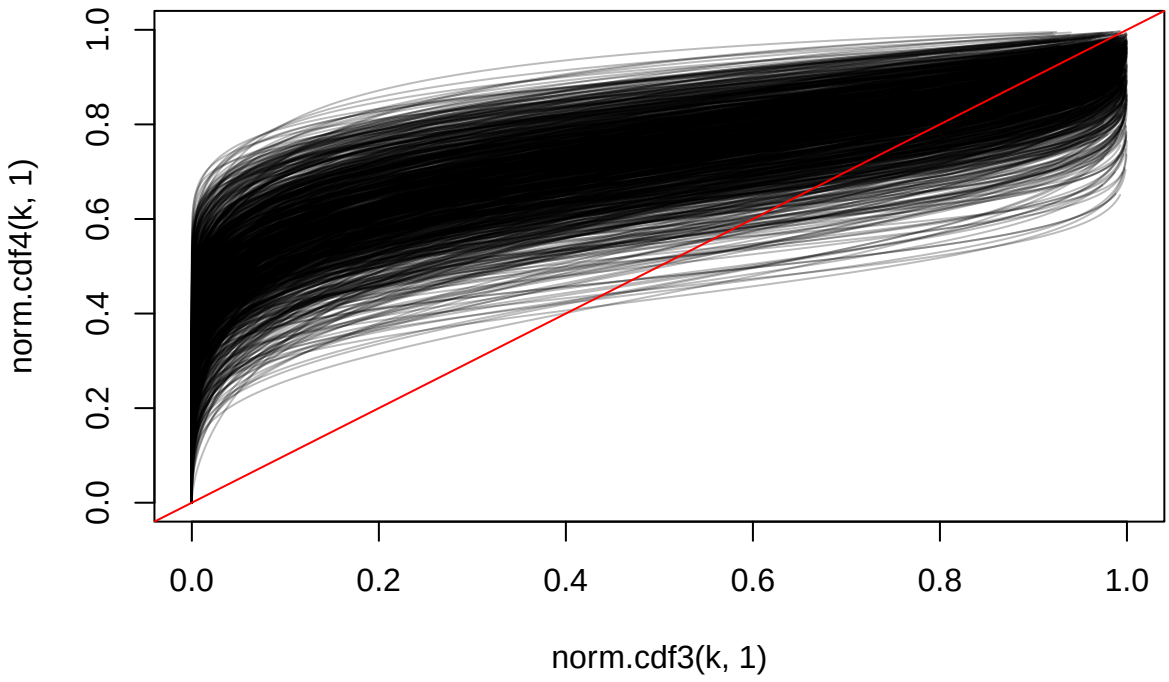
```
## [1] 0.6828119
```

```
sd(AUC.chd)
```

```
## [1] 0.07603128
```

```
# again, use Normal for more informative posterior
norm.cdf3 <- function(x,n) pnorm(x, mu.chain.dnajc.neg[n], sd = sigma.chain.dnajc.neg[n])
# CDF of negative arm

norm.cdf4 <- function(x,n) pnorm(x, mu.chain.dnajc.pos[n], sd = sigma.chain.dnajc.pos[n])

# max dnajc.pos ~ -5.24
# min dnajc.pos ~ -8.02
# max dnajc.neg ~ -5.015
# min dnajc.neg ~ -5.97

k = seq(-8.03, -5.0, 0.01)
N.draws = 1000 # Number of parameter draws to look at
plot(norm.cdf3(k, 1), norm.cdf4(k,1), type = "l", xlim=c(0,1), ylim=c(0,1)) # Plot first ROC Curve
```

```
# cannot use plot(add = T) here since above is plot.default method.
for(i in 2:N.draws){
  lines(norm.cdf3(k,i), norm.cdf4(k,i), type = "l", col="#00000044")
}
abline(a = 0, b = 1, col = "red") # Add 45 degree line
```




```
AUC.dnajc <- rep(0, N.draws)
for(i in 1:N.draws){
  AUC.dnajc[i] <- trapz(sort(norm.cdf3(k,i)), sort(norm.cdf4(k,i)))
  # trapz calls out an approxmation using the trapezoid rule. We have to
  # sort the cdf values first!
}
mean(AUC.dnajc)
```

```
## [1] 0.6998387
```

