University of Cape Town

STA5003W

Bayesian Decision Modelling

Computational Assignment

Author:
Raisa Salie

Student Number:
SLXRAI001

June 18, 2020

Contents

1 Cost Overrun Problem

We are given cost overrun data for contractors i on similar projects j.

i	N_i	Cost Overruns (X_{ij})						
1	4	2.588	2.364	0.153	0.418			
2	7	0.016	0.243	0.330	0.167	2.507	1.091	1.311
3	3	1.431	0.444	0.053				

(1)

We are also given that the X_{ij} follow a Pareto distribution of the form

$$f\left(x|\alpha_i\right) = \frac{\alpha_i}{(1+x)^{\alpha_i+1}} \quad \text{for } x > 0. \tag{2}$$

Each α_i arises independently from an identical exponential distribution of the form

$$\pi(\alpha|\lambda) = \lambda \mathrm{e}^{-\lambda\alpha} \quad \text{for } \alpha > 0 \tag{3}$$

where we use a prior $\lambda \propto \lambda^{-1}$ for the hyper-parameter, since it is non-negative.

We are required to implement a hierarchical model in `R`. To do so a number of a derivations are required, as well as an understanding of the implementations of sampling methods and algorithms used.

1.1 Part A: Deriving Conditional Posterior for $\boldsymbol{\alpha}$

The likelihood function is given by

$$\mathscr{L}(\boldsymbol{x};\boldsymbol{\alpha};\lambda) = \prod_{i=1}^{3}\prod_{j=1}^{N_i} \frac{\alpha_i}{(1+x_{ij})^{\alpha_i+1}} \tag{4}$$

$$= \prod_{i=1}^{3} \frac{\alpha_i^{N_i}}{\prod_{j=1}^{N_i}(1+x_{ij})^{\alpha_i+1}} \tag{5}$$

$$= \prod_{i=1}^{3} \frac{\alpha_i^{N_i}}{Y_i^{\alpha_i+1}} \tag{6}$$

where $Y_i = \prod_{j=1}^{N_i}(1 + X_{ij}.)$. The posterior density is given by

$$\pi(\boldsymbol{\alpha}|\boldsymbol{x}) \propto \int_{\lambda} \pi(\boldsymbol{\alpha}, \boldsymbol{\lambda}|\boldsymbol{x}) \tag{7}$$

$$\propto \int_{\lambda} \mathscr{L}(\boldsymbol{x}; \boldsymbol{\alpha}; \lambda)\pi(\boldsymbol{\alpha}, \lambda|\boldsymbol{x})\pi(\lambda)d\lambda \tag{8}$$

$$\propto \prod_{i=1}^{3} \frac{\alpha_i^{N_i}}{Y_i^{\alpha_i+1}} \int_0^{\infty} \lambda^2 e^{-\lambda \sum_k \alpha_k} d\lambda \tag{9}$$

$$\propto \prod_{i=1}^{3} \frac{\alpha_i^{N_i}}{Y_i^{\alpha_i+1}} \times \frac{\Gamma(3)}{(\sum_k \alpha_k)^3} \times \int_0^{\infty} \frac{(\sum_k \alpha_k)^3}{\Gamma(3)} \lambda^{3-1} e^{\lambda \sum_k \alpha_k} d\lambda \tag{10}$$

$$\propto \prod_{i=1}^{3} \frac{\alpha_i^{N_i}}{Y_i^{\alpha_i+1}} \times \frac{2}{(\sum_{j=1}^{3} \alpha_j)^3}. \tag{11}$$

1.2 Part B: Deriving Conditional Posterior for α_i

The conditional posterior for a single α_i is given by

$$\pi(\alpha_i|\boldsymbol{\alpha}_k, \boldsymbol{x}) \propto \frac{\alpha_i^{N_i}}{Y_i^{\alpha_i+1}} \times \frac{2}{(\sum_{j=1}^{3} \alpha_j)^3} \tag{12}$$

$$\propto \alpha_i^{N_i} e^{-Z_i \alpha_i} \times \frac{2}{(\alpha_i + \sum_{k \neq i} \alpha_k^0)^3} \tag{13}$$

$$\propto g(\alpha_i) \times h(\alpha_i) \tag{14}$$

where $Z_i = \ln Y_i$.

Hence, $\pi(\alpha_i|\boldsymbol{\alpha}_k, \boldsymbol{x})$ is written as a product of $Gamma(N_i+1, Z_i)$ and $Pa(\sum_{k \neq i} \alpha_k^0, 2)$ distribution functions.

1.3 Part C: Gibbs Sampling Scheme

A Gibbs sampler was used to obtain a sample from the posterior distribution $\pi(\boldsymbol{\alpha}|\boldsymbol{x})$. A total of 4000 samples were required. The code for the computations can be found in the Appendix.

The algorithm proceeds as follows:

1. Randomly intialise the starting values $\boldsymbol{\alpha}^0 = (\alpha_1^0, \alpha_2^0, \alpha_3^0)$.
2. At each iteration $t \in \{1, ..., 4000\}$:
   (a) Sample from $\pi(\alpha_1|\alpha_2^{t-1}, \alpha_3^{t-1})$ to get α_1^t.

(b) Sample from $\pi(\alpha_2|\alpha_1^t, \alpha_3^{t-1})$ to get α_2^t.

(c) Sample from $\pi(\alpha_3|\alpha_1^t, \alpha_2^t)$ to get α_3^t.

(d) Store $\boldsymbol{\alpha}^t$ and repeat from (a).

1.3.1 Accept Reject Algorithm for sampling from $\pi(\alpha_i|\boldsymbol{\alpha_k})$

At each sampling in step (2.) of the Gibbs scheme, an accept-reject algorithm is used to generate a sample of size 1 from the conditional posterior $\pi(\alpha_i|\boldsymbol{\alpha_k})$. The Pareto component of the conditional posterior of α_i is used as the candidate distribution, i.e. the distribution described by h, since it is heavier in the tail.

We have that $g(\boldsymbol{t}, \alpha_i) \propto \mathscr{L}(\boldsymbol{x}; \alpha_i; \lambda)$, since the likelihood cannot be factorised further. Then we have

$$\pi(\alpha_i)g(\boldsymbol{t}, \alpha_i) \propto \frac{1}{\lambda} \times \lambda e^{-\lambda\alpha_i} \times \alpha_i^{N_i} e^{-z_i(\alpha_i+1)} \tag{15}$$

$$\propto \alpha_i^{N_i} e^{-\alpha_i(z_i+\lambda)} \tag{16}$$

which is the density function for a $Gamma(\alpha = N_i + 1, \beta = z_i + \lambda)$ distribution.

