STAT 8700 Final Question 3

Brian Detweiler

Thursday, December 15th

3. Consider the following data representing survival times of a random sample of 20 electrical components:

```
components <- c(51, 3, 17, 13, 5, 4, 17, 1, 5, 3, 8, 22, 1, 1, 13, 8, 15, 3, 1, 13)
```

Suppose that you cannot decide which model to use and are considering the possibility that the data could be either Gamma Distributed, or Weibull Distributed, or log-Normal Distributed.

(a) Use DIC to decide which model is preferable.

First, we'll look at the Gamma parameterized by the mean, as shown in [1].

```
fileName <- "Final_3.a.jags"

modelString ="
model{
  for (i in 1:count) {
    y[i] ~ dgamma(sh, ra)
  }

  y.pred ~ dgamma(sh, ra)
  # parameterized by mean (m) and standard deviation (sd)
  sh <- pow(m, 2) / pow(sd, 2)
  ra <- m / pow(sd, 2)
  m ~ dunif(0,100)
  sd ~ dunif(0,100)
}
"

writeLines(modelString, con=fileName)

components.model = jags.model(file=fileName,
                              data=list(y=components,
                                        count=length(components)),
                              n.chains=4)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
```

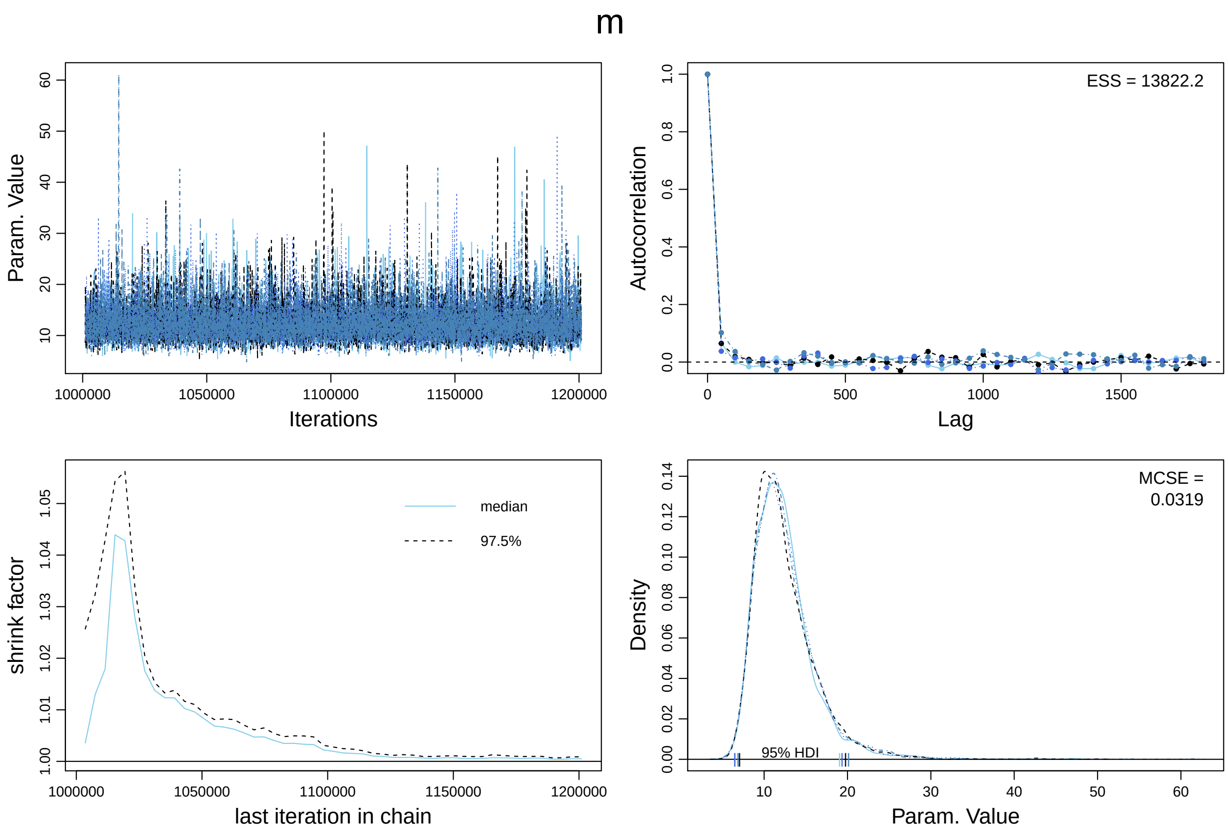
```
##    Observed stochastic nodes: 20
##    Unobserved stochastic nodes: 3
##    Total graph size: 35
##
## Initializing model
```

```r
update(components.model, n.iter=1000000)

components.samples <- coda.samples(model = components.model,
                                   variable.names = c("y.pred", "y", "sh", "ra", "m", "sd"),
                                   n.iter = 200000,
                                   thin = 50)
summary(components.samples)
```

```
##
## Iterations = 1001050:1201000
## Thinning interval = 50
## Number of chains = 4
## Sample size per chain = 4000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##            Mean       SD  Naive SE Time-series SE
## m      12.46807  3.75100 0.0296543      0.0320188
## ra      0.08131  0.03322 0.0002626      0.0002644
## sd     13.48284  5.33361 0.0421659      0.0467556
## sh      0.93161  0.26962 0.0021316      0.0021223
## y[1]   51.00000  0.00000 0.0000000      0.0000000
## y[2]    3.00000  0.00000 0.0000000      0.0000000
## y[3]   17.00000  0.00000 0.0000000      0.0000000
## y[4]   13.00000  0.00000 0.0000000      0.0000000
## y[5]    5.00000  0.00000 0.0000000      0.0000000
## y[6]    4.00000  0.00000 0.0000000      0.0000000
## y[7]   17.00000  0.00000 0.0000000      0.0000000
## y[8]    1.00000  0.00000 0.0000000      0.0000000
## y[9]    5.00000  0.00000 0.0000000      0.0000000
## y[10]   3.00000  0.00000 0.0000000      0.0000000
## y[11]   8.00000  0.00000 0.0000000      0.0000000
## y[12]  22.00000  0.00000 0.0000000      0.0000000
## y[13]   1.00000  0.00000 0.0000000      0.0000000
## y[14]   1.00000  0.00000 0.0000000      0.0000000
## y[15]  13.00000  0.00000 0.0000000      0.0000000
## y[16]   8.00000  0.00000 0.0000000      0.0000000
## y[17]  15.00000  0.00000 0.0000000      0.0000000
## y[18]   3.00000  0.00000 0.0000000      0.0000000
## y[19]   1.00000  0.00000 0.0000000      0.0000000
## y[20]  13.00000  0.00000 0.0000000      0.0000000
## y.pred 12.53792 15.39968 0.1217451      0.1230118
##
## 2. Quantiles for each variable:
##
##            2.5%      25%      50%     75%   97.5%
## m       7.51380  9.95957 11.76468 14.1231 21.6751
```

```
## ra      0.02844  0.05729  0.07734  0.1007  0.1565
## sd      7.36292 10.05547 12.28324 15.4124 26.8055
## sh      0.48915  0.73626  0.90481  1.0944  1.5341
## y[1]   51.00000 51.00000 51.00000 51.0000 51.0000
## y[2]    3.00000  3.00000  3.00000  3.0000  3.0000
## y[3]   17.00000 17.00000 17.00000 17.0000 17.0000
## y[4]   13.00000 13.00000 13.00000 13.0000 13.0000
## y[5]    5.00000  5.00000  5.00000  5.0000  5.0000
## y[6]    4.00000  4.00000  4.00000  4.0000  4.0000
## y[7]   17.00000 17.00000 17.00000 17.0000 17.0000
## y[8]    1.00000  1.00000  1.00000  1.0000  1.0000
## y[9]    5.00000  5.00000  5.00000  5.0000  5.0000
## y[10]   3.00000  3.00000  3.00000  3.0000  3.0000
## y[11]   8.00000  8.00000  8.00000  8.0000  8.0000
## y[12]  22.00000 22.00000 22.00000 22.0000 22.0000
## y[13]   1.00000  1.00000  1.00000  1.0000  1.0000
## y[14]   1.00000  1.00000  1.00000  1.0000  1.0000
## y[15]  13.00000 13.00000 13.00000 13.0000 13.0000
## y[16]   8.00000  8.00000  8.00000  8.0000  8.0000
## y[17]  15.00000 15.00000 15.00000 15.0000 15.0000
## y[18]   3.00000  3.00000  3.00000  3.0000  3.0000
## y[19]   1.00000  1.00000  1.00000  1.0000  1.0000
## y[20]  13.00000 13.00000 13.00000 13.0000 13.0000
## y.pred  0.13797  2.96224  7.77009 16.5744 52.5706
```