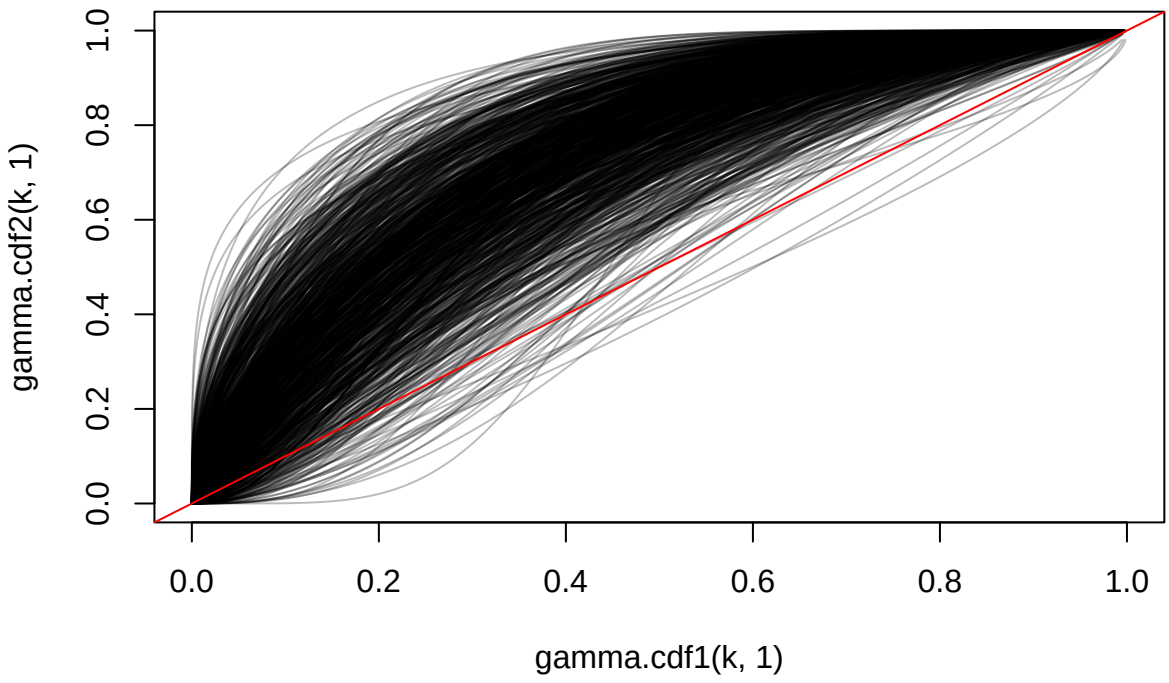
```
sd(AUC.dnajc)
```

```
## [1] 0.08523834
```

do the same for the Gamma Models

```
# Functions for both Gamma CDFS
gamma.cdf1 <- function(x,n) pgamma(x, sh.chain.chd.neg[n], rate = ra.chain.chd.neg[n])
# CDF of negative arm
```

```r
gamma.cdf2 <- function(x,n) pgamma(x, sh.chain.chd.pos[n], rate = ra.chain.chd.pos[n])
# CDF of positive arm

#For ROC curves, one can take threshold values (they should correspond to min/max of the empirical dens

# Sequence of thresholds - ALL EXPONENTIATED FOR GAMMA
# max chd.pos ~ 1.84
# min chd.pos ~ -0.2
# max chd.neg ~ 2.56
# min chd.neg ~ 1.02

k = seq(-0.8, 12.9, 0.05) # use exp() min and max values for k
N.draws = 1000 # Number of parameter draws to look at
plot(gamma.cdf1(k, 1), gamma.cdf2(k,1), type = "l", xlim=c(0,1), ylim=c(0,1)) # Plot first ROC Curve

# cannot use plot(add = T) here since above is plot.default method.
for(i in 2:N.draws){
  lines(gamma.cdf1(k,i), gamma.cdf2(k,i), type = "l", col="#00000044")
}
abline(a = 0, b = 1, col = "red") # Add 45 degree line
```




```r
AUC.chd.gamma <- rep(0, N.draws)
for(i in 1:N.draws){
  AUC.chd.gamma[i] <- trapz(sort(gamma.cdf1(k,i)), sort(gamma.cdf2(k,i)))
  # trapz calls out an approxmation using the trapezoid rule. We have to
  # sort the cdf values first!
```