We are required to find a suitable constant $\mathscr{C}$ defined by

$$\mathscr{C} = \max_{\alpha\in\Theta|h(\alpha)>0} \frac{\pi(\alpha)g(\mathbf{t}, \alpha)}{h(\alpha)}. \tag{17}$$

Such a value $\mathscr{C}$ is obtained in `R` by finding the value α^* for which $\log(\frac{\pi(\alpha)g(\mathbf{t},\alpha)}{h(\alpha)}) = \log(\pi(\alpha)g(\mathbf{t}, \alpha)) - \log(h(\alpha))$ is maximised using the `nlm` function. Thereafter $\mathscr{C}$ is calculated using α^*. This step is done at each sampling of $\pi(\alpha_i|\boldsymbol{\alpha}_{k\neq i})$ at each iteration t. That is, for each of the 4000 run we are required to obtain 3 such optimizations (one for each sampling step).

Thereafter we define the function

$$\gamma(\theta) = \frac{\pi(\theta)g(\mathbf{t}, \theta)}{\mathscr{C}h(\theta)}. \tag{18}$$

The accept-reject algorithm proceeds as follows:

1. Generate a value, say x, from the distribution described by $h(\theta) = \frac{2}{\theta+\sum_{k\neq i}\alpha_k^0}$ (if we are sampling from $\pi(\alpha_i|\boldsymbol{\alpha_k})$ in step (2.) of the Gibbs sampler). Calculate $\gamma(x)$.
2. Draw a value, say u, from the distribution $U(0, 1)$.
3. Accept x if $u \leq \gamma(x)$ and terminate the algorithm, else reject and return to step (1.).

1.3.2 Probability Integral Transform for Sampling from $h(\theta)$

At step (1.) of the accept-reject algorithm we are required to sample from the Pareto distribution described by h. This is done using the probability integral transform method. The cumulative distribution function (cdf) of the candidate distribution h is required, and can be obtained by first principles as follows.

$$p = F_h(x) \tag{19}$$

$$= \int_0^x h(\theta)d\theta \tag{20}$$

$$= 1 - \left(\sum_{k\neq i} \alpha_k^0\right)^2 \left(x + \sum_{k\neq i} \alpha_k^0\right) \tag{21}$$

$$\implies F_h^{-1}(p) = \frac{\sum_{k\neq i} \alpha_k^0}{\sqrt{1-p}} - \sum_{k\neq i} \alpha_k^0 = x \tag{22}$$

To sample from the distribution described by h, we draw a value p from $U \sim (0, 1)$ and calculate $x = F_h^{-1}(p)$. Alternatively, the `actuar` package (Dutang et al. 2008) allows one to easily sample from a Pareto distribution. Within the package the density of the distribution is defined as

$$f(x) = \frac{\alpha\theta^\alpha}{(x+\theta)^{\alpha+1}}, x > 0 \tag{23}$$

where $\alpha = 2$ is the shape parameter and $\theta = \sum_{k\neq i} \alpha_k^0$ is the scale parameter in the case of h. The function `rpareto` can be used to sample from h. Both of these sampling methods were explored with similar results.

1.3.3 MLE Estimation

The MLE estimates of $\boldsymbol{\alpha}$ can be calculated by differentiating the log-likelihood function, which yields $\hat{\alpha}_i = \frac{N_i}{Z_i}$. Alternatively, this can be done in `R` by using `optim` to minimise the negative log-likelihood. Both approaches achieve the following estimates.

$$\hat{\boldsymbol{\alpha}}_{MLE} = (1.34, 2.00, 2.29) \tag{24}$$

1.4 Gibbs Results

The aforementioned computations were executed in `R`, the code for which can be found in the Appendix. The following posterior estimates were obtained.

$$\hat{\boldsymbol{\alpha}}_{Gibbs} = (1.41, 1.94, 2.10) \tag{25}$$

$$\hat{\mathbf{SE}}_{Gibbs} = (0.41, 0.49, 1.16). \tag{26}$$

The scale parameter α determines the "flatness" of the Pareto distribution. Hence, higher α estimates correspond to contractors having cost overruns which are more peaked, indicating that their initial costing is less reliable than those with lower α estimates.

The MLE and Gibbs sampler estimates do not differ greatly. The ordering of contractors is consistent. The introduction of prior information regarding the distribution of the α_i has brought the estimates closer together. This is perhaps because they arise from the same distribution according to the prior information. Hence, the prior information suggests that the contractors' cost overrun behaviour is more homogeneous than is reflected in the data.

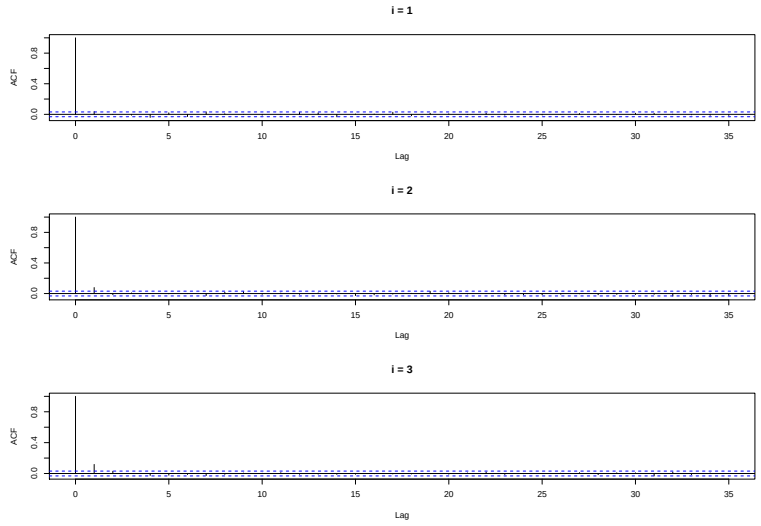

Figure 1: ACF plots for α_i

The histograms in Figure 3 show the posterior distributions obtained from the Gibbs sampling.

1.4.1 Graphical Analysis of Series

The traceplots in Figure 2 appear to have no patterns or trends, and the chain appears to oscillate about a stable mean almost instantaneously. This is likely a result of the starting values, which were chosen as the MLE estimates. However, the chains eventually exhibit the same properties after some burn-in with more radical starting values. It was decided to use 100 as the number of burn-in iterations, since the chain converges quickly. However, this would likely be higher had the chain not been initialised at the MLE estimates. The histograms in Figure 3 show the distributions arising from these series post burn-in.

