Using the 'GammaModel' package to model age-at-death counts using the Gamma distribution

Adrian Timpson

July 27, 2018

This vignette provides a guide to using the R package GammaModel, which was created to perform the analysis in the paper 'Modelling caprine age-at-death profiles using the Gamma distribution', JAS 2018, A. Timpson et al. It assumes the user has some basic familiarity with programming in R.

1 Installation

The GammaModel package can be installed directly from GitHub, using the 'devtools' package on the CRAN. Invoke R then install and load devtools by typing:

```
install.packages('devtools')
library(devtools)
```

The GammaModel package can then be installed and loaded, and a summary of the available help files and data sets included in the package can be browsed:

```
install_github('UCL/GammaModel')
library(GammaModel)
help(GammaModel)
```

2 Age-at-death data

Age-at-death data are typically integer counts of teeth that fall into discrete age classes. However, due to varying archaeological preservation the age of some samples are less precise and therefore are assigned to several possible age classes. An age-at-death dataset of 10 European Neolithic sites can be loaded and inspected by typing:

```
library(GammaModel)
data(Neolithic)
print(Neolithic)
```

```
     ABCD A AB  B BC  C CD  D BCD DEF DEFG EF  G EFG HI GHI EFGHI DEFGHI
FON1    2 5  2  0  2  6  4  1   0   0    0  2  1   6  0   0     2      2
TRA1    0 1  0  1  3 10  1  7   0   0    0 12 10   1  0   0     1      0
MOLD    1 0  0  0  0  6  5  8   0   2    0 12  6   0  3   9     3      0
TES     2 0  0  3  3 15  2  7   1   0    0 18  2   0  4   0     0      0
TRA2    0 0  0  1  1  9 12  6   0   7    0  5  0  18  0   5     6      0
PPI     0 0  0  3  2  7  0  4   0   0    0 29 25   0  0   0     3      0
WIK     0 0  0  3  0  8  1 11   0   1    0 32 18   0  1   0     0      0
```

```
PFE     0 0  0 12  2 26  0  6   0   0    0 11 18   4  0   3     0       0
PCS     0 0  0  0  0 41  0 20   0   0    0 20  6   0 10   0     6       0
LAD     1 0  1  2  1  8  5 33   0  26    2 28 14   1  0   0     0       0
```

Further information about the dataset can be found by typing:

```
help(Neolithic)
```

Notice for example site 'PPI' has two counts in the multi-class BC. Crucially, this is not equivalent to 1 count in B and 1 count in C. Instead, BC=2 is equivalent to (B=1 and C=1) or (B=2 and C=0) or (B=0 and C=2). Therefore a key component in calculating the probability of the data given any proposed model is to first generate all possible arrangements of the age-at-death counts, using the allArrangements() function. Notice counts in classes E and F are automatically combined into a multi-class EF, since there were no counts in the raw data that were assigned to only E or F.

```
library(GammaModel)
data(Neolithic)
counts <- Neolithic['PPI',]
aa <- allArrangements(counts)
print(aa)
```

```
   A B C D EF  G HI
1  0 5 7 4 32 25  0
2  0 4 8 4 32 25  0
3  0 3 9 4 32 25  0
4  0 5 7 4 31 26  0
5  0 4 8 4 31 26  0
6  0 3 9 4 31 26  0
7  0 5 7 4 31 25  1
8  0 4 8 4 31 25  1
9  0 3 9 4 31 25  1
10 0 5 7 4 30 27  0
11 0 4 8 4 30 27  0
12 0 3 9 4 30 27  0
13 0 5 7 4 30 26  1
14 0 4 8 4 30 26  1
15 0 3 9 4 30 26  1
16 0 5 7 4 30 25  2
17 0 4 8 4 30 25  2
18 0 3 9 4 30 25  2
19 0 5 7 4 29 28  0
20 0 4 8 4 29 28  0
21 0 3 9 4 29 28  0
22 0 5 7 4 29 27  1
23 0 4 8 4 29 27  1
24 0 3 9 4 29 27  1
25 0 5 7 4 29 26  2
26 0 4 8 4 29 26  2
27 0 3 9 4 29 26  2
28 0 5 7 4 29 25  3
29 0 4 8 4 29 25  3
30 0 3 9 4 29 25  3
```