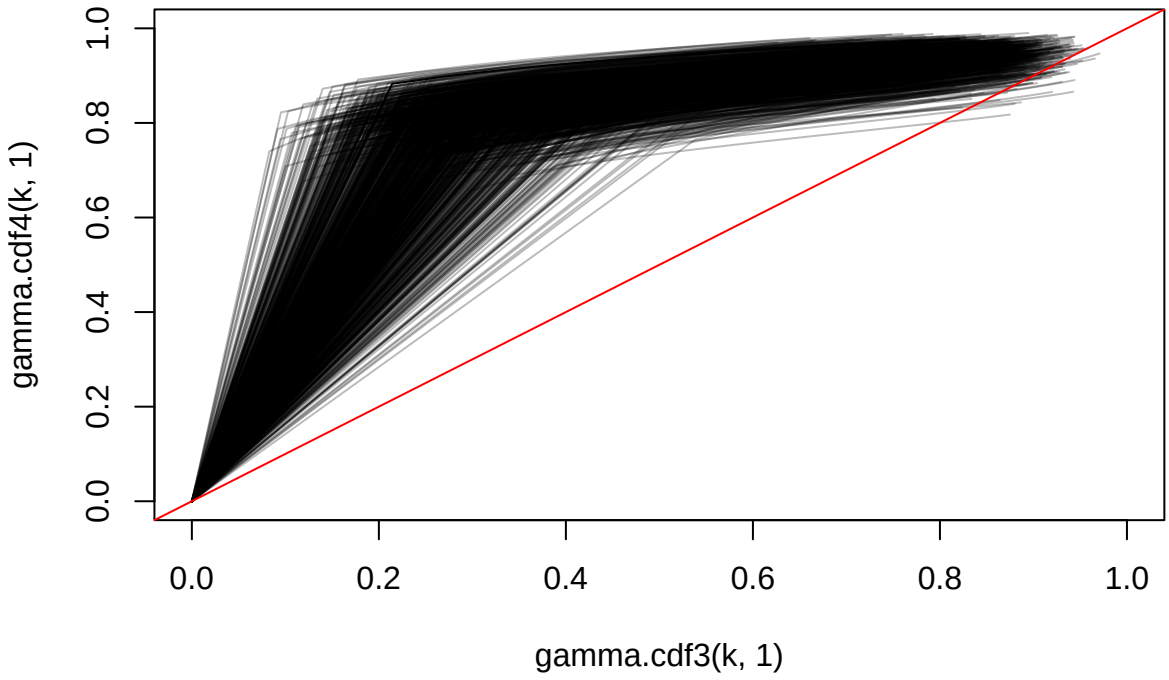
```
}
mean(AUC.chd.gamma)
```

```
## [1] 0.6886407
```

```
sd(AUC.chd.gamma)
```

```
## [1] 0.07877378
```

```
# Gamma for dnjac
gamma.cdf3 <- function(x,n) pgamma(x, sh.chain.dnajc.neg[n], rate = ra.chain.dnajc.neg[n])
# CDF of negative arm

gamma.cdf4 <- function(x,n) pgamma(x, sh.chain.dnajc.pos[n], rate = ra.chain.dnajc.pos[n])

# all EXPONENTIATED
# max dnajc.pos ~ 0.005
# min dnajc.pos ~ 0.0003
# max dnajc.neg ~ 0.0066
# min dnajc.neg ~ 0.0025

k = seq(0, 0.1, 0.001) # go beyond max since Gamma dist. has a long tail
N.draws = 1000 # Number of parameter draws to look at
plot(gamma.cdf3(k, 1), gamma.cdf4(k,1), type = "l", xlim=c(0,1), ylim=c(0,1)) # Plot first ROC Curve

# cannot use plot(add = T) here since above is plot.default method.
for(i in 2:N.draws){
  lines(gamma.cdf3(k,i), gamma.cdf4(k,i), type = "l", col="#00000044")
}
abline(a = 0, b = 1, col = "red") # Add 45 degree line
```

