Bayesian Data Analysis Assignment 1

Benjamin Cox, S1621312

Question 1

a)

Our probability vector is $\theta = (\theta_1, \dots, \theta_6)$ and our outcome vector is $c = (c_1, \dots, c_6)$. We are drawing from a multinomial distribution (in the same way that 10 Bern(p) trials are the same as 1 Bin(10,p) trial distributionally), ie

$$c \sim \text{Multinomial}(120, \theta).$$

Therefore the likelihood of θ given c with n trials is the following:

$$L(\theta|c) = \frac{n!}{c_1!c_2!\cdots c_6!}\theta_1^{c_1}\cdots\theta_6^{c_6}.$$

A suitable conjugate prior for this would be the Dirichlet distribution (K is the number of possible outcomes, in our case 6),

$$f(x|\alpha, K) = \frac{\Gamma(\sum_{i=1}^{K}\alpha_i)}{\prod_{i=1}^{k}\Gamma(\alpha_i)}\prod_{i=1}^{K}x_i^{\alpha_i - 1}.$$

The Jeffrey's prior for the multinomial corresponds to a Dirichlet distribution with

$$\alpha_i = 1/2 \ \forall i \in \{1, \dots, K\}.$$

The above results (conjugacy and Jeffreys prior) are given in [1].

b)

Our posterior distribution for θ is

$$\begin{aligned} p(\theta|c) &\propto \left(\frac{\Gamma(3)}{\Gamma(0.5)^6}\theta_1^{-1/2}\cdots\theta_6^{-1/2}\right)\left(\frac{n!}{c_1!c_2!\cdots c_6!}\theta_1^{c_1}\cdots\theta_6^{c_6}\right) \\ &= \text{Dirichlet}\left(\alpha = c + 0.5, K = 6\right). \end{aligned}$$

The expected value of the Dirichlet Distribution is given by $\mathbb{E}\left[X_i\right] = \frac{\alpha_i}{\sum \alpha_i}$, so in our case

$$\mathbb{E}\left[\theta_i|c\right] = \frac{c_i + \frac{1}{2}}{\sum c_i + 3}.$$

This corresponds to values of

$\mathbb{E}[\theta_1]$	$\mathbb{E}[\theta_2]$	$\mathbb{E}[\theta_3]$	$\mathbb{E}[\theta_4]$	$\mathbb{E}[\theta_5]$	$\mathbb{E}[\theta_6]$
0.142	0.199	0.183	0.142	0.199	0.134

We are going to compute symmetric 95% credible intervals for each θ_i, hence we must marginalise them. We could (theoretically) calculate a 95% credible region in the 6 dimensional parameter space, but this would get extremely complicated really quickly, and would also be hard to interpret.

Fortunately the marginal distributions of the Dirichlet are a lot easier, as they are beta distributions. Write $\alpha_0 = \sum \alpha_k$, then we have

$$\theta_i \sim \text{Beta}\left(\alpha_i, \alpha_0 - \alpha_i\right).$$

We can substitute in our expressions for α_i to obtain

$$\theta_i \sim \text{Beta}\left(c_i + 1/2, c_0 - c_i + 2.5\right).$$

Using this result we obtain the following 95% credible intervals:

θ	Lower	Upper
θ_1	0.08657456	0.20897331
θ_2	0.1337529	0.2738778
θ_3	0.1200040	0.2556043
θ_4	0.08657456	0.20897331
θ_5	0.1337529	0.2738778
θ_6	0.08007849	0.19945622

c)

We are going to test the null hypothesis of a fair die. For the multinomial distribution there is no easy quantile function, so we are going to use the likelihood-ratio test and Pearson's χ^2 test. These approach the true p-value from below and above respectively, so doing both should give us a very good idea.

Computing the p-value using the likelihood-ratio test we get $p_{lr} = 0.377$. Computing the p-value using Pearson's χ^2 test we obtain $p_{\chi^2} = 0.377$. Thus we can be fairly sure that this is very close to the true p-value. Therefore we do not reject the null hypothesis that the die is fair.

d)

The posterior predictive distribution is the 'Dirichlet-Multinomial' distribution. The pmf for this is given by

$$f(x|n,\alpha) = \frac{n!\Gamma(\sum \alpha_i)}{\Gamma(n + \sum \alpha_i)} \prod_{k=1}^{K} \frac{\Gamma(x_k + \alpha_k)}{(x_k!)\Gamma(\alpha_k)}$$

for n the number of trials and $alpha_1, \ldots, \alpha_k > 0$.

Taken as our posterior predictive under the Jeffrey's prior we have

$$c_{\text{new}} \sim \text{DirMNom}(60, \text{c} + 1/2).$$

We can simulate from this. We draw 10,000 times from this distribution and find that with probability 0.737 we have more 5s than 6s in our next 60 trials.

e)

We incorporate these into our likelihood, denoting the new count vector as d. Our new posterior is

$$\theta \sim \text{Dirichlet}(c + d + 1/2, 6).$$

Our new posterior means are

$\mathbb{E}[\theta_1]$	$\mathbb{E}[\theta_2]$	$\mathbb{E}[\theta_3]$	$\mathbb{E}[\theta_4]$	$\mathbb{E}[\theta_5]$	$\mathbb{E}[\theta_6]$
0.163	0.183	0.216	0.138	0.167	0.138

with 95% marginal credible intervals given by

θ	Lower	Upper
θ_1	0.1189814	0.2113785
θ_2	0.1371556	0.2340370
θ_3	0.1667039	0.2698204
θ_4	0.0975285	0.1838316
θ_5	0.1225962	0.2159303
θ_6	0.0939960	0.1791973

The credible intervals have narrowed, as expected for more observations. It is of note that the new credible interval for θ_3 (barely) does not contain the value required for a 'fair' dice. This is evidence that the dice is not fair.

Question 2

a)

We have two equally credible opinions on the distribution of $1/\lambda$. We have one stating that it lies mostly between 5 and 10, and another that states that it lies between 0 and 25. We are going to use the Gamma distribution as our prior for λ.

We are going to calculate parameters for this such that the corresponding inverse gamma distributions has mean of the midpoint and standard deviation of 1/4 the range.

Calculating these parameters we obtain

$$\alpha_1 = 38, \beta_1 = 277.5, \qquad \alpha_2 = 6, \beta_2 = 62.5,$$

with the indices corresponding to the originating expert.

This means that our mixture prior is of the form

$$\lambda \sim 0.5 \cdot \Gamma(38, 277.5) + 0.5 \cdot \Gamma(6, 62.5)$$

b)

We have 100 points of data with a mean of 9.877 minutes. We calculate our posterior. The gamma distribution is a conjugate prior to the exponential distribution, so it is quite simple to calculate our new parameters.

We calculate

$$\alpha_1^{post} = 138, \beta_1^{post} = 1265.2, \qquad \alpha_2^{post} = 106, \beta_2^{post} = 1050.2$$

using the formulae $\alpha^{post} = \alpha + n, \beta^{post} = \beta + \sum_i x_i = \beta + n\bar{x}$. To calculate the posterior mixing proportion Q we use

$$Q = \frac{q \frac{\beta_1^{\alpha_1}}{\Gamma(\alpha_1)} \frac{\Gamma(\alpha_1^{post})}{\beta_1^{post\ \alpha_1^{post}}}}{q \frac{\beta_1^{\alpha_1}}{\Gamma(\alpha_1)} \frac{\Gamma(\alpha_1^{post})}{\beta_1^{post\ \alpha_1^{post}}} + (1-q) \frac{\beta_2^{\alpha_2}}{\Gamma(\alpha_2)} \frac{\Gamma(\alpha_2^{post})}{\beta_2^{post\ \alpha_2^{post}}}}.$$

This quantity involves extremely large numbers, so we work on the log scale and exponentiate at the end.

