STA2201 Assignment 2 Solutions Outline

```r
library(tidyverse)
library(rstan)
library(tidybayes)
library(here)
library(loo)
library(kableExtra)
```

Question 1: IQ

Part a

To find the posterior distribution of μ, notice that it is proportional to the following quantity:

$$P(\mu|Y_1, Y_2, \ldots, Y_n, \sigma^2) \propto P(Y_1, Y_2, \ldots, Y_n|\mu, \sigma^2)P(\mu|\sigma^2) = P(Y_1, Y_2, \ldots, Y_n|\mu, \sigma^2)P(\mu|\mu_0, \sigma^2_{\mu_0})$$

The normalizing constant is not needed to identify the distribution form of the posterior distribution. Notice that we know that $\mu_0 = 100$, and $\sigma = \sigma_{\mu_0} = 15$ and can substitute these values in to simplify. The right hand side of the above can then be expressed as follows:

$$
\begin{aligned}
P(Y_1, Y_2, \ldots, Y_n|\mu, \sigma^2)P(\mu|\mu_0, \sigma^2_{\mu_0}) &= \left[\prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\left(\frac{y_i-\mu}{\sigma}\right)^2}\right]\left[\frac{1}{\sqrt{2\pi}\sigma_{\mu_0}} e^{-\frac{1}{2}\left(\frac{\mu-\mu_0}{\sigma_{\mu_0}}\right)^2}\right] \\
&= \left[\prod_{i=1}^{n} \frac{1}{15\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{y_i-\mu}{15}\right)^2}\right]\left[\frac{1}{15\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{\mu-\mu_0}{15}\right)^2}\right] \\
&= \left[\left(\frac{1}{15\sqrt{2\pi}}\right)^n e^{\sum_{i=1}^{n} -\frac{1}{2}\left(\frac{y_i-\mu}{15}\right)^2}\right]\left[\frac{1}{15\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{\mu-\mu_0}{15}\right)^2}\right] \\
&= \left(\frac{1}{15\sqrt{2\pi}}\right)^{n+1} e^{\sum_{i=1}^{n}\left(-\frac{1}{2}\left(\frac{y_i-\mu}{15}\right)^2\right)-\frac{1}{2}\left(\frac{\mu-\mu_0}{15}\right)^2} \\
&\propto e^{-\frac{1}{2\cdot 15^2}\left(\sum_{i=1}^{n}\left(y_i^2-2y_i\mu+\mu^2\right)+\left(\mu^2-2\mu\mu_0+\mu_0^2\right)\right)} \\
&= e^{-\frac{1}{2\cdot 15^2}\left(\sum_{i=1}^{n} y_i^2-2\mu\sum_{i=1}^{n} y_i+n\mu^2+\mu^2-2\mu\mu_0+\mu_0^2\right)} \\
&= e^{-\frac{1}{2\cdot 15^2}\left((n+1)\mu^2-2\left(\sum_{i=1}^{n} y_i+\mu_0\right)\mu+\sum_{i=1}^{n} y_i^2+\mu_0^2\right)} \\
&= e^{\left(-\frac{n+1}{2\cdot 15^2}\left(\mu^2-2\frac{\left(\sum_{i=1}^{n} y_i+\mu_0\right)}{n+1}\mu+\frac{\sum_{i=1}^{n} y_i^2+\mu_0^2}{n+1}\right)\right)} \\
&= e^{\left(-\frac{n+1}{2\cdot 15^2}\left(\mu^2-2\frac{\left(\sum_{i=1}^{n} y_i+\mu_0\right)}{n+1}\mu+\left(\frac{\left(\sum_{i=1}^{n} y_i+\mu_0\right)}{n+1}\right)^2-\left(\frac{\left(\sum_{i=1}^{n} y_i+\mu_0\right)}{n+1}\right)^2+\frac{\sum_{i=1}^{n} y_i^2+\mu_0^2}{n+1}\right)\right)} \\
&= e^{\left(-\frac{n+1}{2\cdot 15^2}\left(-\left(\frac{\left(\sum_{i=1}^{n} y_i+\mu_0\right)}{n+1}\right)^2+\frac{\sum_{i=1}^{n} y_i^2+\mu_0^2}{n+1}\right)\right)} e^{\left(-\frac{n+1}{2\cdot 15^2}\left(\mu-\frac{\left(\sum_{i=1}^{n} y_i+\mu_0\right)}{n+1}\right)^2\right)} \\
&\propto e^{\left(-\frac{n+1}{2\cdot 15^2}\left(\mu-\frac{\left(\sum_{i=1}^{n} y_i+\mu_0\right)}{n+1}\right)^2\right)} \\
&= e^{\left(-\frac{1}{2}\left(\frac{\mu-\frac{\mu_0+\sum_{i=1}^{n} y_i}{n+1}}{\frac{15^2}{n+1}}\right)^2\right)}
\end{aligned}
$$

The last line implies that the posterior distribution is normally distribution with parameters $\mu|Y_1, Y_2, \ldots, Y_n, \sigma^2 \sim N\left(\frac{\mu_0+\sum_{i=1}^{n} y_i}{n+1}, \frac{15^2}{n+1}\right)$. As $\mu_0 = 100$, $n = 10$, and $\bar{y} = 113$, $\frac{\mu_0+\sum_{i=1}^{n} y_i}{n+1} = \frac{100+10*113}{11} = \frac{1230}{11}$ and $\frac{15^2}{n+1} = \frac{225}{11}$. So, $\mu|Y_1, Y_2, \ldots, Y_n, \sigma^2 \sim N\left(\frac{1230}{11}, \frac{225}{11}\right)$.