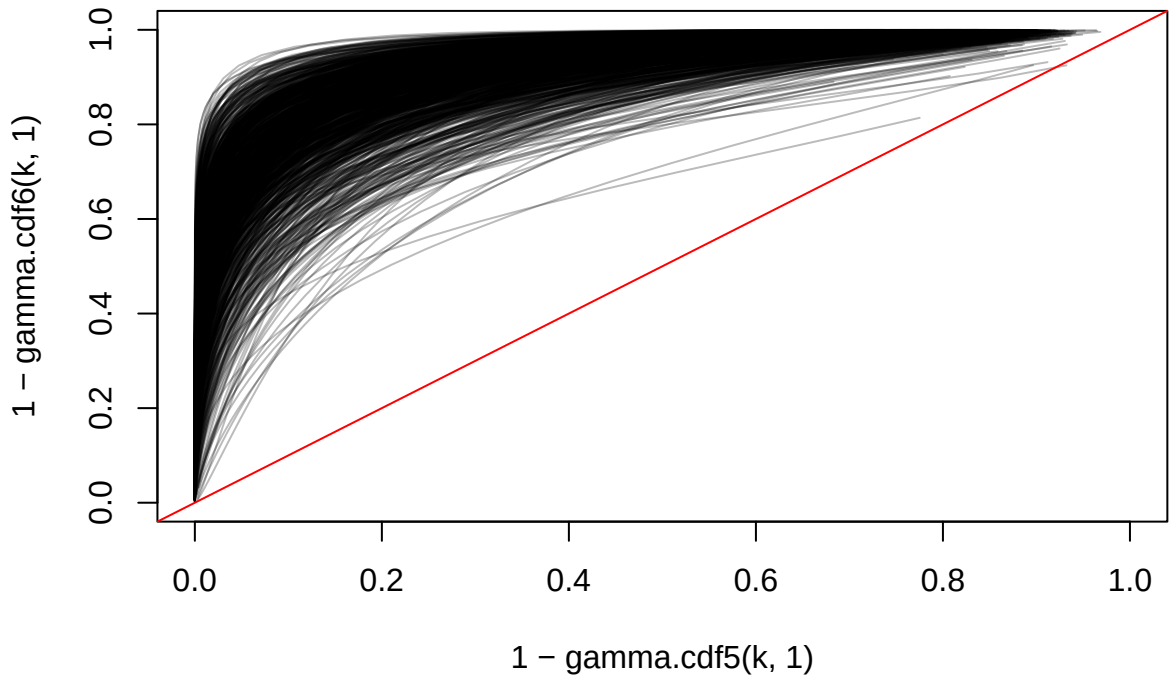



```
# AUC.dnajc <- rep(0, N.draws)
# for(i in 1:N.draws){
#   AUC.dnajc[i] <- trapz(sort(norm.cdf3(k,i)), sort(norm.cdf4(k,i)))
#   # trapz calls out an approxmation using the trapezoid rule. We have to
#   # sort the cdf values first!
# }
# mean(AUC.dnajc)
# sd(AUC.dnajc)
```