Overall we obtain a posterior of the form

$$p(\lambda|x) = 0.40455 \cdot \Gamma(138, 1265.2) + 0.59545 \cdot \Gamma(106, 1050.2).$$

Of course the posterior for $1/\lambda$ is analogous, with inverse gamma rather than gamma.

We calculate a posterior 95% credible interval for $1/\lambda$ as $(8.00, 11.84)$, and a mean for $1/\lambda$ of 9.69.

To calculate the posterior probability of waiting for more than 20 minutes we need the posterior predictive distribution. We find that

$$\Pr[x > 20] = 0.127.$$

We note that the posterior distribution is a gamma mixture whereas the posterior predictive distribution is a generalised Pareto distribution. The pdf is given by

$$p(t_*|t) = 0.40455\ f_l(\beta_1^{post}, \alpha_1^{post}) + 0.59545\ f_l(\beta_2^{post}, \alpha_2^{post}),$$

where

$$f_l(x|a, b) = \frac{a}{b}\left(1 + \frac{x}{b}\right)^{-(a-1)}.$$

This family of distributions is often used in survival analysis to model survival times, so it is natural that it could turn up here.

c)

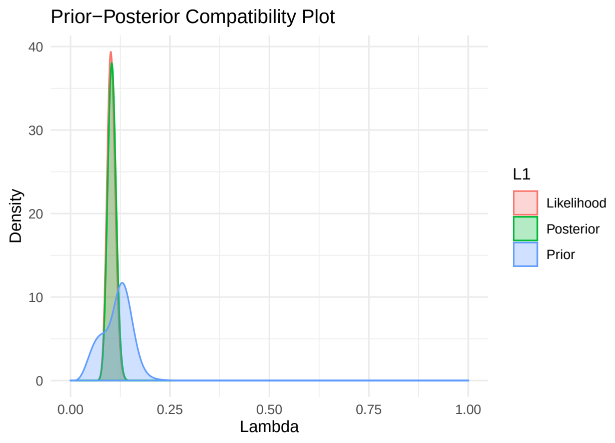

Figure 1: Prior-Likelihood-Posterior Plot

As seen in Figure (1) the posterior is highly influenced by the likelihood and not so much by the prior. The prior is compatible with the data as an initial estimate, and and served it's purpose well as the posterior has converged (mostly) onto the likelihood.

The prior is compatible as there is no conflict between the data and the posterior.

d)

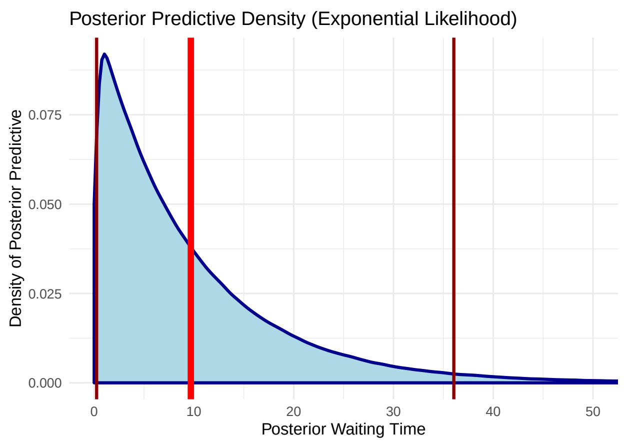

Figure 2: Posterior Predictive Density Plot, bright red line indicates mean, darker red lines indicate 95% CI.

We plot our posterior predictive density. We obtain a mean of 9.69 with symmetric 95% credible interval given by (0.24, 36). This is very much in line with what we expect from the likelihood, indicating good fit.

e)

We are now to take the Lindley distribution as our likelihood. This distribution has pdf of

$$f(t|\lambda) = \frac{\lambda^2}{\lambda+1}(1+t)e^{-\lambda t} \implies \ln(f(t|\lambda)) = 2\ln(\lambda) - \ln(\lambda+1) + \ln(1+t) - \lambda t.$$

The log pdf is important for implementing into our MC algorithm.

Given our data we have likelihood of the form

$$L(\lambda|\underline{t}) = \left(\frac{\lambda^2}{\lambda+1}\right)^n \exp\left(-\lambda\sum_{i=1}^n t_i\right)\prod_{i=1}^n(1+t_i),$$

hence we have a posterior of the form

$$p(\lambda|\underline{t}) \propto (\Gamma(38, 277.5) + \Gamma(6, 62.5))\left(\frac{\lambda^2}{\lambda+1}\right)^n \exp\left(-\lambda\sum_{i=1}^n t_i\right)\prod_{i=1}^n(1+t_i)$$
$$\propto \left(\frac{277.5^{38}}{\Gamma(38)}\lambda^{37}e^{-277.5\cdot\lambda} + \frac{62.5^6}{\Gamma(6)}\lambda^5 e^{-62.5\cdot\lambda}\right)\left(\frac{\lambda^2}{\lambda+1}\right)^n \exp\left(-\lambda\sum_{i=1}^n t_i\right)\prod_{i=1}^n(1+t_i)$$

We calculate our posterior. We obtain an expected waiting time of 10.4 minutes, an increase from our previous.

We also plot the new posterior predictive.

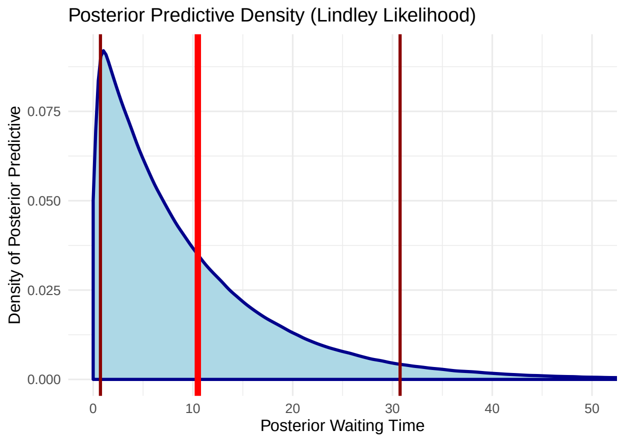

Figure 3: Posterior Predictive Density Plot for Lindley Likelihood, bright red line indicates mean, darker red lines indicate 95% CI.

Notice that the credible interval for the posterior predictive derived from the Lindley likelihood are a lot tighter.

f)

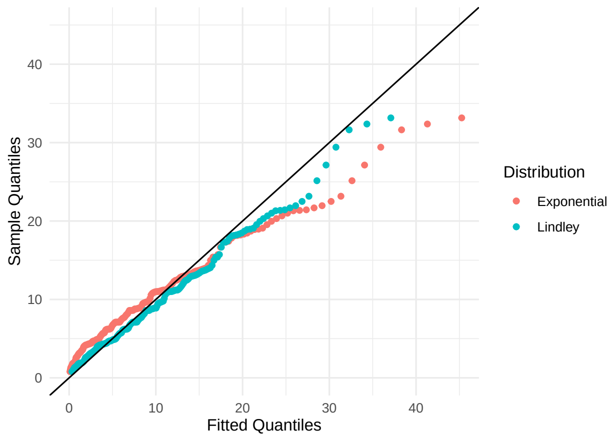

Figure 4: Q-Q plot of sample quantiles against posterior predictive quantiles

We see that the Lindley distribution fits the data much better than the exponential distribution. We perform Kolmogorov-Smirnoff tests comparing our posterior predictives and the data. For the Lindley likelihood we obtain a distance of 0.0685 with p-value of 0.736. For the exponential likelihood we obtain a distance of 0.1797 with corresponding p-value of 0.003.