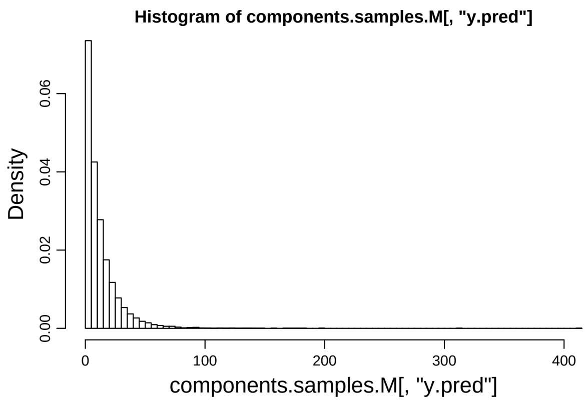
```r
diagMCMC(components.samples)

components.dic <- dic.samples(model = components.model, n.iter = 200000, thin = 50)
```




```r
components.dic
```

```
## Mean deviance:  135.5
## penalty 2.579
## Penalized deviance: 138.1
```

```r
components.samples.M <- as.matrix(components.samples)

hist(components.samples.M[,"y.pred"], breaks=100, freq=FALSE)
```



Now, we'll look at the Gamma parameterized by the mode, as also shown in [1].

```
fileName <- "Final_3.a.1.jags"

modelString ="
model{
  for (i in 1:count) {
    y[i] ~ dgamma(sh, ra)
  }

  y.pred ~ dgamma(sh, ra)

  gmean <- sh / ra

  # parameterized by mode (m) and standard deviation (sd):
  sh <- 1 + m * ra
  ra <- ( m + sqrt( m^2 + 4 * sd^2 ) ) / ( 2 * sd^2 )


  m ~ dunif(0,100)
  sd ~ dunif(0,100)
}
"

writeLines(modelString, con=fileName)

components.model.1 = jags.model(file=fileName,
```

```r
                              data=list(y=components,
                                        count=length(components)),
                              n.chains=4)
```