Try Shifted Gamma models for DNAJC6

```
# Gamma for dnjac
gamma.cdf5 <- function(x,n) pgamma(x, sh.chain.dnajc.neg2[n], rate = ra.chain.dnajc.neg2[n])
# CDF of negative arm

gamma.cdf6 <- function(x,n) pgamma(x, sh.chain.dnajc.pos2[n], rate = ra.chain.dnajc.pos2[n])

# all SHIFTED
# max max(dnajc.pos) - min(dnajc.pos) + 0.5 ~ 3.2
# min min(dnajc.pos) - min(dnajc.pos) + 0.5 = 0.5
# max max(dnajc.neg) - min(dnajc.neg) + 0.5 = 1.45
# min min(dnajc.neg) - min(dnajc.neg) + 0.5 = 0.5

k = seq(0.5, 5, 0.1) # go beyond max since Gamma dist. has a long tail
N.draws = 1000 # Number of parameter draws to look at
plot(1-gamma.cdf5(k, 1), 1-gamma.cdf6(k,1), type = "l", xlim=c(0,1), ylim=c(0,1)) # Plot first ROC Curv
```

```
# cannot use plot(add = T) here since above is plot.default method.
for(i in 2:N.draws){
  lines(1-gamma.cdf5(k,i), 1-gamma.cdf6(k,i), type = "l", col="#00000044")
}
abline(a = 0, b = 1, col = "red") # Add 45 degree line
```




```
AUC.dnajc.gam <- rep(0, N.draws)
for(i in 1:N.draws){
  AUC.dnajc.gam[i] <- trapz(sort(1-gamma.cdf5(k,i)), sort(1-gamma.cdf6(k,i)))
  # trapz calls out an approxmation using the trapezoid rule. We have to
  # sort the cdf values first!
}
mean(AUC.dnajc.gam)
```

```
## [1] 0.7273247
```

```
sd(AUC.dnajc.gam)
```

```
## [1] 0.07181626
```

Comparing CHD8 models

```
mean(AUC.chd<AUC.chd.gamma)
```

```
## [1] 0.515
```

Comparing DNAJC6 models

```
mean(AUC.dnajc<AUC.dnajc.gam)
```

```
## [1] 0.607
```

Comparing CHD8 and DNAJC6 as Biomarkers

```
mean(AUC.chd<AUC.dnajc)
```

```
## [1] 0.568
```

There is a a roughly 56.8% chance that DNAJC6 has a higher AUC.

```
mean(AUC.chd.gamma<AUC.dnajc.gam)
```

```
## [1] 0.649
```

Biomarker CHD8 will have a lower AUC than DNAJC6 64.9% of the time according to the two Gamma models. That being said, the two Gamma models were evaluated differently. Because of the higher expression of CHD8 as measured by microarray analysis, the true values were able to be used for the Gamma model. On the other hand, DNAJC6 had magnitudes lower expression for both classes of optimal and suboptimal debulking, making it unfeasible to exponentiate the log-transformed expression values, as the low shape parameter lead to an inaccurate posterior distribution with density squished near 0 and poor AUC. As follows, a shift was necessary to coerce a reasonable shape value for a more centered posterior.