This is extremely good evidence that the Lindley distribution is a better fit for the data than the Exponential distribution.

Question 3

a)

In order to make our results directly comparable we are going to mean centre the data before we fit the initial linear model. This should only affect the intercept term.

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	0.2439	0.0643	3.79	0.0002
SexI	-0.8249	0.1024	-8.06	0.0000
SexM	0.0577	0.0833	0.69	0.4887
Length	-0.4583	1.8091	-0.25	0.8000
Diameter	11.0751	2.2273	4.97	0.0000
Height	10.7615	1.5362	7.01	0.0000
Whole.weight	8.9754	0.7254	12.37	0.0000
Shucked.weight	-19.7869	0.8174	-24.21	0.0000
Viscera.weight	-10.5818	1.2937	-8.18	0.0000
Shell.weight	8.7418	1.1247	7.77	0.0000

This points to length and Male-Female difference not having a significant effect on age. Length is likely due to high correlation with height (Pearson correlation of 0.828) and diameter (Pearson correlation of 0.987). Most of the variables have extremely high correlation, so this is going to require a long MCMC run.

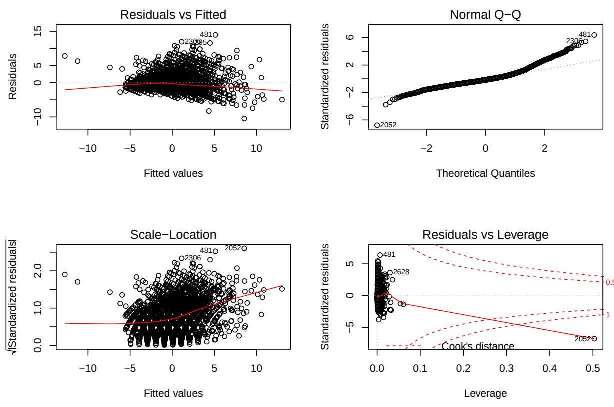

Figure 5: Diagnostic plots for the standard linear model

Looking at diagnostic plots we see a large number of extreme observations with high leverage. Therefore it makes sense to perform a robust regression using t distributed errors.

b)

We are going to fit a Bayesian linear model of the form

$$\begin{aligned} y &\sim t_{\nu+1}(X\beta, \sigma), \\ \beta_i &\sim \text{Normal}(0, 10^4), \\ \sigma &\sim \Gamma^{-1}(0.1, 0.1), \\ \nu &\sim \text{Exponential}(0.034), \end{aligned}$$

where X is the $n \times p$ centred design matrix, β is the p vector of regression coefficients, and the t distribution is understood to be shifted and scaled such that $\frac{y - X\beta}{\sigma}$ has a classic t distribution with $\nu + 1$ degrees of freedom.

c)

Fitting the model we obtain the following quantities:

	mean	se_mean	sd	2.5%	25%	50%	75%	97.5%	n_eff	Rhat
beta[1]	-0.22	0.00	0.06	-0.33	-0.25	-0.22	-0.18	-0.11	28239.55	1.00
beta[2]	-0.67	0.00	0.08	-0.83	-0.73	-0.67	-0.61	-0.51	32862.97	1.00
beta[3]	0.12	0.00	0.07	-0.02	0.07	0.12	0.17	0.26	33774.18	1.00
beta[4]	1.88	0.01	1.51	-1.13	0.86	1.89	2.90	4.82	31346.96	1.00
beta[5]	7.28	0.01	1.86	3.64	6.02	7.27	8.53	10.96	31154.47	1.00
beta[6]	15.80	0.01	2.04	11.81	14.43	15.80	17.17	19.80	50288.95	1.00
beta[7]	7.51	0.01	0.80	5.94	6.97	7.51	8.04	9.06	23090.61	1.00
beta[8]	-16.35	0.01	0.90	-18.12	-16.95	-16.35	-15.74	-14.59	26678.37	1.00
beta[9]	-9.22	0.01	1.20	-11.56	-10.03	-9.22	-8.41	-6.89	34525.85	1.00
beta[10]	6.72	0.01	1.17	4.43	5.92	6.71	7.50	9.05	28005.24	1.00
sigma	1.39	0.00	0.03	1.33	1.37	1.39	1.41	1.45	35667.27	1.00
nu	1.87	0.00	0.16	1.58	1.76	1.86	1.97	2.19	35705.46	1.00
y_new[1]	-0.34	0.01	2.47	-4.85	-1.41	-0.34	0.74	4.22	56374.07	1.00
lp__	-6470.43	0.02	2.47	-6476.21	-6471.85	-6470.08	-6468.65	-6466.64	21223.39	1.00

Table 1: Regression Coefficient Table

d)

The model table contains our $\hat{R}$ values, and shows that we have good reason to believe that the chains have converged.

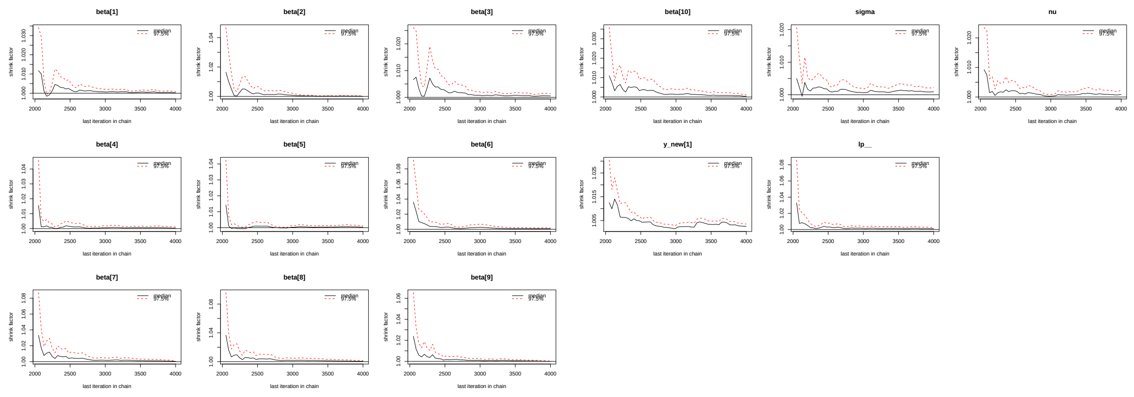

Figure 6: Plots of $\hat{R}$ as chains progressed

Although Figure 6 is a bit small it is clear that after our burn-in $\hat{R}$ was very close to 1, and got closer as we computed more iterations.

e)

We are going to alter some of the prior hyperparameters in order to test sensitivity. We are going to assume the following model:

$$
\begin{aligned}
y &\sim t_\nu(X\beta, \sigma),\\
\beta_i &\sim \text{Normal}(0, 10),\\
\sigma &\sim \Gamma^{-1}(7, 30),\\
\nu &\sim \text{Exponential}(0.034),
\end{aligned}
$$

The hyperparameters are quite different, so this should be a good test of prior sensitivity. We are going to run the model for less iterations as we do not intend to use this model for inference, merely for prior sensitivity analysis.