The number of combinations can become huge if there are many multi-class assignments, which can generate large memory demands:

```
library(GammaModel)
data(Neolithic)
counts <- Neolithic['TRA2',]
x <- allArrangements(counts)
print(counts)

     ABCD A AB B BC C CD D BCD DEF DEFG EF G EFG HI GHI EFGHI DEFGHI
TRA2    0 0  0 1  1 9 12 6   0   7    0  5 0  18  0   5     6      0

print(nrow(x))

[1] 58032
```

3 Goodness of Fit (GOF)

The GammaModel package also provides several age-class models, each of which is described by the probability of death occuring in age classes A to I:

```
 Class             age
     A   0 to 2 months
     B   2 to 6 months
     C  6 to 12 months
     D    1 to 2 years
     E    2 to 3 years
     F    3 to 4 years
     G    4 to 6 years
     H    6 to 8 years
     I 8 years or more
```

These models can be loaded and viewed as follows:

```
# payne models
data(models.payne)
print(models.payne)

        A    B    C    D    E    F    G    H   I
meat 0.15 0.10 0.05 0.20 0.20 0.05 0.05 0.10 0.1
milk 0.52 0.05 0.03 0.04 0.07 0.05 0.04 0.10 0.1
wool 0.15 0.10 0.05 0.05 0.07 0.06 0.06 0.26 0.2

# redding models
data(models.redding)
print(models.redding)

            AB     C     D     E     F     G     H     I
energy   0.096 0.128 0.300 0.226 0.011 0.078 0.098 0.063
security 0.096 0.259 0.265 0.130 0.011 0.078 0.098 0.063
```

Further information about these model probabilities can be found by typing:

```
help(models.payne)
help(models.redding)
```

A Goodness of Fit test (GOF) evaluates how typical some observed data (or more extreme) is under a particular model. This can be achieved using the GOF() function as follows:

```
data(Neolithic)
data(models.payne)
counts <- Neolithic['FON1',]
model <- models.payne['meat',]
GOF(counts, model)
```

P-values are estimated by sampling 10,000 arrangements of the data, but for improved precision this default N argument value may be increased:

```
GOF(counts, model, N = 20000)
```

4 Maximum Likelihood

Both the Maximum Likelihood Estimates (MLE) and the Maximum Likelihood parameter values can be calculated for either the Gamma model or the age-class model, for some given count data. Both require a search of parameter space, and this is achieved using a Differential Evolution stochastic algorithm for the Gamma model, and a Random Search algorithm for the age-class model.

4.1 Age-class Maximum Likelihood

Notice in the example below, the most likely probability for age-class HI = 0 (to 4 d.p), despite the data having a count in multi-class EFGHI (which could come from HI). This illustrates the interactive influence of data in other age-classes. For example, the large number of counts in classes EF and G (12 and 10 respectively) mean that the single count in EFGHI is far more likely to have come from EF or G than from HI.

```
data(Neolithic)
counts <- Neolithic['TRA1',]
MLpar <- ageClassMLparameters(counts)
MLE <- ageClassLogMLE(counts)
print(counts) # raw data

     ABCD A AB B BC  C CD D BCD DEF DEFG EF  G EFG HI GHI EFGHI DEFGHI
TRA1    0 1  0 1  3 10  1 7   0   0    0 12 10   1  0   0     1      0

print(MLpar) # most likely model parameters (probabilities)

       A      B      C      D     EF      G HI
1 0.0213 0.0482 0.2605 0.1594 0.2767 0.2339  0

print(MLE) # log maximum likelihood estimate

[1] -5.132455
```

4.2 Gamma Maximum Likelihood

Maximum Likelihood Gamma parameters 'shape' and 'mean' can be found as follows:

```
data(Neolithic)
counts <- Neolithic['TRA1',]
MLpar <- gammaMLparameters(counts)
MLE <- gammaLogMLE(counts)
print(MLpar) # most likely model parameters

  shape  mean
1 1.532 2.409

print(MLE) # log maximum likelihood estimate

[1] -11.21172

# find ML parameters for three datasets
p1 <- gammaMLparameters(Neolithic['TRA1',])
p2 <- gammaMLparameters(Neolithic['TES',])
p3 <- gammaMLparameters(Neolithic['WIK',])
```