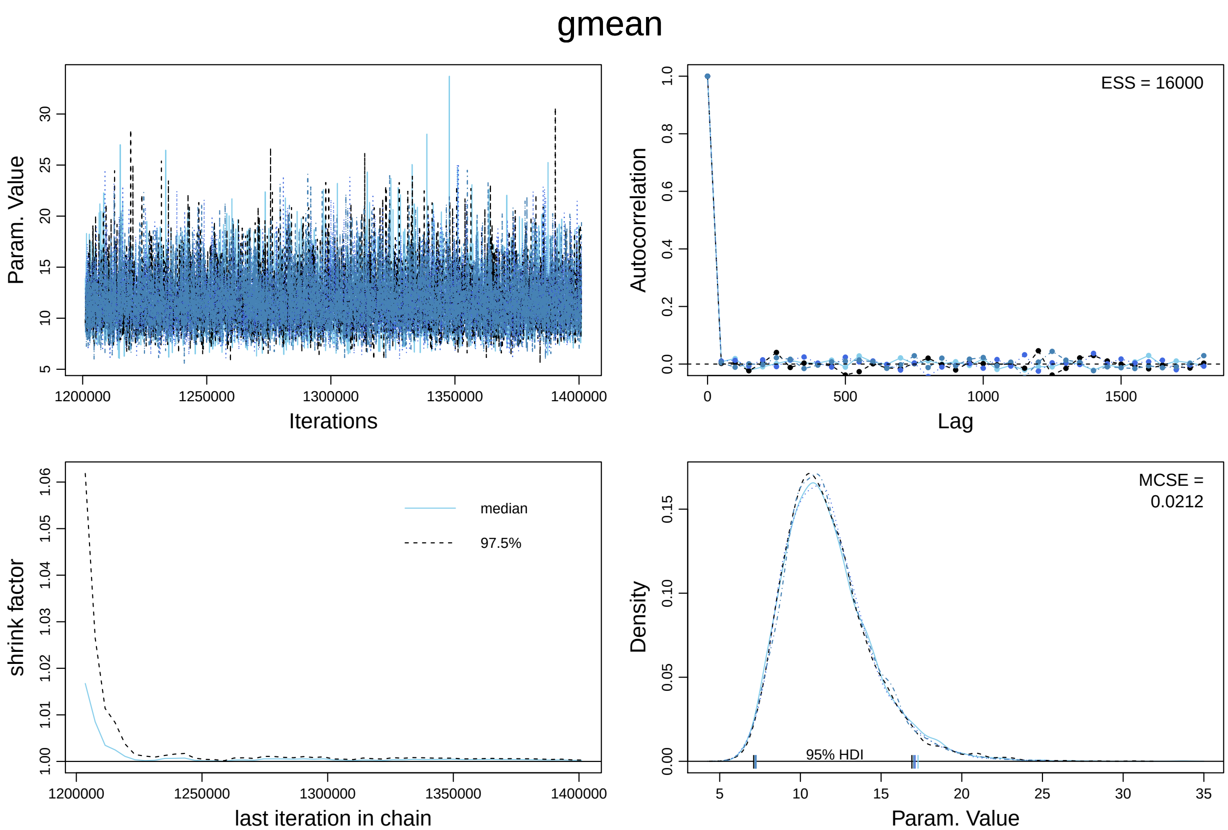
```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 20
##    Unobserved stochastic nodes: 3
##    Total graph size: 45
##
## Initializing model
```

```r
update(components.model.1, n.iter=1200000)

components.samples.1 <- coda.samples(model = components.model.1,
                                     variable.names = c("y.pred", "y", "gmean", "sh", "ra", "m", "sd
                                     n.iter = 200000,
                                     thin = 50)

summary(components.samples.1)
```

```
##
## Iterations = 1201050:1401000
## Thinning interval = 50
## Number of chains = 4
## Sample size per chain = 4000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##           Mean      SD  Naive SE Time-series SE
## gmean  11.6936 2.68529 0.0212291      0.0212307
## m       1.8004 1.32434 0.0104698      0.0105179
## ra      0.1086 0.02925 0.0002312      0.0002313
## sd     10.7316 2.64837 0.0209372      0.0209381
## sh      1.2082 0.18387 0.0014536      0.0014303
## y[1]   51.0000 0.00000 0.0000000      0.0000000
## y[2]    3.0000 0.00000 0.0000000      0.0000000
## y[3]   17.0000 0.00000 0.0000000      0.0000000
## y[4]   13.0000 0.00000 0.0000000      0.0000000
## y[5]    5.0000 0.00000 0.0000000      0.0000000
## y[6]    4.0000 0.00000 0.0000000      0.0000000
## y[7]   17.0000 0.00000 0.0000000      0.0000000
## y[8]    1.0000 0.00000 0.0000000      0.0000000
## y[9]    5.0000 0.00000 0.0000000      0.0000000
## y[10]   3.0000 0.00000 0.0000000      0.0000000
## y[11]   8.0000 0.00000 0.0000000      0.0000000
## y[12]  22.0000 0.00000 0.0000000      0.0000000
## y[13]   1.0000 0.00000 0.0000000      0.0000000
## y[14]   1.0000 0.00000 0.0000000      0.0000000
## y[15]  13.0000 0.00000 0.0000000      0.0000000
```

```
## y[16]   8.0000  0.00000 0.0000000      0.0000000
## y[17]  15.0000  0.00000 0.0000000      0.0000000
## y[18]   3.0000  0.00000 0.0000000      0.0000000
## y[19]   1.0000  0.00000 0.0000000      0.0000000
## y[20]  13.0000  0.00000 0.0000000      0.0000000
## y.pred 11.7216 11.57897 0.0915398      0.0918943
##
## 2. Quantiles for each variable:
##
##            2.5%      25%     50%     75%   97.5%
## gmean   7.60684  9.80952 11.3053 13.1001 18.0711
## m       0.07163  0.73935  1.5495  2.6130  4.8944
## ra      0.06122  0.08801  0.1052  0.1254  0.1758
## sd      6.78605  8.87195 10.3213 12.1094 17.0175
## sh      1.00654  1.07014  1.1600  1.2937  1.6818
## y[1]   51.00000 51.00000 51.0000 51.0000 51.0000
## y[2]    3.00000  3.00000  3.0000  3.0000  3.0000
## y[3]   17.00000 17.00000 17.0000 17.0000 17.0000
## y[4]   13.00000 13.00000 13.0000 13.0000 13.0000
## y[5]    5.00000  5.00000  5.0000  5.0000  5.0000
## y[6]    4.00000  4.00000  4.0000  4.0000  4.0000
## y[7]   17.00000 17.00000 17.0000 17.0000 17.0000
## y[8]    1.00000  1.00000  1.0000  1.0000  1.0000
## y[9]    5.00000  5.00000  5.0000  5.0000  5.0000
## y[10]   3.00000  3.00000  3.0000  3.0000  3.0000
## y[11]   8.00000  8.00000  8.0000  8.0000  8.0000
## y[12]  22.00000 22.00000 22.0000 22.0000 22.0000
## y[13]   1.00000  1.00000  1.0000  1.0000  1.0000
## y[14]   1.00000  1.00000  1.0000  1.0000  1.0000
## y[15]  13.00000 13.00000 13.0000 13.0000 13.0000
## y[16]   8.00000  8.00000  8.0000  8.0000  8.0000
## y[17]  15.00000 15.00000 15.0000 15.0000 15.0000
## y[18]   3.00000  3.00000  3.0000  3.0000  3.0000
## y[19]   1.00000  1.00000  1.0000  1.0000  1.0000
## y[20]  13.00000 13.00000 13.0000 13.0000 13.0000
## y.pred  0.46879  3.78113  8.3725 15.8722 42.3513
```