We obtain the following table of results:

	mean	se_mean	sd	2.5%	25%	50%	75%	97.5%	n_eff	Rhat
beta[1]	-0.21	0.00	0.06	-0.32	-0.25	-0.21	-0.17	-0.10	5572.99	1.00
beta[2]	-0.68	0.00	0.08	-0.84	-0.73	-0.68	-0.62	-0.51	6239.80	1.00
beta[3]	0.12	0.00	0.07	-0.02	0.07	0.12	0.17	0.25	6566.89	1.00
beta[4]	1.98	0.02	1.47	-0.89	0.97	1.98	2.99	4.82	6567.40	1.00
beta[5]	7.23	0.02	1.80	3.71	5.98	7.24	8.46	10.72	6482.90	1.00
beta[6]	15.25	0.02	2.01	11.29	13.91	15.27	16.63	19.13	9565.22	1.00
beta[7]	7.36	0.01	0.78	5.82	6.83	7.36	7.89	8.86	4357.55	1.00
beta[8]	-16.20	0.01	0.90	-17.95	-16.81	-16.20	-15.60	-14.44	4957.90	1.00
beta[9]	-8.98	0.01	1.19	-11.31	-9.78	-9.00	-8.19	-6.64	6498.79	1.00
beta[10]	6.93	0.02	1.15	4.74	6.15	6.91	7.70	9.25	5590.50	1.00
sigma	1.40	0.00	0.03	1.34	1.38	1.40	1.42	1.46	6930.44	1.00
nu	1.90	0.00	0.16	1.61	1.79	1.89	2.00	2.22	7134.90	1.00
y_new[1]	-0.36	0.02	2.54	-4.77	-1.41	-0.35	0.72	4.13	10775.23	1.00
lp__	-6497.93	0.04	2.39	-6503.44	-6499.35	-6497.65	-6496.19	-6494.16	4370.59	1.00

This is nearly identical to the table obtained previously, so we can be confident in our results. All of the differences in means are well within their standard deviations, so overall we are happy.

f)

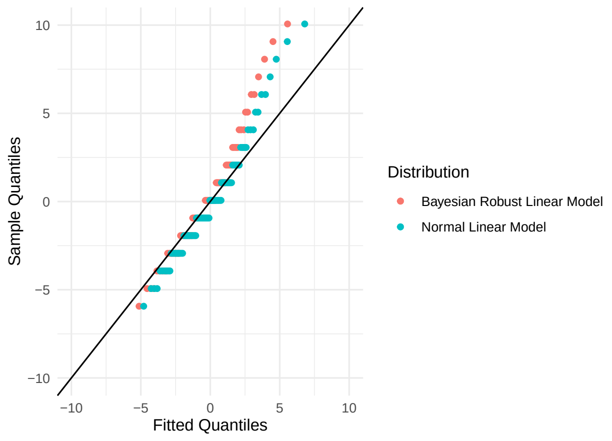

Figure 7: QQ plots comparing our models

Looking at figure 7 neither of our models fit the data very well. Performing KS tests we find a distance of 0.11827 between our predictions from the Robust Bayesian and the true data, compared with a distance of 0.15466 for the normal linear and the true data. We note that the true data is rather skewed with estimated skewness 1.11. This has not been compensated for in the model. This means that there are more old abalones than predicted by the models. This is likely due to their characteristics not changing that much after they age enough. This is shown by the deviation of the upper quantiles of the predictions from the sample quantiles.

g)

95% credible intervals are given in Table 1. We see that the credible intervals for β_3 and β_4 contain 0. These are the coefficients linked to 'isMale' and Length respectively. This observation is identical to that of the normal linear model.

h)

The 95% credible interval for the age relating to the new data is given in the Table 1 as $(-4.85, 4.22)$ with mean of -0.34. We plot the density of the posterior predictive and obtain the following:

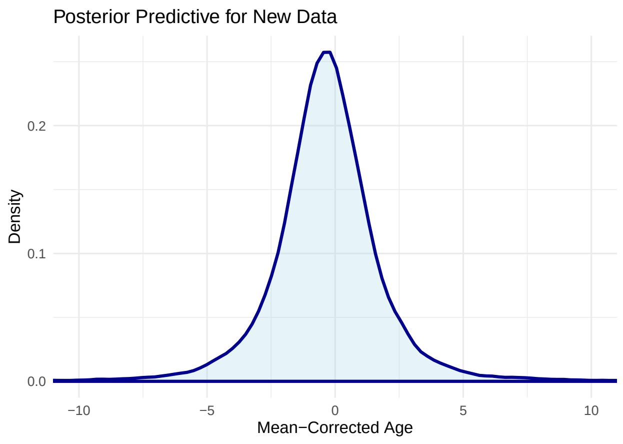

Figure 8: Posterior predictive density for new data

References

[1] Frank Tuyl. “A Note on Priors for the Multinomial Model”. In: *The American Statistician* 71.4 (2017), pp. 298–301.

A R Code for Question 1

```r
library(extraDistr)

obs <- c(17, 24, 22, 17, 24, 16)
obs2 <- c(22, 20, 30, 16, 16, 16)
value <- 1:6
names(obs) <- c("1", "2", "3", "4", "5", "6")
names(obs2) <- names(obs)
n1 <- sum(obs)
n2 <- sum(obs2)
ysum <- sum(obs * value)
#categorical
#jeffreys prior is dirichlet(a = 1/2)

#theta_1, ..., theta_6 ~ Dirichlet(a_1, ..., a_6)
#c_1, ..., c_6 ~ Multinomial(theta_1, ..., theta_6)
#a'_k = a_k + ysum

j_prior_params <- rep(1 / 2, 6)

post_params <- j_prior_params + obs
post_params_new <- j_prior_params + obs + obs2

post_pred <- post_params / sum(post_params)
post_pred_new <- post_params_new / sum(post_params_new)

alpha = 0.05
for (i in post_params){
  print(qbeta(c(0.025, 0.975), i, sum(post_params) - i))
}
for (i in post_params_new){
  print(qbeta(c(0.025, 0.975), i, sum(post_params_new) - i))
}

# n <- 1e6
# rope_r <- 0.05#1/30
# A <- rdirichlet(n, post_params)
# B <- as.matrix(A)
# f <- function(x) all((x > 1/6-rope_r)*(x < 1/6+rope_r))
# mean(apply(B, MARGIN = 1, FUN = f))

# nh_p <- rep(1/6, 6)
# n <- 1e2
# size <- sum(obs)
# nh_lik <- t(rmultinom(n, size, nh_p))

#pearson chi-squared

expected <- n1 * rep(1/6, 6)
observed <- obs
sq_diff <- (observed - expected)^2
pear_stat <- sq_diff / expected
chi_stat <- sum(pear_stat)
chi_dof <- length(obs) - 1
chi_p <- pchisq(chi_stat, chi_dof)
chi_p

nh <- rep(1/6, 6)
mle <- obs/n1
lr_stat <- -2 * sum(obs * log(nh/mle))
lr_chi_stat <- pchisq(lr_stat, chi_dof)
lr_chi_stat

n <- 1e5

C <- rdirmnom(n, 60, post_params)
D <- as.matrix(C)
ts <- D[,5] > D[,6]
sum(ts)/n
```