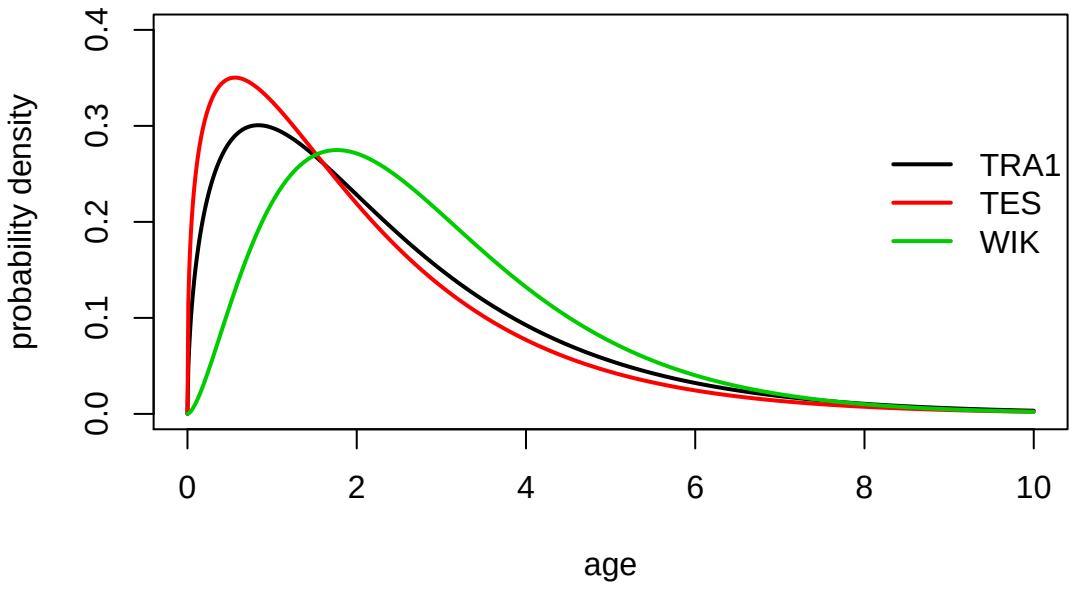
These can then be used to plot the full Gamma distribution using the dgamma() function. Notice this function requires 'shape' and 'rate' parameters, where rate = shape/mean.

```r
# x-axis to cover 10 years
x <- seq(0,10,length.out=1000)

# full Gamma distributions
y1 <- dgamma(x, p1$shape, p1$shape/p1$mean)
y2 <- dgamma(x, p2$shape, p2$shape/p2$mean)
y3 <- dgamma(x, p3$shape, p3$shape/p3$mean)

plot(NULL,xlim=c(0,10),ylim=c(0,0.4),xlab='age', ylab='probability density')
lines(x,y1,col=1,lwd=2)
lines(x,y2,col=2,lwd=2)
lines(x,y3,col=3,lwd=2)

# legend
legend(x=8, y=0.3, bty='n', col=1:3, lwd=2, legend= c('TRA1','TES','WIK'))
```