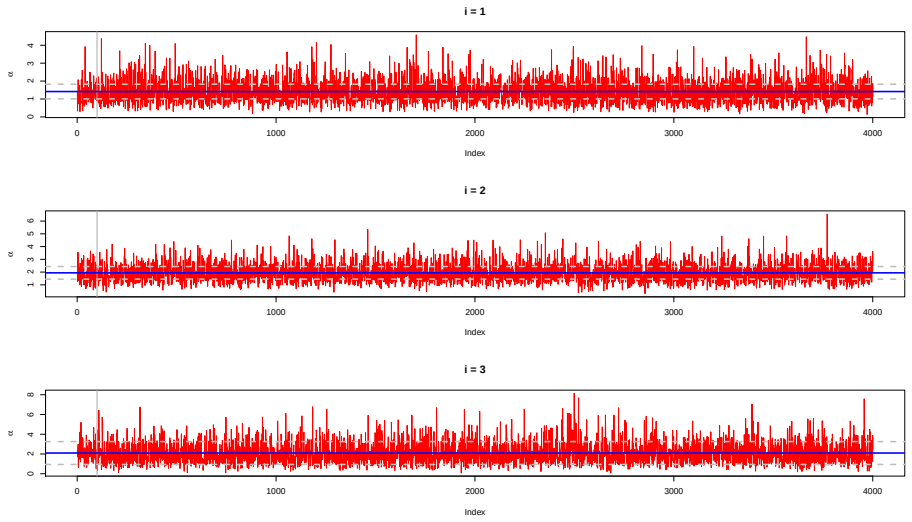

Figure 2: Traceplots for each λ_i

The autocorrelation coefficient plots in Figure 1 show that the within series correlation for α_1 drops to below significance at lag 1. This indicates that we are successfully drawing independent values from the target distribution, and confirms the convergence of the series. However, for α_2 and α_3, there is slight autocorrelation at lag 1. This implies that to ensure the desired independence, we could include every second iteration in the sample. However, given that both correlations lie below 0.2, this may not be necessary.

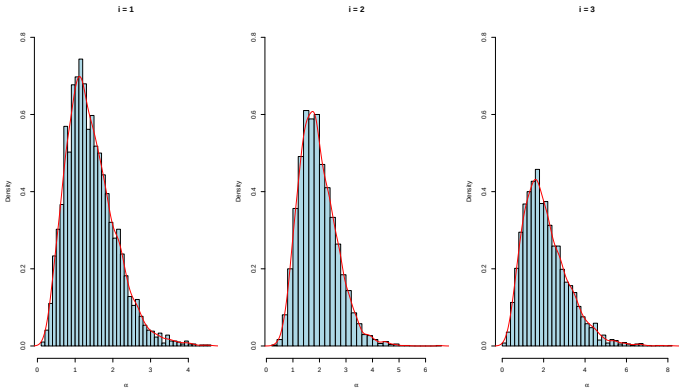

Figure 3: Posterior distribution for each α_i

2 Breakdowns of Production Facility Problem

We are given the number of breakdowns which occurred at breakdown facility i over n_i weeks.

n_i :	3	7	4	8	5	9
X_i :	10	33	3	39	5	50

(27)

Let us assume that the breakdowns at different facilities occur independently. The same hierarchical structure is applied as for the previous problem. However, in this instance the `rstan` package will be used for the computational aspects.

Suppose each X_i arises from a Poisson distribution with rate parameter λ_i, which each independently arise from a common log-normal distribution with parameters μ and σ.

2.1 Part A: Implementing `rstan` to Execute Hierarchical Method

The following distributions were provided, and used in the implementation of `rstan`.

$$X_i \sim (\lambda_i n_i) \tag{28}$$
$$\lambda_i \sim log - normal(\mu, \sigma) \tag{29}$$
$$\mu \sim diffused \tag{30}$$
$$\sigma \sim U(0.1, 0.5) \tag{31}$$

Expectations and credibility intervals for each λ_i were obtained using the `rstan` package in `R`. The hierarchical approach explained in Section 1 was implemented. The Gibbs sampler was run across 5 independent chains with 4000 iterations each, 2000 of which were considered burn-in. The results are summarised in Table 1.

Table 1: Results obtained using diffused prior for μ and uniform prior for σ

	$\hat{Mean}_{data}$	$Mean$	SE_{Mean}	SD	2.5%	25%	50%	75%	97.5%
λ_1	3.33	3.27	0.02	0.87	1.82	2.66	3.19	3.78	5.25
λ_2	4.71	4.38	0.02	0.73	3.08	3.87	4.33	4.85	5.95
λ_3	0.75	1.68	0.02	0.53	0.84	1.30	1.61	1.99	2.88
λ_4	4.88	4.61	0.02	0.75	3.31	4.08	4.55	5.10	6.18
λ_5	1.00	1.70	0.01	0.50	0.88	1.34	1.64	2.00	2.81
λ_6	5.56	5.22	0.02	0.73	3.91	4.72	5.19	5.70	6.71

Table 1 shows some evidence of shrinkage towards the mean. Extreme observations in the sample, for example facilities 6 and 3, are drawn closer to the central mean in the Bayesian estimates. This is the effect of the prior distributions.

It should be noted that the parameters we have obtained are unstable, since the priors used to obtain them are uninformative. Furthermore, we have very little data on each production facility (observations over n_i weeks), and there is a risk of overfitting to this data. Should the procedure be repeated with a different sample, it is likely that the estimates would be different. Thus, there is likely insufficient shrinkage toward the true mean. This problem can be remedied by eliciting more information from decision-makers regarding the prior distributions of λ_i. Should this information be incorporated into the model, the degree of shrinkage may improve.

2.1.1 Convergence

The success of the algorithm is determined by whether convergence was reached in the series. The traceplots in Figure 4 suggest that convergence was reached in each λ_i series. The number of burn-in samples was prudently selected in order to ensure convergence post burn-in. At first glance it appears that most of the chains converge at significantly earlier iterations, however there are some irregular fluctuations which occur closer to the burn-in cut-off (see λ_3 in Figure 4).