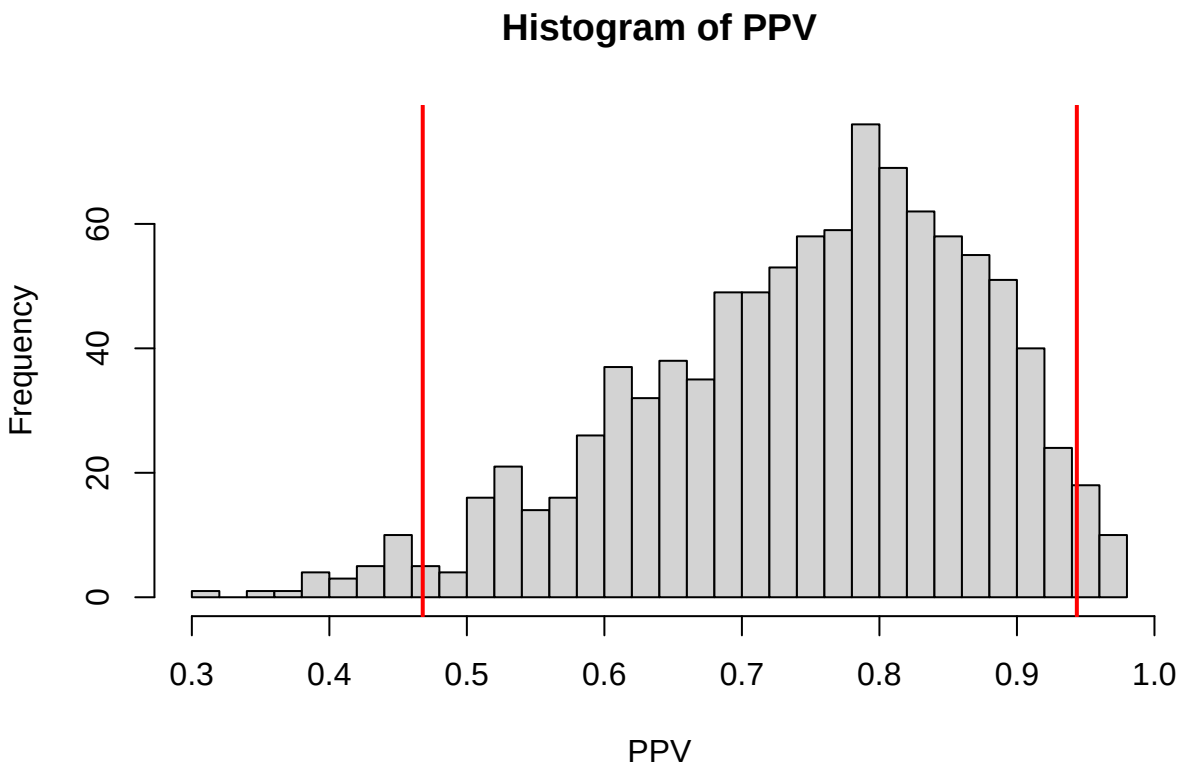
3) Consider a cutoff of 2 on CHD8 expressions.

Estimate the positive predictive value (PPV) of observing CHD8 expressions greater than 2. Provide a 95% posterior interval on the PPV. Briefly interpret what the interval tells us in your own words.

```
# get prev
pi = sum(YY==1)/(sum(YY==0)+sum(YY==1))

PPV = rep(0,1000)

cutoff = 2 # exponentiated the Gamma dist., so expression value is 2

for (i in 1:1000){
  PPV[i] = ((gamma.cdf2(cutoff,i))*pi)/((gamma.cdf2(cutoff,i))*pi+(gamma.cdf1(cutoff,i))*(1-pi))
}

mean(PPV)
```

```
## [1] 0.7488884
```

```
hist(PPV,breaks = 30)
abline(v=quantile(PPV,0.975),col = 'red',lwd = 2)
abline(v=quantile(PPV,0.025),col = 'red',lwd = 2)
```




```
mean_PPV <- mean(PPV)
names(mean_PPV) <- "Mean"

c(quantile(PPV,0.025), mean_PPV, quantile(PPV,0.975))
```

```
##      2.5%      Mean     97.5%
## 0.4678990 0.7488884 0.9435492
```

The interval tells us that, given our current gamma distribution model and sample for CHD8, the probability of optimal debulking given the presence of the biomarker above a threshold - $P(1 \mid CHD8 > 2)$ - lies between 0.468 and 0.944. Given the wide range of the interval representing considerable uncertainly, it seems that a larger sample size or a parallel study of the same biomarker would lead us to a better approximation of whether CHD8 could be a viable biomarker to predict optimal debulking at a cutoff of 2.