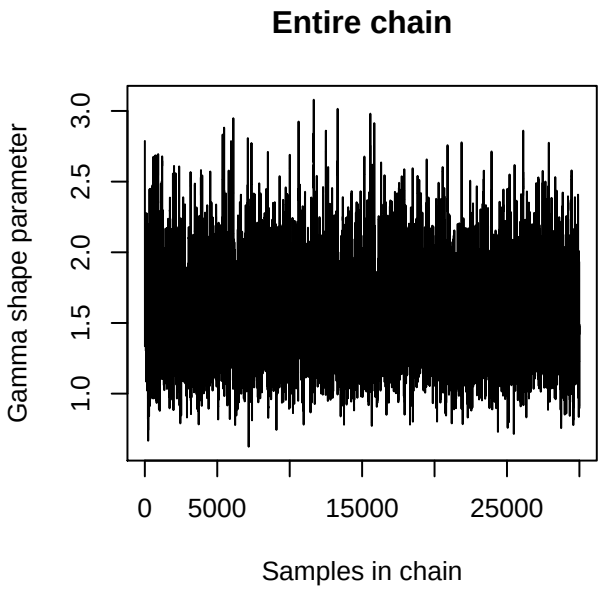


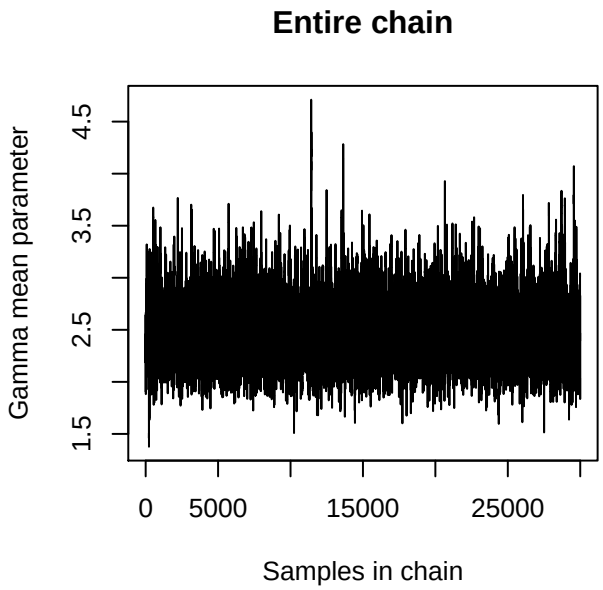
5 Gamma likelihood distribution

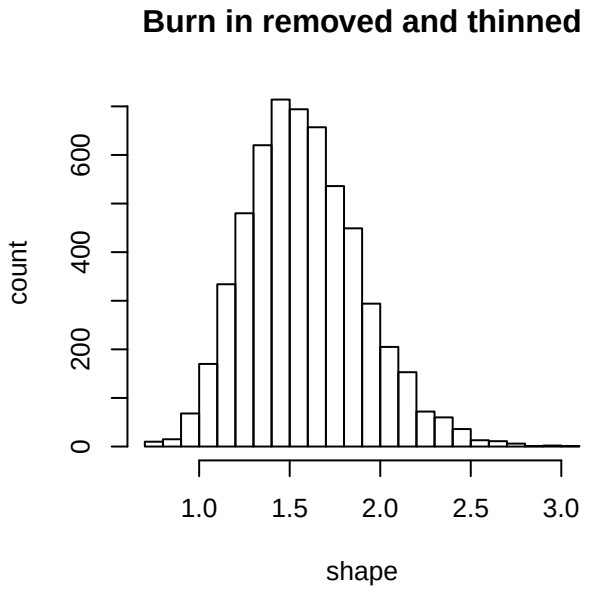
We can find the full joint likelihood distribution of the Gamma parameters 'shape' and 'mean' via Markov Chain Monte Carlo (MCMC), using the function mcmc(), to show the full uncertainty due to small sample sizes and multi-class assignments. By default, mcmc() will output a progress update for each 20 percent completion. At completion the Acceptance Ratio (AR) is printed, and the full MCMC chain is plotted. Together these allow the user to check the chain is well mixed, and a 'hairy catepillar' appearance togther with an acceptance ratio of around 0.4 suggests an efficient search. Otherwise, the total number of iterations can be increased from the default argument N =30,000, and the default prop = 0.4 can be reduced, ensuring smaller average jumps by the proposal function.

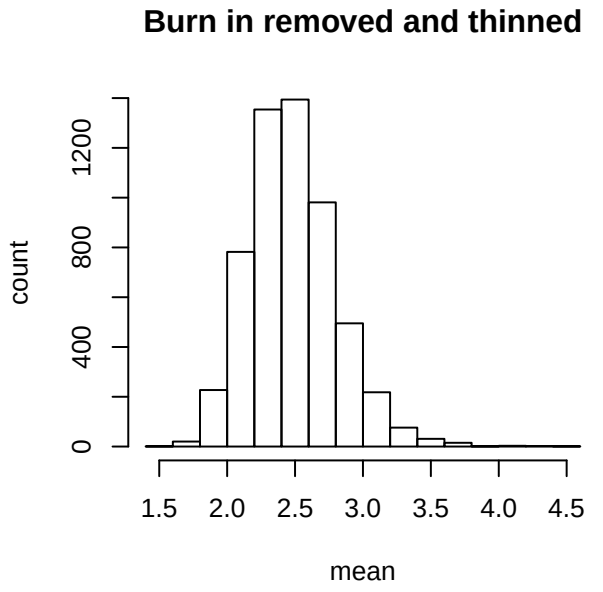
```
data(Neolithic) # load data
counts <- Neolithic['TRA1',]
pars <- mcmc(counts) # generate a MCMC chain

[1] "6000 of 30000 samples completed"
[1] "12000 of 30000 samples completed"
[1] "18000 of 30000 samples completed"
[1] "24000 of 30000 samples completed"
[1] "30000 of 30000 samples completed"
[1] "acceptance ratio =  0.454"
```