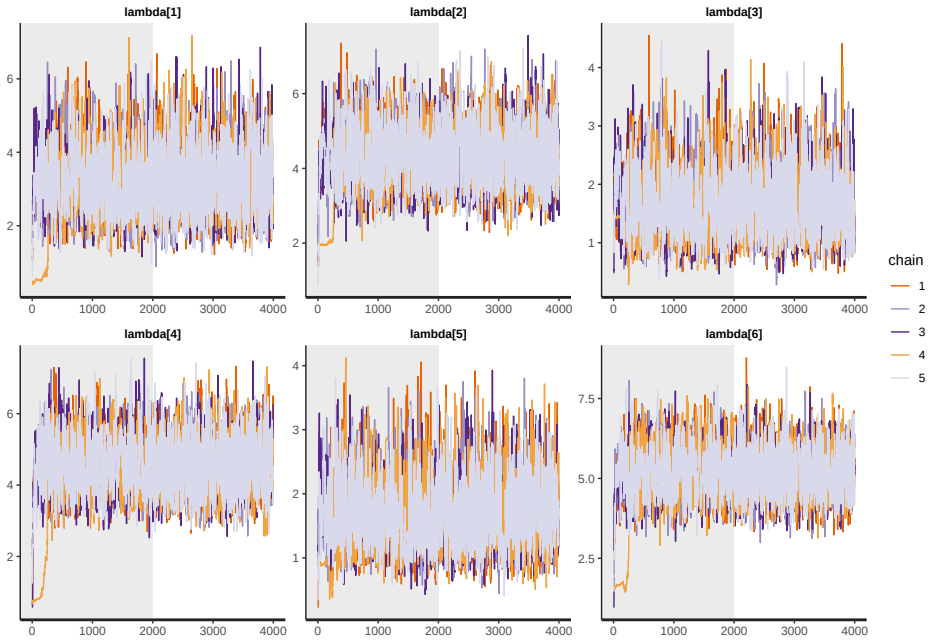

Figure 4: Traceplots for sampling λ_i for $i \in \{1, ..., 6\}$

In Contrast, the traceplot for σ (Figure 5) seems less random, and indicates that perhaps the choice of prior distribution is not suitable. In particular, the question arises whether the ceiling of the prior distribution is too low.

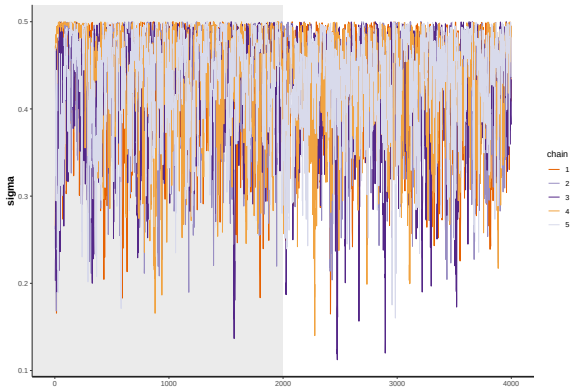

Figure 5: Traceplot for sampling σ

2.2 Part B: Sensitivity Analysis

The initial run produced the distributions of the μ and σ represented in Figure 6. The distribution of σ does not match the uniform prior.

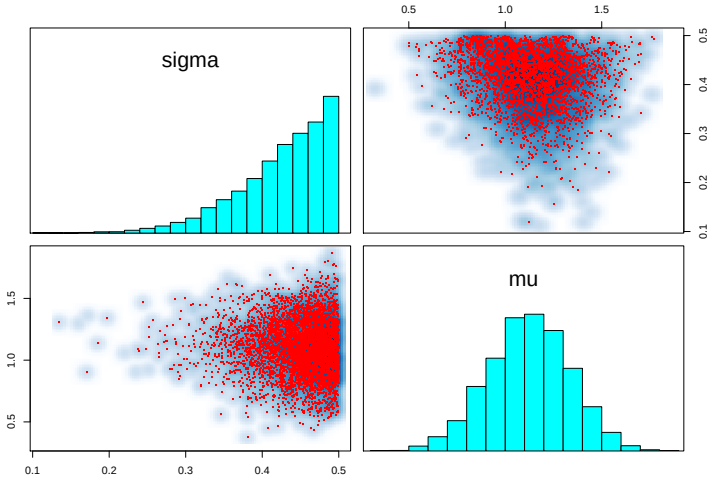

Figure 6: Distribution of μ and σ series

The distribution of σ is skewed, as is evident from its traceplot. The distribution of μ appears normal.

2.2.1 Normal Prior for μ and Extended Uniform Prior for σ

Given the distribution of μ in Figure 6, it was hypothesized that a more suitable prior is a normal distribution with mean and variance 1.

An extended uniform prior distribution, $U(0.1, 1)$ was used to address the concerns of the skewness of the initial prior distribution. The results obtained are summarised in Table 2. The parameter estimates obtained are not dissimilar from those in Table 1. Although there are some changes in λ_3 and λ_5. Therefore, the results have some sensitivity to prior distributions.

Table 2: Results obtained using a $N(1, 1)$ prior for μ, $U(0.1, 1)$ prior for σ

	Mean	SE_{Mean}	SD	2.5%	25%	50%	75%	97.5%
λ_1	3.24	0.01	0.95	1.66	2.56	3.14	3.82	5.34
λ_2	4.58	0.01	0.79	3.15	4.02	4.54	5.10	6.27
λ_3	1.25	0.01	0.51	0.44	0.88	1.18	1.55	2.45
λ_4	4.73	0.01	0.76	3.38	4.20	4.69	5.21	6.36
λ_5	1.36	0.01	0.49	0.59	1.01	1.30	1.65	2.48
λ_6	5.39	0.01	0.75	3.99	4.87	5.37	5.88	6.93

2.2.2 Conjugate Priors

Another idea which was explored was the use of conjugate priors for μ and σ. Given that they arise from a log-normal distribution, normal and Gamma distributions respectively are conjugate priors.

A $N(1, 1)$ prior for μ was again used. The choice of parameters for the Gamma distribution were selected such that the majority of the peak of the distribution lied between 0.1 and 0.5. The parameters `shape = 2` and `rate = 8`, produced a probability between these two boundaries of about 70%, so these parameters were used. Note that the `rate` argument in the `dgamma()` function corresponds to $\frac{1}{\beta}$.

Table 3: Results obtained using a $N(1,1)$ prior for μ, $Gamma(2, \frac{1}{8})$ for σ

	$Mean$	SE_{mean}	SD	2.5%	25%	50%	75%	97.5%
λ_1	3.28	0.01	1.00	1.64	2.56	3.17	3.87	5.52
λ_2	4.62	0.01	0.80	3.19	4.06	4.58	5.12	6.29
λ_3	1.02	0.01	0.50	0.28	0.65	0.95	1.31	2.21
λ_4	4.80	0.01	0.78	3.40	4.26	4.76	5.28	6.45
λ_5	1.19	0.01	0.49	0.44	0.84	1.12	1.47	2.31
λ_6	5.47	0.01	0.78	4.04	4.93	5.44	5.99	7.09

There are some differences between the estimates obtained in Table 1 and 3. Although the deviations in estimates are not large, there is some indication of sensitivity to change of priors.