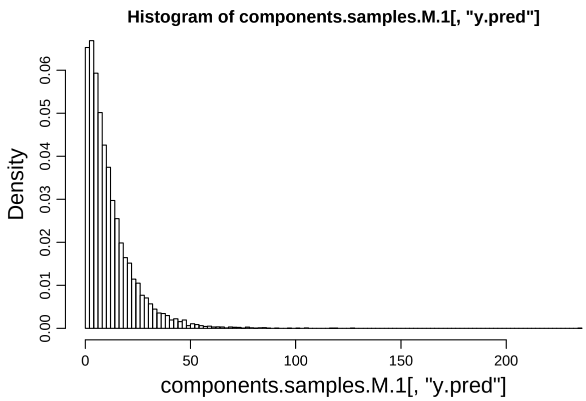
```r
diagMCMC(components.samples.1)

components.dic.1 <- dic.samples(model = components.model.1, n.iter = 200000, thin = 50)
```




```
components.dic.1
```

```
## Mean deviance:  134.8
## penalty 1.403
## Penalized deviance: 136.2
```

```
components.samples.M.1 <- as.matrix(components.samples.1)
hist(components.samples.M.1[,"y.pred"], breaks=100, freq=FALSE)
```



The Gamma parameterized by the mode has a slightly better deviance score.

Let's try the Weibull.

```
fileName <- "Final_3.a.2.jags"
modelString ="
model{
  for (i in 1:count) {
    y[i] ~ dweib(a, b)
  }

  y.pred ~ dweib(a, b)

  a ~ dgamma(0.01, 0.01)
  b ~ dgamma(0.01, 0.01)
}
"

writeLines(modelString, con=fileName)

components.model.2 = jags.model(file=fileName,
                              data=list(y=components,
                                        count=length(components)),
                              n.chains=4)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
```