B R Code for Question 2

```r
library(tidyverse)
library(HDInterval)
library(extraDistr)

par(mfrow = c(1, 1))

waiting_times  <-
  c(
    0.8,
    0.8,
    1.3,
    1.5,
    1.8,
    1.9,
    1.9,
    2.1,
    2.6,
    2.7,
    2.9,
    3.1,
    3.2,
    3.3,
    3.5,
    3.6,
    4.0,
    4.1,
    4.2,
    4.2,
    4.3,
    4.3,
    4.4,
    4.4,
    4.6,
    4.7,
    4.7,
    4.8,
    4.9,
    4.9,
    5,
    5.3,
    5.5,
    5.7,
    5.7,
    6.1,
    6.2,
    6.2,
    6.2,
    6.3,
    6.7,
    6.9,
    7.1,
    7.1,
    7.1,
    7.1,
    7.4,
    7.6,
    7.7,
    8,
    8.2,
    8.6,
    8.6,
    8.6,
    8.8,
    8.8,
    8.9,
    8.9,
    9.5,
    9.6,
    9.7,
    9.8,
    10.7,
    10.9,
    11,
    11,
    11.1,
    11.2,
    11.2,
    11.5,
    11.9,
    12.4,
    12.5,
    12.9,
    13,
    13.1,
    13.3,
    13.6,
    13.7,
    13.9,
    14.1,
    15.4,
    15.4,
    17.3,
    17.3,
    18.1,
    18.2,
    18.4,
    18.9,
    19,
    19.9,
    20.6,
    21.3,
    21.4,
    21.9,
    23.0,
    27,
    31.6,
    33.1,
    38.5
  )

gammaShRaFromMeanSD = function(mean , sd) {
  if (mean <= 0)
    stop("mean must be > 0")
  if (sd <= 0)
    stop("sd must be > 0")
  shape = mean ^ 2 / sd ^ 2
  rate = mean / sd ^ 2
  return(list(shape = shape , rate = rate))
}

gammaShRaFromModeSD = function(mode , sd) {
```

```r
  if (mode <= 0)
    stop("mode must be > 0")
  if (sd <= 0)
    stop("sd must be > 0")
  rate = (mode + sqrt(mode ^ 2 + 4 * sd ^ 2)) / (2 * sd ^ 2)
  shape = 1 + mode * rate
  return(list(shape = shape, rate = rate))
}

iGammaShScFromMeanVar = function(mean, var) {
  shape <- (mean ^ 2 / var) + 2
  scale <- (mean ^ 3 / var) + mean
  return(list(shape = shape, scale = scale))
}

ex_1_prior <- iGammaShScFromMeanVar(7.5, (0.25 * (10 - 5)) ^ 2)
ex_2_prior <- iGammaShScFromMeanVar(12.5, (0.25 * (25 - 0)) ^ 2)
#prior is 0.5G(9, 0.6) + 0.5G(1,0.08)

n <- length(waiting_times)
mean_wait <- mean(waiting_times)

#posterior is Q*G(9+n, 0.6+n*xbar) + (1-Q)*G(12.5+n, 12.5+n*xbar)

p <- 0.5

find_mix_coefs_2gamma <- function(a1, b1, a2, b2, q1, n, xbar) {
  q2 <- 1 - q1
  a1post <- a1 + n
  a2post <- a2 + n
  b1post <- b1 + (n * xbar)
  b2post <- b2 + (n * xbar)
  log_c1 <-
    (a1 * log(b1) - lgamma(a1)) + (lgamma(a1post) - (a1post * log(b1post)))
  log_c2 <-
    (a2 * log(b2) - lgamma(a2)) + (lgamma(a2post) - (a2post * log(b2post)))
  log_Q <-
    (log(q1) + log_c1) - (log(q1) + log_c1 + log(1 + q2 / q1 * exp(log_c2 -
                                                                    log_c1)))
  #log_Q <- log(q1) + log_c1 - log(q1 * exp(log_c1) + q2 * exp(log_c2))
  Q <- exp(log_Q)
  # c1 <- ((b1^a1)/(gamma(a1))) * ((gamma(a1post))/ (b1post^a1post))
  # c2 <- ((b2^a2)/(gamma(a2))) * ((gamma(a2post))/ (b2post^a2post))
  # Q <- (q1 * c1)/(q1*c1 + q2*c2)
  return(c(Q, 1 - Q))
}

mcoefs <-
  find_mix_coefs_2gamma(ex_1_prior[[1]],
                        ex_1_prior[[2]],
                        ex_2_prior[[1]],
                        ex_2_prior[[2]],
                        p,
                        n,
                        mean_wait)

lambda <- seq(0, 1, len = 5000)

likelihood <-
  as.matrix(apply(as.array(waiting_times), 1, dexp, lambda))
likelihood <- apply(likelihood, 1, prod)

area = sfsmisc::integrate.xy(lambda, likelihood)
const = 1 / area
likelihood <- const * likelihood

prior <-
  p * dgamma(lambda, ex_1_prior[[1]], ex_1_prior[[2]]) + (1 - p) * dgamma(lambda, ex_2_prior[[1]], ex_2_prior[[2]])
posterior <-
  mcoefs[[1]] * dgamma(lambda, ex_1_prior[[1]] + n, ex_1_prior[[2]] + n *
                         mean_wait) + mcoefs[[2]] * dgamma(lambda, ex_2_prior[[1]] + n, ex_2_prior[[2]] + n *
                                                             mean_wait)

plot(
  lambda,
  prior,
  col = "darkgreen",
  ylab = "Density",
  xlab = expression(lambda),
  type = "l",
  ylim = c(0, 45),
  lwd = 2
)
lines(lambda, likelihood, col = "blue2", lwd = 2)
lines(lambda, posterior, col = "red", lwd = 2)
legend(
  "topright",
  legend = c("prior", "scaled likelihood", "posterior"),
  lty = c(1, 1, 1),
  lwd = c(2, 2, 2),
  col = c("darkgreen", "blue2", "red"),
  bty = "n"
)

#prior is relatively flat, posterior is very compatible with data and not very influenced by prior

post_mean <-
  mcoefs[[1]] * ((ex_1_prior[[1]] + n) / (ex_1_prior[[2]] + n * mean_wait)) + mcoefs[[2]] *
  ((ex_2_prior[[1]] + n) / (ex_2_prior[[2]] + n * mean_wait))
post_mean_wait <- 1 / post_mean

posterior_ci <-
  mcoefs[[1]] * qgamma(c(0.025, 0.975), ex_1_prior[[1]] + n, ex_1_prior[[2]] + n *
                         mean_wait) + mcoefs[[2]] * qgamma(c(0.025, 0.975), ex_2_prior[[1]] + n, ex_2_prior[[2]] + n *
                                                             mean_wait)

pci_area <-
```

```r
  sfsmisc::integrate.xy(lambda, posterior, posterior_ci[[1]], posterior_ci[[2]])


time <- seq(0, 30, len = 5000)

post_wait <-
  mcoefs[[1]] * dinvgamma(time, ex_1_prior[[1]] + n, ex_1_prior[[2]] + n *
                            mean_wait) + mcoefs[[2]] * dinvgamma(time, ex_2_prior[[1]] + n, ex_2_prior[[2]] + n *
                                                                 mean_wait)

post_wait_ci <-
  mcoefs[[1]] * qinvgamma(c(0.025, 0.975), ex_1_prior[[1]] + n, ex_1_prior[[2]] + n *
                            mean_wait) + mcoefs[[2]] * qinvgamma(c(0.025, 0.975), ex_2_prior[[1]] + n, ex_2_prior[[2]] + n *
                                                                 mean_wait)
post_wait_20 <-
  mcoefs[[1]] * pinvgamma(20, ex_1_prior[[1]] + n, ex_1_prior[[2]] + n *
                            mean_wait) + mcoefs[[2]] * pinvgamma(20, ex_2_prior[[1]] + n, ex_2_prior[[2]] + n *
                                                                 mean_wait)

post_wait_sim <-
  mcoefs[[1]] * rinvgamma(1e5, ex_1_prior[[1]] + n, ex_1_prior[[2]] + n *
                            mean_wait) + mcoefs[[2]] * rinvgamma(1e5, ex_2_prior[[1]] + n, ex_2_prior[[2]] + n *
                                                                 mean_wait)

N <- 1e6

components <-
  sample(
    1:2,
    prob = c(mcoefs[[1]], mcoefs[[2]]),
    size = N,
    replace = TRUE
  )
alphas <- c(ex_1_prior[[1]] + n, ex_2_prior[[1]] + n)
betas <-
  c(ex_1_prior[[2]] + n * mean_wait, ex_2_prior[[2]] + n * mean_wait)
samples <- rinvgamma(N, alphas[components], betas[components])

plot(density(samples))
mean(samples)
quantile(samples, c(0.025, 0.975))

post_pred <-
  Renext::rlomax(N, betas[components], alphas[components])

plot(density(post_pred))
mean(post_pred)
quantile(post_pred, c(0.025, 0.975))
sum(post_pred > 20) / N

library(coda)
library(rstan)

rstan_options(auto_write = TRUE)

b_mm_data <-
  list(
    K = 2,
    N = n,
    y = waiting_times,
    theta = c(0.5, 0.5),
    alpha = c(6, 38),
    beta = c(62.5, 277.5)
  )

# b_data <- list(N=n, y = waiting_times, gprior=c(38,278))
#
# b_m <- stan_model(file = 'a1q2simp.stan')
#
# b_m_fit <- sampling(
#   b_m,
#   data = b_data,
#   chains = 7,
#   control = list(adapt_delta = 0.8),
#   iter = 1e5
# )
# plot(b_m_fit)
# b_m_fit
#
# smps <- extract(b_m_fit)
# tp <- traceplot(b_m_fit, pars = c("lambda"))
# tp

# b_m_coda <- As.mcmc.list(b_m_fit)
# gelman.plot(b_m_coda, ask=FALSE)
# gelman.diag(b_m_coda)
# plot(density(smps$postdraw))
# sum(smps$postdraw > 20) / (length(smps$postdraw))
# mean(smps$postdraw)
# quantile(smps$postdraw, c(0.025, 0.975))
# plot(density(smps$ewt))
# sum(smps$ewt > 20) / (length(smps$ewt))
# mean(smps$ewt)
# quantile(smps$ewt, c(0.025, 0.975))

b_mm <- stan_model(file = 'a1q2.stan')

b_mm_fit <- sampling(
  b_mm,
  data = b_mm_data,
  chains = 7,
  control = list(adapt_delta = 0.8),
  iter = 40000
)
plot(b_mm_fit)
b_mm_fit