2.3 Part C: Predictive Distribution

The samples from the predictive distribution can be obtained using `rstan` within the `generated quantities` block. Since it specified that the distribution pertains to the number of breakdowns over 6 weeks, we are required to sample from $Poisson(6\hat{\lambda})$ where $\hat{\lambda} \sim log - normal(\hat{\mu}, \hat{\sigma})$.

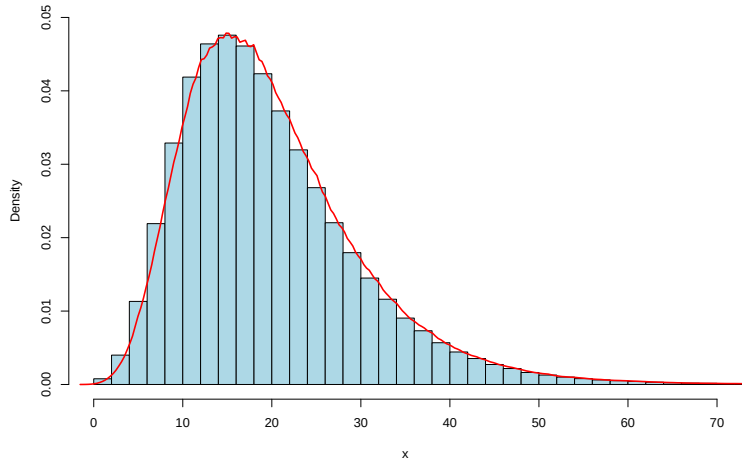

Figure 7: Predictive distribution for out of sample facility for 6 weeks

3 Equipment Life-time Problem

We are required to model equipment lifetime using the Weibull distribution. This is defined in `stan` as

$$f(x|\sigma,\nu) = \frac{\nu x^{\nu-1}}{\sigma^{\nu}} \exp\left[-\left(\frac{x}{\sigma}\right)^{\nu}\right] \quad \text{for } x > 0. \tag{32}$$

Note that ν is denoted the failure rate, where $\nu > 1$ implies increasing failure rate, i.e. that equipment is more prone to failure as it ages.

A sample of 20 observations were recorded, and denoted X_i arising from the Weibull distribution of interest.

0.32	0.63	0.73	0.38	0.54	0.95	0.60	1.03	0.66	0.41
0.48	0.27	0.60	0.21	0.39	0.28	1.03	0.68	0.10	0.69

3.1 Deriving Prior Distributions from Decision-Maker Statements

The form of the prior distributions need to be established for the use of `rstan`. These can be elicited using the decision-maker's statements outlined below.

- The failure rate on engineering grounds is certainly increasing, so that $\nu > 1$ for sure. The decision maker states that she is 95 % sure that $\nu \leq 3$. The Pareto distribution will be used to model this information.
- The decision maker further states that she is sure the median of X lies between 0.2 and 1.5. This can be used to formulate a probability statement.

3.1.1 Prior ubling Distribution of ν

In `rstan` the Pareto distribution function is defined as follows (Team et al. 2018).

$$f(y|y_{min},\alpha) = \frac{\alpha y_{min}^{\alpha}}{y^{\alpha+1}} \tag{33}$$

This distribution function will constitute the skeleton of the prior for ν. The decision maker has expressed that $\nu > 1$, indicating that ν_{min} will be 1. Further, the decision maker's statement that she is 95% sure that $\nu \leq 3$ implies that the cumulative probability $F_{\nu}(3) = 0.95$. This information can be used to establish a reasonable

value for α_ν with the use of the cumulative distribution function.

$$F_\nu(\nu) = 1 - \left(\frac{1}{\nu}\right)^{\alpha_\nu} \tag{34}$$

$$\implies F_\nu(3) = 0.95 = 1 - \left(\frac{1}{3}\right)^{\alpha_\nu} \tag{35}$$

$$\implies \alpha_\nu = -\frac{\log(0.05)}{\log(3)} = 2.7268 \tag{36}$$

$$\implies \pi(\nu) \sim Pareto(\nu_{min} = 1, \alpha_\nu = 2.73) \tag{37}$$

3.1.2 Prior Distribution of σ

The median of the Weibull distribution of the form

$$f(x|\sigma,\nu) = \frac{\nu x^{\nu-1}}{\sigma^\nu} \exp\left[-\left(\frac{x}{\sigma}\right)^\nu\right] \quad \text{for } x > 0 \tag{38}$$

can be expressed as $\sigma(\log(2))^{\frac{1}{\nu}}$. From the decision makers statement, we can form the following probability statement.

$$\Pr[0.2 < \sigma(\log(2))^{\frac{1}{\nu}} < 1.5] = 1 \tag{39}$$

$$\implies \Pr\left[\frac{0.2}{(\log(2))^{\frac{1}{\nu}}} < \sigma < \frac{1.5}{(\log(2))^{\frac{1}{\nu}}}\right] = 1 \tag{40}$$

$$\implies \pi(\sigma|\nu) \sim U\left(\frac{2}{(\log(2))^{\frac{1}{\nu}}}, \frac{1.5}{(\log(2))^{\frac{1}{\nu}}}\right) \tag{41}$$

$$\tag{42}$$

3.2 Results

The derived posteriors were used in `rstan` to implement a Gibbs sampling scheme (see Appendix). The algorithm was run across 6 independent chains with 4000 iterations each, 2000 of which were considered burn-in. As with the productions facility breakdowns problem, this choice was made prudently.

3.2.1 Posterior Expectations and Credibility Intervals

The required results are summarise in Table 4. Estimated for the mean and credibility intervals were obtained for the mean of X, μ, and $\Pr[X \leq 0.5]$. This implies that the expected average equipment lifetime is 0.56 units of time. The probability that equipment lifetime is less than the threshold of 0.5 units of time is 0.47. Note that the estimate for ν implies that the number of failures are doubling per unit time. Failure rate is rapidly increasing as equipment ages.

Table 4: Posterior expectation and credibility intervals

	Mean	SE_{Mean}	SD	2.5%	25%	50%	75%	97.5%
μ	0.56	$9.7e^{-4}$	0.07	0.44	0.51	0.55	0.60	0.71
$\Pr[X \leq 0.5]$	0.47	$6.0e^{-1}$	0.09	0.30	0.42	0.47	0.53	0.65
ν	2.02	0.01	0.39	1.32	1.75	1.99	2.27	2.84
σ	0.63	0.001	0.08	0.49	0.58	0.62	0.67	0.79

3.2.2 Graphical Representations

The traceplots in Figure 8 show that the two series converged. The resulting distributions shown in Figure 10 appear symmetric and roughly normal.