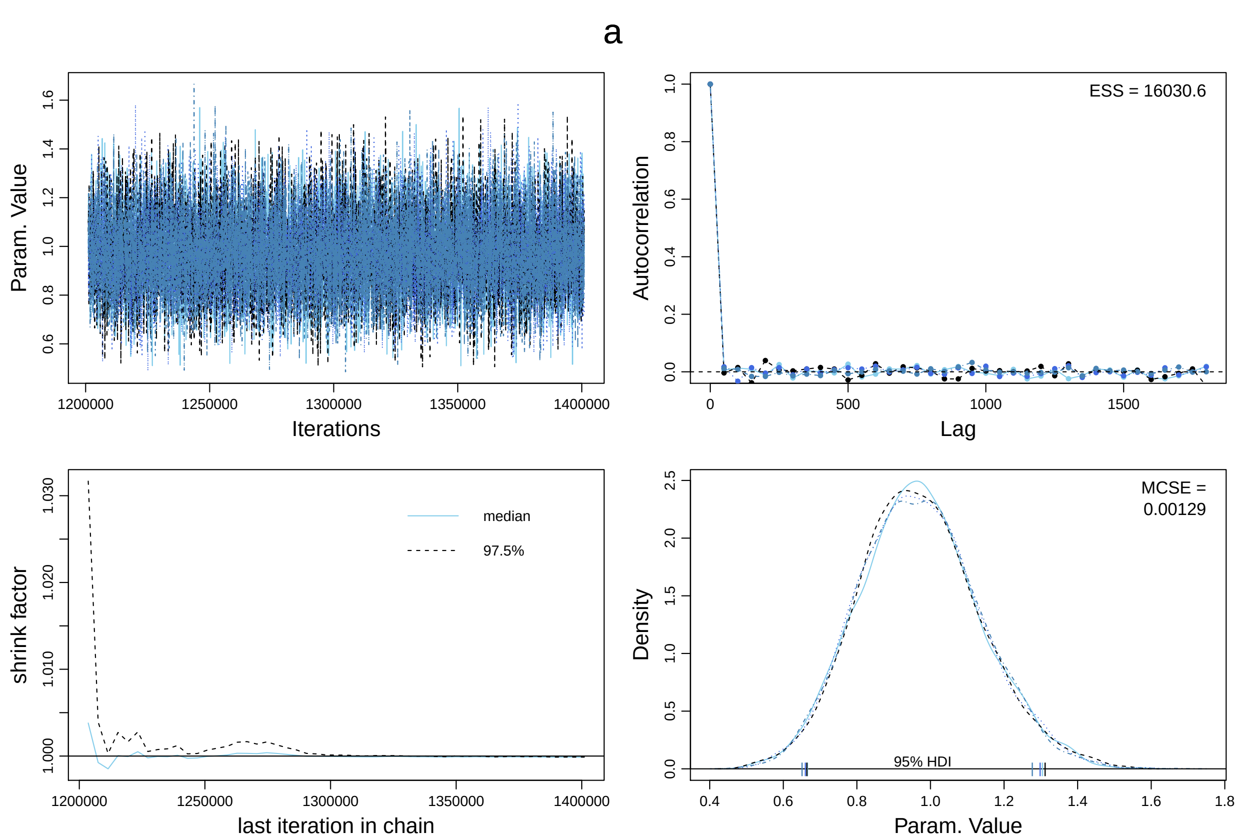
```
## Graph information:
##    Observed stochastic nodes: 20
##    Unobserved stochastic nodes: 3
##    Total graph size: 28
##
## Initializing model
```

```r
update(components.model.2, n.iter=1200000)

components.samples.2 <- coda.samples(model = components.model.2,
                                     variable.names = c("y.pred", "y", "a", "b"),
                                     n.iter = 200000,
                                     thin = 50)

summary(components.samples.2)
```

```
##
## Iterations = 1201050:1401000
## Thinning interval = 50
## Number of chains = 4
## Sample size per chain = 4000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##           Mean       SD  Naive SE Time-series SE
## a       0.9705  0.16378 0.0012948      0.0012940
## b       0.1163  0.05835 0.0004613      0.0004545
## y[1]   51.0000  0.00000 0.0000000      0.0000000
## y[2]    3.0000  0.00000 0.0000000      0.0000000
## y[3]   17.0000  0.00000 0.0000000      0.0000000
## y[4]   13.0000  0.00000 0.0000000      0.0000000
## y[5]    5.0000  0.00000 0.0000000      0.0000000
## y[6]    4.0000  0.00000 0.0000000      0.0000000
## y[7]   17.0000  0.00000 0.0000000      0.0000000
## y[8]    1.0000  0.00000 0.0000000      0.0000000
## y[9]    5.0000  0.00000 0.0000000      0.0000000
## y[10]   3.0000  0.00000 0.0000000      0.0000000
## y[11]   8.0000  0.00000 0.0000000      0.0000000
## y[12]  22.0000  0.00000 0.0000000      0.0000000
## y[13]   1.0000  0.00000 0.0000000      0.0000000
## y[14]   1.0000  0.00000 0.0000000      0.0000000
## y[15]  13.0000  0.00000 0.0000000      0.0000000
## y[16]   8.0000  0.00000 0.0000000      0.0000000
## y[17]  15.0000  0.00000 0.0000000      0.0000000
## y[18]   3.0000  0.00000 0.0000000      0.0000000
## y[19]   1.0000  0.00000 0.0000000      0.0000000
## y[20]  13.0000  0.00000 0.0000000      0.0000000
## y.pred 11.0688 13.01340 0.1028800      0.1048719
##
## 2. Quantiles for each variable:
##
##            2.5%     25%     50%     75%   97.5%
## a       0.66753  0.8565  0.9653  1.0769  1.3079
```

```
## b         0.03548  0.0744  0.1056  0.1459  0.2625
## y[1]     51.00000 51.0000 51.0000 51.0000 51.0000
## y[2]      3.00000  3.0000  3.0000  3.0000  3.0000
## y[3]     17.00000 17.0000 17.0000 17.0000 17.0000
## y[4]     13.00000 13.0000 13.0000 13.0000 13.0000
## y[5]      5.00000  5.0000  5.0000  5.0000  5.0000
## y[6]      4.00000  4.0000  4.0000  4.0000  4.0000
## y[7]     17.00000 17.0000 17.0000 17.0000 17.0000
## y[8]      1.00000  1.0000  1.0000  1.0000  1.0000
## y[9]      5.00000  5.0000  5.0000  5.0000  5.0000
## y[10]     3.00000  3.0000  3.0000  3.0000  3.0000
## y[11]     8.00000  8.0000  8.0000  8.0000  8.0000
## y[12]    22.00000 22.0000 22.0000 22.0000 22.0000
## y[13]     1.00000  1.0000  1.0000  1.0000  1.0000
## y[14]     1.00000  1.0000  1.0000  1.0000  1.0000
## y[15]    13.00000 13.0000 13.0000 13.0000 13.0000
## y[16]     8.00000  8.0000  8.0000  8.0000  8.0000
## y[17]    15.00000 15.0000 15.0000 15.0000 15.0000
## y[18]     3.00000  3.0000  3.0000  3.0000  3.0000
## y[19]     1.00000  1.0000  1.0000  1.0000  1.0000
## y[20]    13.00000 13.0000 13.0000 13.0000 13.0000
## y.pred    0.19389  2.7007  7.0470 14.6935 45.2699
```