```

```r
smps <- extract(b_mm_fit)
tp <- traceplot(b_mm_fit, pars = c("lambda", "mwt"))
tp

b_mm_coda <- As.mcmc.list(b_mm_fit)
gelman.plot(b_mm_coda, ask = FALSE)
gelman.diag(b_mm_coda)

plot(density(smps$postdraw))
sum(smps$postdraw > 20) / (length(smps$postdraw))
mean(smps$postdraw)
quantile(smps$postdraw, c(0.025, 0.975))

plot(density(smps$mwt))
sum(smps$mwt > 20) / (length(smps$mwt))
mean(smps$mwt)
quantile(smps$mwt, c(0.025, 0.975))


require(rjags)

model = jags.model(
  file = "a1q2jags.jags",
  data =
    list(
      y = waiting_times,
      n = n,
      a = c(38, 6),
      b = c(277.5, 62.5),
      p = c(0.5, 0.5)
    ),
  n.chains = 10
)

# Burnin for 1000 samples
update(model, 100000, progress.bar = "none")

# Running the model
res = coda.samples(
  model,
  variable.names = c("lambda", "ypred", "texp"),
  n.iter = 200000,
  progress.bar = "none"
)
summary(res)

qqplot(smps$postdraw, waiting_times, xlim = c(0, 40))
qqplot(post_pred, waiting_times, xlim = c(0, 40))
qqplot(res[[1]][, "ypred"], waiting_times, xlim = c(0, 40))


b_mm_lind <- stan_model(file = 'a1q2lindley.stan')

b_mm_lind_fit <- sampling(
  b_mm_lind,
  data = b_mm_data,
  chains = 7,
  control = list(adapt_delta = 0.8),
  iter = 40000
)
plot(b_mm_lind_fit)
b_mm_lind_fit

smps_lind <- extract(b_mm_lind_fit)
tp_lind <- traceplot(b_mm_lind_fit, pars = c("lambda", "mwt"))
tp_lind

b_mm_lind_coda <- As.mcmc.list(b_mm_lind_fit)
gelman.plot(b_mm_lind_coda, ask = FALSE)
gelman.diag(b_mm_lind_coda)

plot(density(smps_lind$postdraw))
sum(smps_lind$postdraw > 20) / (length(smps_lind$postdraw))
mean(smps_lind$postdraw)
quantile(smps_lind$postdraw, c(0.025, 0.975))

plot(density(smps_lind$mwt))
sum(smps_lind$mwt > 20) / (length(smps_lind$mwt))
mean(smps_lind$mwt)
quantile(smps_lind$mwt, c(0.025, 0.975))

qqplot(smps_lind$postdraw, waiting_times, xlim = c(0, 40))
qqplot(smps$postdraw, waiting_times, xlim = c(0, 40))
ks.test(smps_lind$postdraw, waiting_times)
ks.test(smps$postdraw, waiting_times)


dens <- data.frame(postdraw = post_pred)
ggdens <-
  ggplot(data = dens, aes(x = postdraw)) +
  geom_density(size = 1, color = "darkblue", fill = "lightblue") +
  theme_minimal() +
  labs(x = "Posterior Waiting Time", y = "Density of Posterior Predictive", title = "Posterior Predictive Density (Exponential Likelihood)") +
  geom_vline(
    xintercept = mean(dens$postdraw),
    size = 2,
    color = "red"
  ) +
  geom_vline(
    xintercept = quantile(dens$postdraw, c(0.025, 0.975)),
    color = "darkred",
    size = 1
  ) + coord_cartesian(xlim = c(0, 50))
ggdens

lambda <- seq(0, 1, len = 5000)