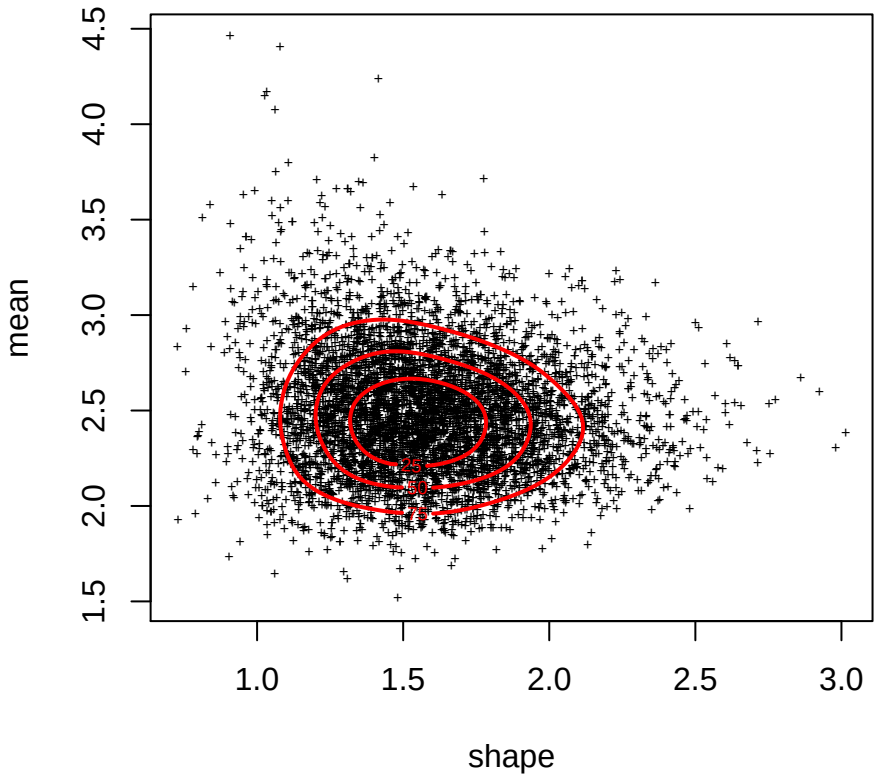




Each row of the returned data frame is a sample from the joint likelihood distribution, after burnin and thinning. This can be graphically represented in various ways.

```
# plot the MCMC chain output
plot(pars, pch='+', cex = 0.5)

# Add contour lines by first generating 2D kernels
# Requires the ks package
library(ks)
H <- matrix(c(0.012, 0, 0, 0.012), 2, 2)
fhat <- kde(x=pars, H=H)
plot(fhat, drawlabels=T, lwd=2, lty=1, add=T, col='red')
```



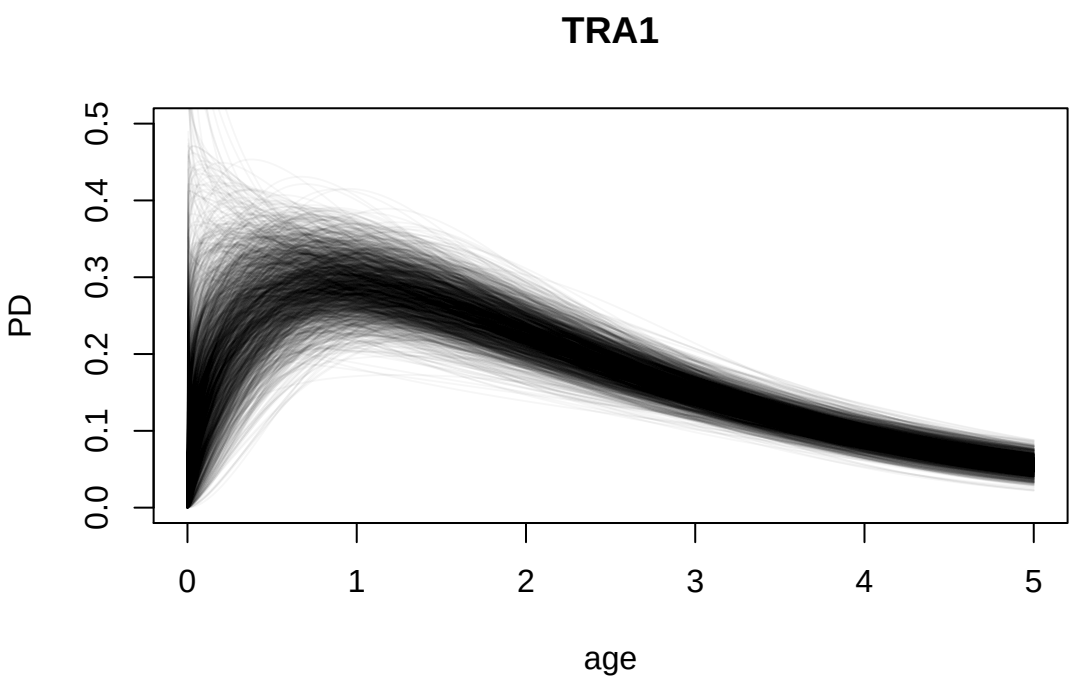
Each sample from the MCMC chain is a pair of Gamma parameters that can be used to construct the full Gamma distribution, thus representing the uncertainty in the distribution directly, rather than via a graphic representation of the underlying parameters.

```r
# calculate the 'rate' parameter for each sample
pars$rate <- pars$shape / pars$mean

# generate an age range of interest
x <- seq(0,5,length.out=1000)

# start with a blank plot
plot(NULL, xlim=c(0,5), ylim=c(0,0.5), xlab='age', ylab='PD', main='TRA1')

# add 2000 samples from the chain
library(scales) # alpha function for transparency
for(n in 1:2000){
        y <- dgamma(x, shape=pars$shape[n], rate=pars$rate[n])
        lines(x,y,col=alpha('black',alpha=0.04))
        }
```