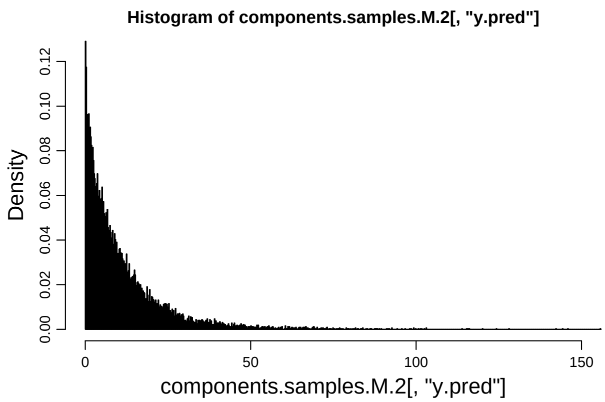
```
diagMCMC(components.samples.2)

components.dic.2 <- dic.samples(model = components.model.2, n.iter = 200000, thin = 50)
```



```r
components.dic.2
```

```
## Mean deviance:  134.9
## penalty 2.195
## Penalized deviance: 137.1
```

```r
components.samples.M.2 <- as.matrix(components.samples.2)
hist(components.samples.M.2[,"y.pred"], breaks=800, freq=FALSE, xlim=c(0, 150))
```



That one did just a little worse. Now we'll try the log-normal.

We'll use a Normal prior for the mean, and a Uniform variance.

We need some hyperpriors for the prior on the mean. We will choose to use a slightly informative prior, since we have some data.

The sample mean is 10.2, but since we don't have a high confidence, we'll put a low precision on it.

$$
\begin{aligned}
Y_i &\sim Log-Normal(\mu, \tau) \\
\mu &\sim Normal(\mu_0, \tau_0) \\
\tau &\sim Uniform(0, 1) \\
\mu_0 &= \overline{y} \\
\tau_0 &= 0.01
\end{aligned}
$$

```r
fileName <- "Final_3.a.3.jags"
modelString ="
model{
  for (i in 1:count) {
    y[i] ~ dlnorm(mu, tau)
  }

  y.pred ~ dlnorm(mu, tau)

  mu ~ dnorm(mu_0, tau_0)
  tau ~ dunif(0, 1)

  mu_0 <- 10.2
  tau_0 <- .01

  sigma2 <- 1 / tau
}
"

writeLines(modelString, con=fileName)

components.model.3 = jags.model(file=fileName,
                              data=list(y=components,
                                        count=length(components)),
                              n.chains=4)
```

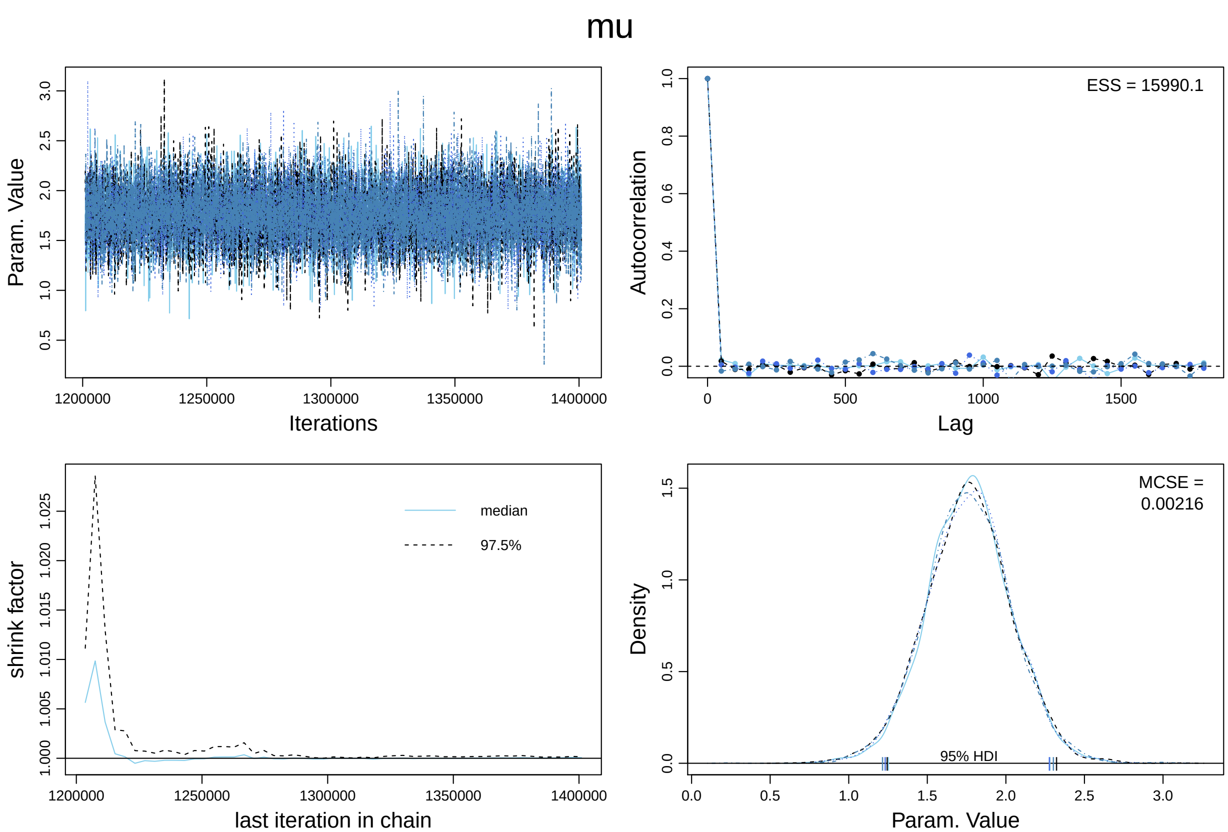
```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 20
##    Unobserved stochastic nodes: 3
##    Total graph size: 31
##
## Initializing model
```

```r
update(components.model.3, n.iter=1200000)

components.samples.3 <- coda.samples(model = components.model.3,
                                     variable.names = c("y.pred", "mu", "tau", "sigma2"),
                                     n.iter = 200000,
                                     thin = 50)

summary(components.samples.3)
```

```
##
## Iterations = 1201050:1401000
## Thinning interval = 50
## Number of chains = 4
## Sample size per chain = 4000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
```

```
##
##             Mean      SD  Naive SE Time-series SE
## mu       1.7659  0.2725 0.002154       0.002155
## sigma2   1.4959  0.4379 0.003462       0.003462
## tau      0.7134  0.1645 0.001300       0.001292
## y.pred  13.8219 41.2711 0.326277       0.329180
##
## 2. Quantiles for each variable:
##
##           2.5%    25%    50%     75%   97.5%
## mu      1.2348 1.5868 1.7660  1.9412  2.2972
## sigma2  1.0178 1.1830 1.3861  1.6818  2.6123
## tau     0.3828 0.5946 0.7214  0.8453  0.9825
## y.pred  0.4859 2.6212 5.9240 13.5833 73.0714
```