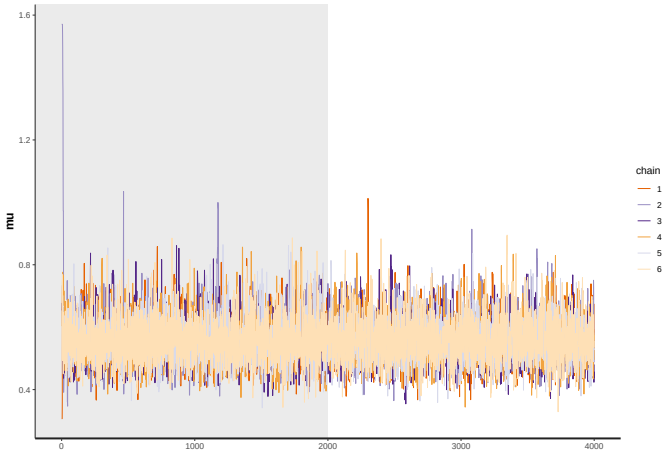

Figure 8: Traceplot for μ series

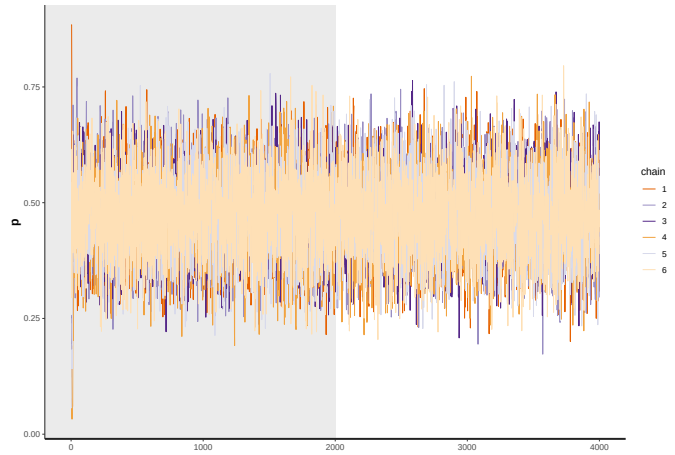

Figure 9: Traceplot for $\Pr[X \leq 0.5]$ series

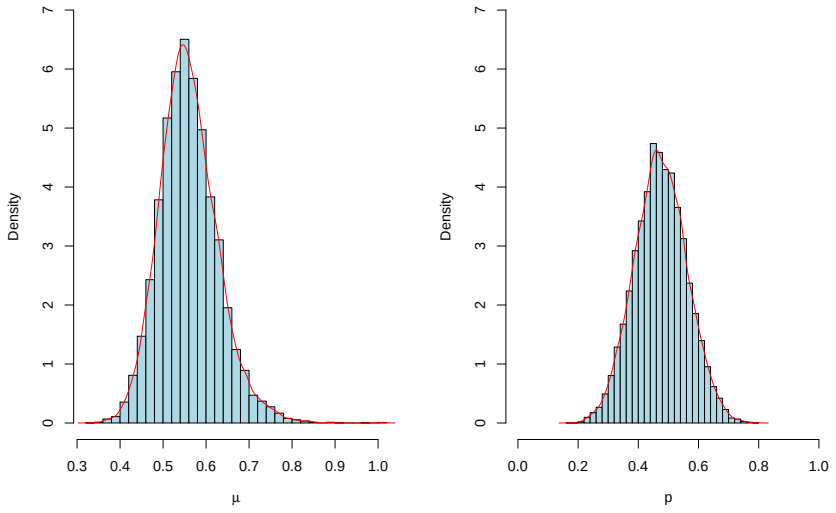

Figure 10: Posterior distribution of μ and $\Pr[X \leq 0.5]$

References

Dutang, C., Goulet, V., Pigeon, M. et al. (2008), 'actuar: An r package for actuarial science', *Journal of Statistical software* **25**(7), 1–37.

Team, S. D. et al. (2018), 'Rstan: the r interface to stan. r package version 2.17. 3'.