6 Summary statistics and their Confidence Intervals

Useful summary statistics and their confidence intervals (CI) can be derived from the MCMC chain output. The ML of the 'mean' parameter may be directly interpreted as the mean slaughter age of the herd. The MCMC chain of values are normally distributed (a consequence of the central limit theorem) and therefore the CI may be found using quantiles. The mode can be interpreted as the most likely slaughter age, and calculated exactly using the ML parameters mean (mu) and shape (k) as follows: mode = (k - 1)(mu / k). Mathematically the mode is not defined for k smaller than 1 since the probability density function (PDF) = zero when age = 0. However the PDF asymptotically increases as the age approaches zero so for real data this constraint is irrelevant, and such distributions can be described as having a peak kill age of 'new-born'. In the same way, the mode can be

calculated for each sample from the MCMC chain. These are not normally distributed so CI are calculated from the Highest Density Interval (HDI) using the p.interval() function in the LaplacesDemon package. Finally, the age range covering the majority of slaughters is typically skewed, often bounded by zero (e.g. site FON1). This can calculated from the 50 percent HDI, again using the p.interval() function from LaplacesDemon.

```
#------------------------------------------------------------
# mean slaughter age
#------------------------------------------------------------
# Maximum Likelihood
MLpar$mean

[1] 2.409

#95% CI
quantile(pars$mean, c(0.025, 0.975))

    2.5%    97.5%
1.935319 3.185995

#------------------------------------------------------------
# mode slaughter age (peak / most common)
#------------------------------------------------------------
# Maximum Likelihood
mode <- (MLpar$shape - 1) * (MLpar$mean / MLpar$shape)

# constrain to age zero
if(mode < 0) mode <- 0
print(mode)

[1] 0.8365457

# 95% CI
# first calculate the mode for each sample of the chain
pars$mode <- (pars$shape - 1) * (pars$mean / pars$shape)

# constrain to age zero
pars$mode[pars$mode < 0] <- 0

# finally the 95% HDI
library(LaplacesDemon)
p.interval(pars$mode, prob = 0.95)

         Lower    Upper
[1,] 0.1571248 1.500954
attr(,"Probability.Interval")
[1] 0.9500089

#------------------------------------------------------------
# Age range covering the majority of slaughters
#------------------------------------------------------------
# rgamma() requires shape and rate parameters
MLpar$rate <- MLpar$shape / MLpar$mean
```

```r
# generate random samples from the distribution
rg <- rgamma(1000000, shape = MLpar$shape, rate = MLpar$rate)

# find the 50% HDI range
p.interval(rg, prob = 0.5)

         Lower    Upper
[1,] 0.2213853 2.054223
attr(,"Probability.Interval")
[1] 0.5
```

Note, the values output above are the result of one MCMC chain, and with therefore differ slightly if the mcmc() function is re-run. For improved accuracy longer or multiple chains are recommended.