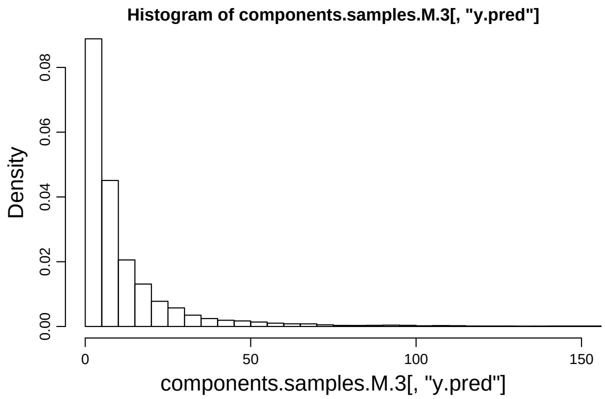
```
diagMCMC(components.samples.3)

components.dic.3 <- dic.samples(model = components.model.3, n.iter = 200000, thin = 50)
```




```
components.dic.3
```

```
## Mean deviance:  133.7
## penalty 1.663
## Penalized deviance: 135.4
```

```r
components.samples.M.3 <- as.matrix(components.samples.3)
hist(components.samples.M.3[,"y.pred"], breaks=800, freq=FALSE, xlim=c(0, 150))
```



This one did slightly better than the others.

Now, we can compare all of the DIC scores and pick the best.

```r
diffdic(components.dic, components.dic.1)
```

```
## Difference: 1.886345
## Sample standard error: 1.656918
```

```r
diffdic(components.dic.1, components.dic.2)
```

```
## Difference: -0.8593319
## Sample standard error: 1.574887
```

```r
diffdic(components.dic.1, components.dic.3)
```

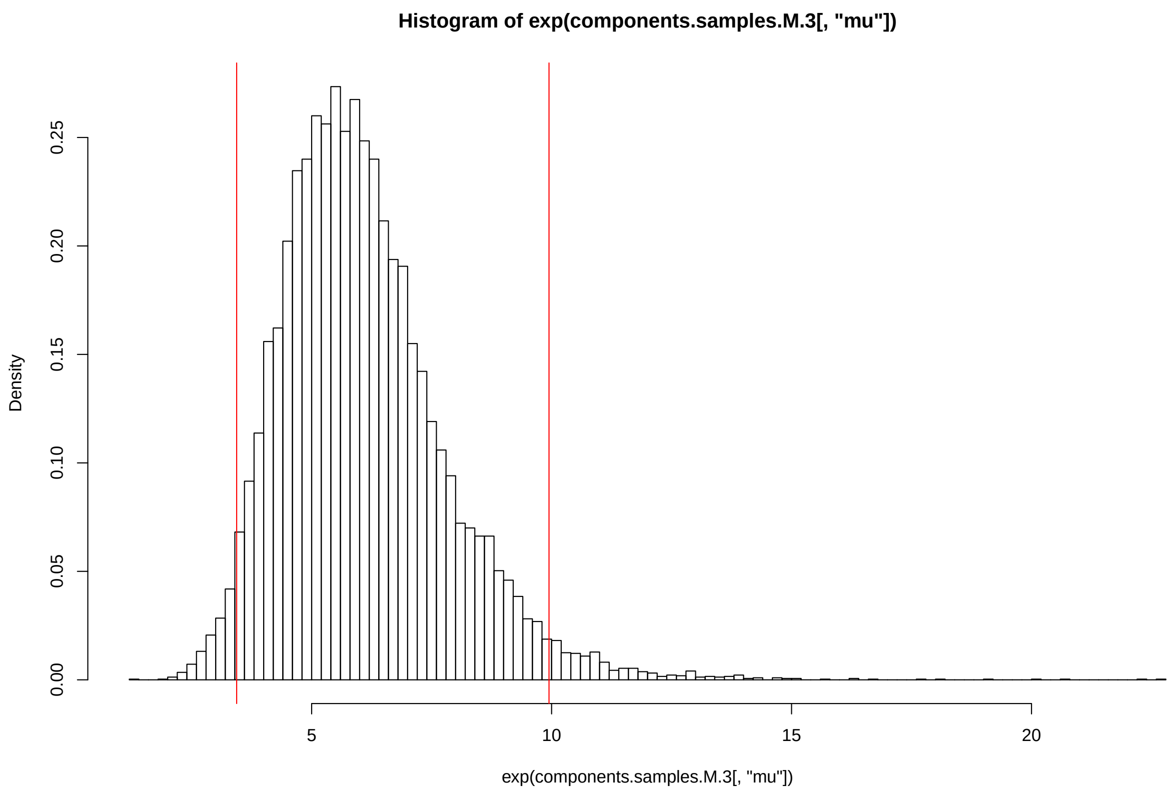
```
## Difference: 0.8374465
## Sample standard error: 3.708619
```