4 Appendix

```
rm(list = ls(all=TRUE))
setwd("~/Documents/Masters 2020:2021/Bayesian DM/BDM Assignment 2")
library(Rfast)
library(actuar)
set.seed(2020)

#####################################
##Question 1
# data
N <- c(4,7,3)
i <- c(1,2,3)
X1j <- c(2.588, 2.364, 0.153, 0.418)
X2j <- c(0.016, 0.243, 0.330, 0.167, 2.507, 1.091, 1.311)
X3j <- c(1.431, 0.444, 0.053)
X <- list(X1j,X2j,X3j)

# Z, Y
YY <- function(i){
  prod(X[[i]]+1)
}

ZZ <- function(i){
  log(YY(i))
}

Y <- c(YY(1), YY(2), YY(3))
Z <- c(ZZ(1), ZZ(2), ZZ(3))

# MLE estimation
# log Likelihood function
LL <- function(alphs){
  # alphs = vector of 3 alphas

  # value of likelihood
  val <- 1
  for (i in 1:3){
    val <- val*alphs[i]^N[i]/(Y[i])^(alphs[i]+1)
  }
```

```
  # take log
  # negative since optim minimizes
  return(-log(val))
}

# optim performs minimization
mle <- optim(par = runif(3), fn = LL)
mle$par #1.341256 1.998522 2.295341
N/Z # from differentiating - same

###################
# functions for accrej gibbs
gamm <- function(a){
  ni <- N[n]
  zi <- Z[n]

  ((zi+lam)^(ni-1))*a^ni*exp(-a*(zi+lam))/gamma(ni-1)
}

pareto <- function(a){
  # in form given by actuar
  2*(sum(alphsk))^2/(a+sum(alphsk))^3
}

# prob. integral transform
randpar <- function(){
  # random uniform no
  p <- runif(1)

  # return inv cdf of p
  sum(alphsk)/sqrt(1-p) - sum(alphsk)
}

c.optim <- function(a){
  # return -log(pi*g/h) to minimize
  val <- log(gamm(a))-log(pareto(a))

  return(-val)
}
```

```r
# starting values
alphs <- mle$par
lam <- 1/mean(alphs)

# how many samples?
ns <- 4000

# storage of alpha series
A <- matrix(0, nrow = ns+1, ncol = 3)
A[1,] <- alphs

####### Gibbs with Accept-Reject
set.seed(2020)
for (s in 2:(ns+1)){
  for (n in 1:3){
    # set acc indicator
    acc <- FALSE
    # set alpha and alphsk
    alphsk <- alphs[-n]
    alpha <- alphs[n]

    # find C to minimize gamma ratio
    alphastar <- nlm(c.optim, 2)$estimate
    C <- gamm(alphastar)/pareto(alphastar)

    while(!acc){ # loop until sample is accepted
      # generate candidate from pareto
      x <- randpar()
      #x <- rpareto(n = 1, shape = 2, scale = sum(alphsk))

      # gamma ratio
      g <- gamm(x)/(C*pareto(x))

      # generate u
      u <- runif(1)

      # accept-reject
      if(g>=u){
        A[s,n] <- x          # add to matrix
        alphs[n] <- x        # update alpha vector
```

```
        lam <- 1/mean(alphs)# update lambda
        acc <- TRUE         # terminate if accepted
      }
    }
  }
}

# estimates
# burn-in
burn <- 100
# sample with burn-in removed
samp <- A[-c(1:(burn+1)),]
# posterior mean
alphahat <- matrix(colmeans(samp), ncol = 3, nrow=1)
ones <- matrix(1, nrow = ns-burn, ncol = 1)
# standard error
means <- ones%*%alphahat
SEs <- colSums((samp-means)^2)/(nrow(samp)-1)

# traceplots
pdf("q1trace.pdf", width = 12, compress = FALSE)
par(mfrow=c(3,1))
for (i in 1:3){
  plot(A[,i], type = "l", ylab = expression(alpha),
       main = paste(expression(i), "=",i, sep = " "), col = "red")
  abline(v=100, col = "darkgrey")
  abline(h=alphahat[i], lwd = 2, col = "blue")
  abline(h = alphahat[i]+SEs[i], lty = 2, col = "grey", lwd = 2)
  abline(h = alphahat[i]-SEs[i], lty = 2, col = "grey", lwd = 2)
}
dev.off()

# autocorrelations
pdf("q1acf.pdf", width = 10, compress = F)
par(mfrow=c(3,1))
for (i in 1:3){
  acf(samp[,i], main = paste("i =", i, sep = " "))
}
dev.off()
```

```
# histograms
pdf("q1hists.pdf", width = 12, compress = FALSE)
par(mfrow=c(1,3))
for(i in 1:3){
  hist(samp[,i],
       main = paste(expression(i), "=",i,sep = " "),
       xlab = bquote(alpha),
       breaks = 40, freq = F, col = "lightblue", ylim = c(0,0.8))
  lines(density(samp[,i]), col = "red", lwd = 2)
  abline(v = alphahat[i], lty = 2, col = "blue", lwd = 2)
  abline(v = alphahat[i]+SEs[i], lty = 2, col = "grey", lwd = 2)
  abline(v = alphahat[i]-SEs[i], lty = 2, col = "grey", lwd = 2)
}
dev.off()

#######################################
# Question 2
library(rstan)

#data
N <- c(3, 7, 4, 8, 5, 9)
X <- c(10, 33, 3, 39, 5, 50)
lamdahatdata <- X/N # scale by # weeks
mean(lamdahatdata) # sample mean

# data as named list
brkdwns <- list(J=length(N),X=X, N=diag(N))

# diffused prior mu
# uniform prior sigma
sfile1 <- c(
  "data{
  int<lower=0> J;          // number of facilities
  int<lower=0> X[J];       // number of breakdowns at facility J
  matrix[J,J] N;          // number of weeks observed
  }
  parameters{
  real mu;
  real<lower=0> sigma;
  vector[J] lambda;       // lambda_i of poisson
```

```
  }
  transformed parameters {
  vector[J] theta;
  theta = N * lambda;
  }
  model{
  sigma ~ uniform(0.1, 0.5);
  lambda ~ lognormal(mu, sigma);
  X ~ poisson(theta);
  }
  generated quantities{
  real lamhat;
  real xpred;
  lamhat = lognormal_rng(mu, sigma);
  xpred = poisson_rng(6*lamhat);
  }")

set.seed(2020)
fit1 <- stan(
  model_code = sfile1,
  data = brkdwns,
  chains = 6,
  warmup = 2000,
  iter = 4000,
  cores = 2,
  refresh = 0
)

#estimates and CI
res <- summary(fit1, pars = "lambda")$summary # lambda estimates
#xtable(res)

# joint dist of sigma and mu
pdf("q1sigmu.pdf", width = 10, compress = FALSE)
pairs(fit1, pars = c("sigma", "mu"))
dev.off()

#traceplots
pdf(file = 'q2sigma.pdf', compress = F, width = 10)
traceplot(fit1, pars = c("sigma"), include = TRUE, inc_warmup = TRUE)
```

```
dev.off()
#
pdf(file = 'q2lambda.pdf', compress = F, width = 10)
traceplot(fit1, pars = c("lambda"), include = TRUE, inc_warmup = TRUE)
dev.off()
#
pdf(file = 'q2mu.pdf', compress = F, width = 10)
traceplot(fit1, pars = c("mu"), include = TRUE, inc_warmup = TRUE)
dev.off()

# predictive dist
# extract pars
mu <- summary(fit1, pars = c("mu", "sigma"))$summary[1]
sigma <- summary(fit1, pars = c("mu", "sigma"))$summary[2]

# plotting
nx <-1000000
lams <- rlnorm(n = nx, meanlog = mu, sdlog = sigma)