```

```r
likelihood <-
  as.matrix(apply(as.array(waiting_times), 1, dexp, lambda))
likelihood <- apply(likelihood, 1, prod)

area = sfsmisc::integrate.xy(lambda, likelihood)
const = 1 / area
likelihood <- const * likelihood

prior <-
  p * dgamma(lambda, ex_1_prior[[1]], ex_1_prior[[2]]) + (1 - p) * dgamma(lambda, ex_2_prior[[1]], ex_2_prior[[2]])
posterior <-
  mcoefs[[1]] * dgamma(lambda, ex_1_prior[[1]] + n, ex_1_prior[[2]] + n *
                         mean_wait) + mcoefs[[2]] * dgamma(lambda, ex_2_prior[[1]] + n, ex_2_prior[[2]] + n *
                                                               mean_wait)

prlikpost <-
  reshape2::melt(list(
    Prior = prior,
    Likelihood = likelihood,
    Posterior = posterior
  ))
ggcompplot <-
  ggplot(data = prlikpost) + geom_area(
    aes(
      x = rep(lambda, 3),
      y = value,
      colour = L1,
      fill = L1
    ),
    alpha = 0.3,
    size = 0.5,
    position = "identity"
  ) +
  theme_minimal() +
  labs(x = "Lambda", y = "Density", title = "Prior-Posterior Compatibility Plot")
ggcompplot

mcoefs[[1]] *
  Renext::qlomax(c(0.025, 0.975), betas[1], alphas[1]) +
  mcoefs[[2]] *
  Renext::qlomax(c(0.025, 0.975), betas[2], alphas[2])

dens_lind <- data.frame(postdraw = smps_lind$postdraw)
ggdens_lind <-
  ggplot(data = dens, aes(x = postdraw)) +
  geom_density(size = 1, color = "darkblue", fill = "lightblue") +
  theme_minimal() +
  labs(x = "Posterior Waiting Time", y = "Density of Posterior Predictive", title = "Posterior Predictive Density (Lindley Likelihood)") +
  geom_vline(
    xintercept = mean(dens_lind$postdraw),
    size = 2,
    color = "red"
  ) +
  geom_vline(
    xintercept = quantile(dens_lind$postdraw, c(0.025, 0.975)),
    color = "darkred",
    size = 1
  ) + coord_cartesian(xlim = c(0, 50))
ggdens_lind


quantile_set <- seq(0.01, 0.99, by = 0.01)
data_q <- quantile(waiting_times, quantile_set)
exp_q <- quantile(smps$postdraw, quantile_set)
lind_q <- quantile(smps_lind$postdraw, quantile_set)

quantile_df <- data.frame(obs = data_q, exp = exp_q, lind = lind_q)
ggqq <-
  ggplot(data = quantile_df, aes(y = obs)) + coord_cartesian(xlim = c(0, 45), ylim = c(0, 45)) +
  geom_point(aes(x = exp, colour = "Exponential")) +
  geom_point(aes(x = lind, colour = "Lindley")) +
  theme_minimal() +
  labs(x = "Fitted Quantiles", y = "Sample Quantiles") + geom_abline(slope = 1, intercept = 0) + scale_colour_discrete(name = "Distribution")
ggqq
```

C Stan Code for Question 2

C.1 Exponential Likelihood

```stan
data {          // number of mixture components
  int<lower=1> N;          // number of data points
  real y[N];               // observations
  simplex[2] theta;          // mixing proportions
  positive_ordered[2] alpha; // locations of mixture components
  vector<lower=0>[2] beta;  // scales of mixture components

}
parameters {
  real<lower=0> lambda;
}
model {
  y ~ exponential(lambda);
  target += log_mix(theta[1],
  gamma_lpdf(lambda | alpha[1], beta[1]),
  gamma_lpdf(lambda | alpha[2], beta[2]));
}
generated quantities {
  real<lower=0> mwt = 1 / lambda;
  real postdraw = exponential_rng(lambda);
}
```

C.2 Lindley Likelihood

```
functions{
  real lindley_lpdf(real y, real lambda){
    real lpdf = 2*log(lambda) - log(lambda+1) + log(1+y) - lambda * y;
    return lpdf;
  }
  real lindley_rng(real lambda){
    real u = uniform_rng(0,1);
    real p = lambda/(lambda+1);
    if (u<=p){
      real v = exponential_rng(lambda);
      return v;
    }
    else{
    real w = gamma_rng(2, lambda);
    return w;
    }
  }
}
data {           // number of mixture components
  int<lower=1> N;           // number of data points
  real y[N];                // observations
  simplex[2] theta;           // mixing proportions
  positive_ordered[2] alpha; // locations of mixture components
  vector<lower=0>[2] beta;  // scales of mixture components

}
parameters {
  real<lower=0> lambda;
}
model {
  for (n in 1:N){
    y[n] ~ lindley(lambda);
  }
  target += log_mix(theta[1],
  gamma_lpdf(lambda | alpha[1], beta[1]),
  gamma_lpdf(lambda | alpha[2], beta[2]));
}
generated quantities {
  real<lower=0> mwt =  ((lambda + 2)/(lambda*(lambda + 1)));
  real postdraw = lindley_rng(lambda);
}
```

D JAGS Code for Question 2

```
model{
  for (i in 1:n) {
    y[i] ~ dexp(lambda)
  }
  lambda ~ dgamma(a[pick], b[pick])          # a[1]=9.2, b[1]=13.8;  a[2]=12;  b[2] = 3
  pick ~ dcat(p[1:2])                       # pick takes value 1 or 2 with prior prob p[1] or p[2]
  ypred ~ dexp(lambda)
  texp = 1 / lambda
}
```

E R Code for Question 3

```
library(coda)
library(rstan)
library(rstantools)
library(bayesplot)
library(ggplot2)

library(dplyr)
library(data.table)
library(modeest)

rstan_options(auto_write = TRUE)
options(datatable.fread.datatable = FALSE)
options(mc.cores = parallel::detectCores() - 1)
devAskNewPage(ask = FALSE)
par(ask = F)
#options(mc.cores = 1)

par(mfrow = c(1,1))

abalone <- fread('data/abalone.csv')
names(abalone) <- make.names(names(abalone))
abalone$Age <- abalone$Rings + 1.5
abalone <- subset(abalone, select = -c(Rings))

cmeans <-
  colMeans(Filter(is.numeric, subset(abalone, select = -c(Age))))
amean <- mean(abalone$Age)