It looks like the thrid model, the log-normal had the best fit, although they were all fairly close.

(b) For the preferred model, calculate the 95% credible interval for the mean survival time.

Since the log-normal is a normal distribution when the data are logged, we need to exponentiate the mean.

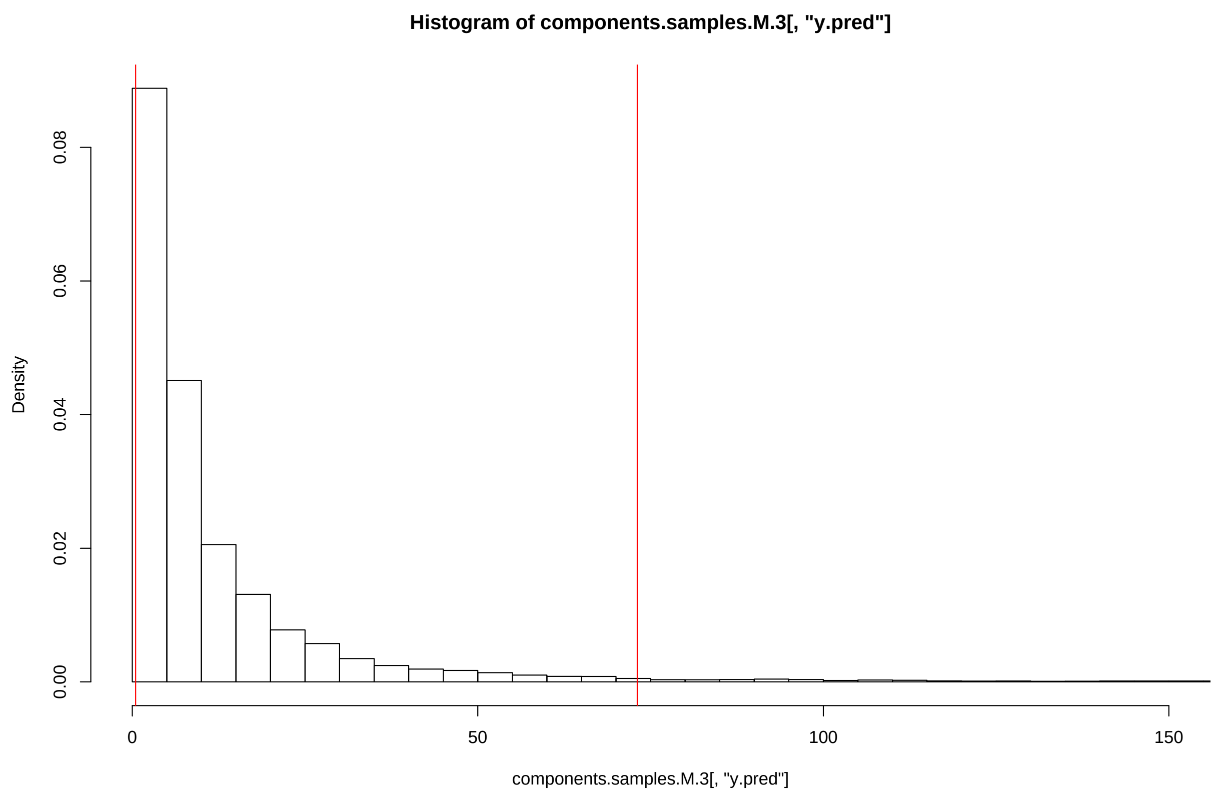
```
hist(exp(components.samples.M.3[,"mu"]), breaks=150, freq=FALSE)
CI <- quantile(exp(components.samples.M.3[,"mu"]), probs = c(0.025, 0.975))
abline(v=CI, col="red")
```



The 95% credible interval for the mean survival time is (3.4376083, 9.9461527).

(c) For the preferred model, calculate the 95% credible interval for the survival time of the next component to be tested.

```
hist(components.samples.M.3[,"y.pred"], breaks=900, freq=FALSE, xlim = c(0, 150))
CI <- quantile(components.samples.M.3[,"y.pred"], probs = c(0.025, 0.975))
abline(v=CI, col="red")
```



The 95% credible interval for the survival time of the next component to be tested is (0.4859456, 73.0714475).

■

References

[1] John K. Kruschke *Gamma Likelihood Parameterized by MODE and SD*
Doing Bayesian Data Analysis, August 9, 2012 http://doingbayesiandataanalysis.blogspot.com/2012/08/gamma-likelihood-parameterized-by-mode.html