x7 <- rpois(n = nx, lambda = 6*lams)

pdf(file = 'q2preddist.pdf', compress = F, width = 10)
hist(x7, breaks = 60, freq = F, main = "",
     xlab = expression("x"), col ="lightblue",
     ylim = c(0,0.05), xlim = c(0, 70))
lines(density(x7), col = "red", lwd = 2)
dev.off()

# use generated values to verify
xpred <- extract(fit1, "xpred")
#pdf(file = 'q2preddist.pdf', compress = F, width = 10)
hist(xpred$xpred, xlim=c(0,100), breaks=60,
     freq=FALSE, col = "lightblue", ylim = c(0,0.05))
lines(density(xpred$xpred), col = "red", lwd = 2)
#lines(density(x7), col = "red", lwd = 2)
#dev.off()

#################
### sensitivity analysis
# uninformative prior for sigma, N(1,1) for mu
sfile2<- c(
```

```
  "data{
  int<lower=0> J;          // number of facilities
  int<lower=0> X[J];       // number of breakdowns at facility J
  matrix[J,J] N;          // number of weeks observed
  }
  parameters{
  real mu;
  real<lower=0> sigma;
  vector[J] lambda;       // lambda_i of poisson

  }
  transformed parameters {
  vector[J] theta;
  theta = N * lambda;
  }
  model{
  mu ~ normal(1,1);
  sigma ~ uniform(0.1, 1);
  lambda ~ lognormal(mu, sigma);
  X ~ poisson(theta);
  }")

set.seed(2020)
fit1 <- stan(
  #file = sfile.stan, # what goes here?
  model_code = sfile2,
  data = brkdwns,
  chains = 8,
  warmup = 2000,
  iter = 4000,
  cores = 2,
  refresh = 0
)

#estimates and CI
plot(fit1)
print(fit1)

#results
res <- summary(fit1, pars = "lambda")$summary
```

```r
#xtable(res)

plot(fit1, pars = c("lambda"))
plot(fit1, pars = c("sigma"))
pairs(fit1, pars = c("sigma", "mu"))

#traceplots to check convergence
pdf("trcase1.pdf", width = 10, compress = F)
traceplot(fit1, pars = c("sigma"))
dev.off()

traceplot(fit1, pars = c("lambda"))
traceplot(fit1, pars = c("mu"))

##### mu N(1,1), sigma gamma
#choosing Gamma pars
#which pars maximise probability between 0.1 and 0.5
max.prob <- function(pars){
  s <- pars[1]
  r <- pars[2]

  diff<-pgamma(q=0.5,shape=s,rate=r) - pgamma(q=0.1,shape=s,rate=r)
  return(-diff)
}
max <- nlm(f = max.prob, p = c(2,8))
s <- max$estimate[1] #106.1765
r <- max$estimate[2] #347.4632

xser <- seq(0, 0.6, 0.01)
par(mfrow=c(1,1))
plot(x=xser, y=sapply(X = xser, FUN = dgamma, shape = 2, rate =8),
     xlim=c(0,0.6), type = "l")

abline(v =0.1)
abline(v=0.5)
#what prob do we get
-max.prob(c(2,8))
####

#data
```

```
N <- c(3, 7, 4, 8, 5, 9)
X <- c(10, 33, 3, 39, 5, 50)

# data as named list
brkdwns <- list(J=length(N),X=X, N=diag(N), R=8, S=2)

sfile3 <- c(
  "data{
  int<lower=0> J;          // number of facilities
  int<lower=0> X[J];       // number of breakdowns at facility J
  matrix[J,J] N;          // number of weeks observed
  real<lower=0> R;       // pars for gamma prior of sigma
  real<lower=0> S;
  }
  parameters{
  real mu;
  real<lower=0> sigma;
  vector[J] lambda;       // lambda_i of poisson
  }
  transformed parameters {
  vector[J] theta;
  theta = N * lambda;
  }
  model{
  mu ~ normal(1,1);
  sigma ~ gamma(S, 1/R); // gamma defined differently R vs Stan
  lambda ~ lognormal(mu, sigma);
  X ~ poisson(theta);
  }
  generated quantities{
  real lamhat;
  real xpred;
  lamhat = lognormal_rng(mu, sigma);
  xpred = poisson_rng(6*lamhat);
  }")

set.seed(2020)
fit3 <- stan(
  #file = sfile.stan, # what goes here?
  model_code = sfile3,
```

```
  data = brkdwns,
  chains = 6,
  warmup = 2000,
  iter = 4000,
  cores = 2,
  refresh = 0
)

#estimates and CI
res <- summary(fit3, pars = "lambda")$summary # lambda estimates
#xtable(res)

################################################
# Question 3
# data
X <- c(0.32, 0.63, 0.73, 0.38, 0.54, 0.95, 0.60, 1.03,
       0.66, 0.41, 0.48, 0.27, 0.60, 0.21, 0.39, 0.28,
       1.03, 0.68, 0.10, 0.69)

equipdata <- list(X = X, J = length(X),
                  anu = -log(0.05)/log(3))
#using generated quantities
equiprun2 <- c(
  "data{
  int<lower=0> J;          // number of observations
  real<lower=0> X[J];      //
  real anu;                // alpha_nu constant
  }
  parameters{
  real<lower=0> nu;          // nu of weibull (failure rate, dist X)
  real sigma;                // sigma of weibull (dist X)
  }
  transformed parameters {
  real mu;                   // mean of x
  mu = sigma * tgamma(1 + 1/nu);
  }
  model{
  nu ~ pareto(1, anu);         // (y_min, alpha)
  sigma ~ uniform(0.2/(log(2)) ^ (1/nu), 1.5/(log(2)) ^ (1/nu));
  X ~ weibull(nu, sigma);      // (alpha, sigma)
```

```
  }
  generated quantities{
  real<lower=0> p;           // Pr[X <= 0,5]
  p = weibull_cdf(0.5, nu, sigma);
  }")

set.seed(2020)
fitequip2 <- stan(
  model_code = equiprun2,
  data = equipdata,
  chains = 6,
  warmup = 2000,
  iter = 4000,
  cores = 2,
  refresh = 0
)

# results
#traceplots
#check convergence
traceplot(fitequip2, pars = "nu")
traceplot(fitequip2, pars = "sigma")

#estimates
xtable(summary(fitequip2)$summary)

# tracepplots for mu and p
pdf("q3mutrace.pdf", width = 10, compress = F)
par(mfrow=c(1,2))
traceplot(fitequip2, pars = "mu", include = TRUE, inc_warmup = TRUE)
dev.off()

pdf("qptrace.pdf", width = 10, compress = F)
traceplot(fitequip2, pars = "p", include = TRUE, inc_warmup = TRUE)
dev.off()

# required plots
# expectation and CI
mures <- summary(fitequip2)$summary[3,]
pres <- summary(fitequip2)$summary[4,]
```

```
#xtable(rbind(mures, pres))

#histograms
#mu and prob(X<0.5)
pdf("q3mu.pdf", width = 10, compress = F)
par(mfrow=c(1,2))
hist(extract(fitequip2)$mu, breaks = 40, col = "lightblue",
     xlab = expression(mu), freq = F, main = "", ylim = c(0,7))
lines(density(extract(fitequip2)$mu), col = "red")
hist(extract(fitequip2)$p, breaks = 40, col = "lightblue",
     xlab = expression(p), freq = F,
     main = "", ylim = c(0,7), xlim = c(0,1))
lines(density(extract(fitequip2)$p), col = "red")
dev.off()
```