```

```r
scale2 <- function(x, na.rm = FALSE)
  (x - mean(x, na.rm = na.rm))
abalone <- mutate_if(abalone, is.numeric, scale2, na.rm = TRUE)
abalone$Sex <- as.factor(abalone$Sex)

X <-
  model.matrix(
    Age ~ Sex + Length + Diameter + Height + Whole.weight + Shucked.weight + Viscera.weight + Shell.weight,
    data = abalone
  )

xlevs <-
  lapply(abalone[, sapply(abalone, is.factor), drop = F], function(j) {
    levels(j)
  })

new_obs <-
  data.frame(
    Sex = 'M',
    Length = 0.515,
    Diameter = 0.400,
    Height = 0.133,
    Whole.weight = 0.531,
    Shucked.weight = 0.231,
    Viscera.weight = 0.122,
    Shell.weight = 0.168
  )

dfcmean <- as.data.frame(t(cmeans))

new_obs[-1] <- new_obs[-1] - dfcmean
mm_new <- model.matrix( ~ ., data = new_obs, xlev = xlevs)

f_lm <- lm(data = abalone, Age ~ .)
summary(f_lm)
pred_f_lm <- predict(f_lm, new_obs)
pred_f_lm
par(mfrow = c(2, 2))
plot(f_lm)
par(mfrow = c(1, 1))
plot(fitted(f_lm), abalone$Age, asp = 1)


b_lm_data <-
  list(
    N = nrow(X),
    K = ncol(X),
    X = X,
    y = abalone$Age,
    N_new = 1,
    x_new = as.array(mm_new),
    p_params = c(1e4, 1e-1, 1e-1, 1/27)
  )

b_lm <- stan_model(file = 'a1q3.stan')

b_lm_fit <- sampling(
  b_lm,
  data = b_lm_data,
  chains = 7,
  control = list(adapt_delta = 0.8),
  iter = 16000
)
plot(b_lm_fit)
b_lm_fit

smps <- extract(b_lm_fit)
# tp <- traceplot(b_lm_fit, pars = c("beta", "sigma"))
# tp

b_lm_coda <- As.mcmc.list(b_lm_fit)
gelman.plot(b_lm_coda, ask=FALSE)
# gelman.diag(b_lm_coda)
# plot(b_lm_coda, ask = FALSE)

eap_beta <- matrix(colMeans(smps$beta), ncol = 1)

b_lm_age <- X %*% eap_beta

n_dens <-
  ggplot(data = data.frame(samp = smps$y_new), aes(x = samp)) + geom_density() + coord_cartesian(xlim = c(-10,10))
n_dens

pred_eap <- mean(smps$y_new)
pred_map <- mlv(smps$y_new, method = "venter")
pred_f_lm
pred_map
pred_eap

available_mcmc(pattern = "_nuts_")

posterior_lm <- as.array(b_lm_fit)
np_lm <- nuts_params(b_lm_fit)
lp_lm <- log_posterior(b_lm_fit)

mcmc_nuts_divergence(np_lm, lp_lm)
mcmc_nuts_acceptance(np_lm, lp_lm)

# pairs(b_lm_fit, pars = c("beta[1]", "beta[2]", "beta[3]"))
# pairs(b_lm_fit, pars = c("beta[4]", "beta[5]", "beta[6]"))
# pairs(b_lm_fit, pars = c("beta[7]", "beta[8]", "beta[9]", "beta[10]"))

color_scheme_set("viridisD")
mcmc_trace(posterior_lm, pars = c("beta[1]", "beta[2]", "beta[3]", "beta[4]", "beta[5]", "beta[6]", "beta[7]", "beta[8]", "beta[9]", "beta[10]", "
      sigma"), np = np_lm, size = 0.05)

# b_lm_data_test <-
#   list(
#     N = nrow(X),
#     K = ncol(X),
#     X = X,
#     y = abalone$Age,
#     N_new = 1,
#     x_new = as.array(mm_new),
#     p_params = c(1e1, 7, 30)
#   )
#
# b_lm_fit_prior_sensitivity <- sampling(
#   b_lm,
#   data = b_lm_data_test,
#   chains = 7,
#   control = list(adapt_delta = 0.8),
#   iter = 3000
# )
#
# b_lm_fit_prior_sensitivity
#
# posterior_lm_psa <- as.array(b_lm_fit_prior_sensitivity)
# np_lm_psa <- nuts_params(b_lm_fit_prior_sensitivity)
# lp_lm_psa <- log_posterior(b_lm_fit_prior_sensitivity)
#
# mcmc_nuts_divergence(np_lm_psa, lp_lm_psa)
# mcmc_nuts_acceptance(np_lm_psa, lp_lm_psa)
#
# pairs(b_lm_fit_prior_sensitivity, pars = c("beta[1]", "beta[2]", "beta[3]"))
# pairs(b_lm_fit_prior_sensitivity, pars = c("beta[4]", "beta[5]", "beta[6]"))
# pairs(b_lm_fit_prior_sensitivity, pars = c("beta[7]", "beta[8]", "beta[9]", "beta[10]"))
#
# color_scheme_set("viridisD")
# mcmc_trace(posterior_lm_psa, pars = c("beta[1]", "beta[2]", "beta[3]", "beta[4]", "beta[5]", "beta[6]", "beta[7]", "beta[8]", "beta[9]", "beta
    [10]", "sigma"), np = np_lm_psa, size = 0.05)


quantile_set <- seq(0.01, 0.99, by = 0.01)
data_q <- quantile(abalone$Age, quantile_set)
bayes_q <- quantile(b_lm_age, quantile_set)
lm_q <- quantile(fitted(f_lm), quantile_set)

quantile_df <- data.frame(obs = data_q, b = bayes_q, l = lm_q)
ggqq <- ggplot(data = quantile_df, aes(y = obs)) + coord_cartesian(xlim = c(-10, 10), ylim = c(-10, 10)) +
  geom_point(aes(x = b, colour = "Bayesian Robust Linear Model")) +
  geom_point(aes(x = l, colour = "Normal Linear Model")) +
  theme_minimal() +
  labs(x = "Fitted Quantiles", y = "Sample Quantiles") + geom_abline(slope = 1, intercept = 0) + scale_colour_discrete(name = "Distribution")
ggqq

#####

# b_lm_normal <- stan_model(file = 'a1q3normal.stan')
#
# b_lm_fit_normal <- sampling(
#   b_lm_normal,
#   data = b_lm_data,
#   chains = 7,
#   control = list(adapt_delta = 0.8),
#   iter = 4000
# )
# plot(b_lm_fit_normal)
# b_lm_fit_normal
#
# smps_norm <- extract(b_lm_fit_normal)
#
# eap_beta_norm <- matrix(colMeans(smps_norm$beta), ncol = 1)
#
# b_lm_age_norm <- X %*% eap_beta_norm
#
# n_dens_norm <-
#   ggplot(data = data.frame(samp = smps_norm$y_new), aes(x = samp)) + geom_density() + coord_cartesian(xlim = c(-10,10))
# n_dens_norm
#
# quantile_set <- seq(0.01, 0.99, by = 0.01)
# data_q <- quantile(abalone$Age, quantile_set)
# bayes_q <- quantile(b_lm_age, quantile_set)
# lm_q <- quantile(fitted(f_lm), quantile_set)
# bayes_n_q <- quantile(b_lm_age_norm, quantile_set)
#
# quantile_df <- data.frame(obs = data_q, b = bayes_q, l = lm_q, bn = bayes_n_q)
# ggqq <- ggplot(data = quantile_df, aes(y = obs)) + coord_cartesian(xlim = c(-10, 10), ylim = c(-10, 10)) +
#   geom_point(aes(x = b, colour = "Bayesian Robust Linear Model")) +
#   geom_point(aes(x = l, colour = "Normal Linear Model")) +
#   geom_point(aes(x = bn, colour = "Bayesian Normal Linear Model")) +
#   theme_minimal() +
#   labs(x = "Fitted Quantiles", y = "Sample Quantiles") + geom_abline(slope = 1, intercept = 0) + scale_colour_discrete(name = "Distribution")
# ggqq

n_dens <-
  ggplot(data = data.frame(samp = smps$y_new), aes(x = samp)) +
  geom_density(size = 1, color = "darkblue", fill = "lightblue", alpha = 0.3) +
  coord_cartesian(xlim = c(-10,10)) +
  theme_minimal() +
  labs(title = "Posterior Predictive for New Data", x = "Mean-Corrected Age", y = "Density")
n_dens
```

F Stan Code for Question 3

F.1 Linear Model with t Errors

```
data {
  int<lower=1> K;
  int<lower=0> N;
  matrix[N, K] X;
  vector[N] y;
  int<lower=0> N_new;
  matrix[N_new, K] x_new;
  vector[4] p_params;
}
parameters {
  vector[K] beta;
  real<lower=0> sigma;
  real<lower=0> nu;
}
model {
  nu ~ exponential(p_params[4]);
  beta ~ normal(0, p_params[1]);
  sigma ~ inv_gamma(p_params[2], p_params[3]);
  y ~ student_t(nu+1, X * beta, sigma);
}
generated quantities {
  vector[N_new] y_new;
  for (n in 1:N_new)
    y_new[n] = student_t_rng(nu+1, x_new[n] * beta, sigma);
}
```

F.2 Linear Model with Normal Errors

```
data {
  int<lower=1> K;
  int<lower=0> N;
  matrix[N, K] X;
  vector[N] y;
  int<lower=0> N_new;
  matrix[N_new, K] x_new;
  vector[3] p_params;
}
parameters {
  vector[K] beta;
  real<lower=0> sigma;
}
model {
  beta ~ normal(0, p_params[1]);
  sigma ~ inv_gamma(p_params[2], p_params[3]);
  y ~ normal(X * beta, sigma);
}
generated quantities {
  vector[N_new] y_new;
  for (n in 1:N_new)
    y_new[n] = normal_rng(x_new[n] * beta, sigma);
}
```