The Bayesian point estimate is $E[\mu|y] = \hat{\mu}_{Bayes} = \frac{1230}{11} \approx 111.8182$. A 95% credible interval for μ can then be found by using the fact that 95% of the density of a normal distribution is within 1.96 standard deviations of the mean. So, our interval is then $CI(\mu) = \left[\frac{1230}{11} - 1.96\frac{225}{11}, \frac{1230}{11} + 1.96\frac{225}{11}\right] \approx [71.7272, 151.9091]$.

Part b

To show the mean squared error (MSE) decomposition, introduce the expected value of $\hat{\mu}$ as follows:

$$
\begin{aligned}
\text{MSE}[\hat{\mu}|\mu^*] &= E\left[(\hat{\mu}-\mu^*)^2|\mu^*\right] \\
&= E\left[(\hat{\mu}-E[\hat{\mu}]+E[\hat{\mu}]-\mu^*)^2|\mu^*\right] \\
&= E\left[(\hat{\mu}-E[\hat{\mu}])^2+2(\hat{\mu}-E[\hat{\mu}])(E[\hat{\mu}]-\mu^*)+(E[\hat{\mu}]-\mu^*)^2|\mu^*\right] \\
&= E\left[(\hat{\mu}-E[\hat{\mu}])^2|\mu^*\right]+E\left[2(\hat{\mu}-E[\hat{\mu}])(E[\hat{\mu}]-\mu^*)|\mu^*\right]+E\left[(E[\hat{\mu}]-\mu^*)^2|\mu^*\right] \\
&= \text{Var}(\hat{\mu}|\mu^*)+2(E[\hat{\mu}]-\mu^*)E\left[(\hat{\mu}-E[\hat{\mu}])|\mu^*\right]+(E[\hat{\mu}]-\mu^*)^2 \\
&= \text{Var}(\hat{\mu}|\mu^*)+2(E[\hat{\mu}]-\mu^*)(E[\hat{\mu}]-E[\hat{\mu}])+\text{Bias}(\hat{\mu}|\mu^*)^2 \\
&= \text{Var}(\hat{\mu}|\mu^*)+\text{Bias}(\hat{\mu}|\mu^*)^2
\end{aligned}
$$

So, $\text{MSE}[\hat{\mu}|\mu^*] = \text{Var}(\hat{\mu}|\mu^*) + \text{Bias}(\hat{\mu}|\mu^*)^2$ as required.

Part c

Recall that the ML estimator is $\hat{\mu}_{MLE} = \frac{1}{n}\sum_{i=1}^n y_i$. Then,

$$E[\hat{\mu}_{MLE}|\mu^*] = E\left[\frac{1}{n}\sum_{i=1}^n y_i\right] = \frac{1}{n}\sum_{i=1}^n E[y_i] = \frac{1}{n}\sum_{i=1}^n \mu^* = \mu^* = 112$$
$$\text{Var}(\hat{\mu}_{MLE}|\mu^*) = \text{Var}\left(\frac{1}{n}\sum_{i=1}^n y_i\right) = \frac{1}{n^2}\sum_{i=1}^n \text{Var}(y_i) = \frac{n\sigma^2}{n^2} = \frac{\sigma^2}{n} = \frac{15^2}{10} = 22.5$$

and,

$$\text{Bias}(\hat{\mu}_{MLE}|\mu^*) = 112 - 112 = 0$$
$$\text{Var}(\hat{\mu}_{MLE}|\mu^*) = 22.5$$
$$\text{MSE}(\hat{\mu}_{MLE}|\mu^*) = 22.5 + 0^2 = 22.5$$

For the Bayes estimator,

$$E[\hat{\mu}_{Bayes}|\mu^*] = E\left[\frac{\mu_0 + \sum_{i=1}^n y_i}{n+1}\right] = \frac{1}{n+1}\left(\mu_0 + \sum_{i=1}^n E[y_i]\right) = \frac{\mu_0 + n\mu^*}{n+1} = \frac{100 + 10 * 112}{11} = \frac{1220}{11}$$
$$\text{Var}(\hat{\mu}_{Bayes}|\mu^*) = \text{Var}\left(\frac{\mu_0 + \sum_{i=1}^n y_i}{n+1}\right) = \frac{1}{(n+1)^2}\sum_{i=1}^n \text{Var}(y_i) = \frac{n\sigma^2}{(n+1)^2} = \frac{10 * 15^2}{11^2} = \frac{2250}{121}$$

$$\text{Bias}(\hat{\mu}_{Bayes}|\mu^*) = \frac{1220}{11} - 112 = -\frac{12}{11} \approx -1.0909$$
$$\text{Var}(\hat{\mu}_{Bayes}|\mu^*) = \frac{2250}{121} \approx 18.595$$
$$\text{MSE}(\hat{\mu}_{Bayes}|\mu^*) = 18.595 + (-1.0909)^2 = 19.7851$$

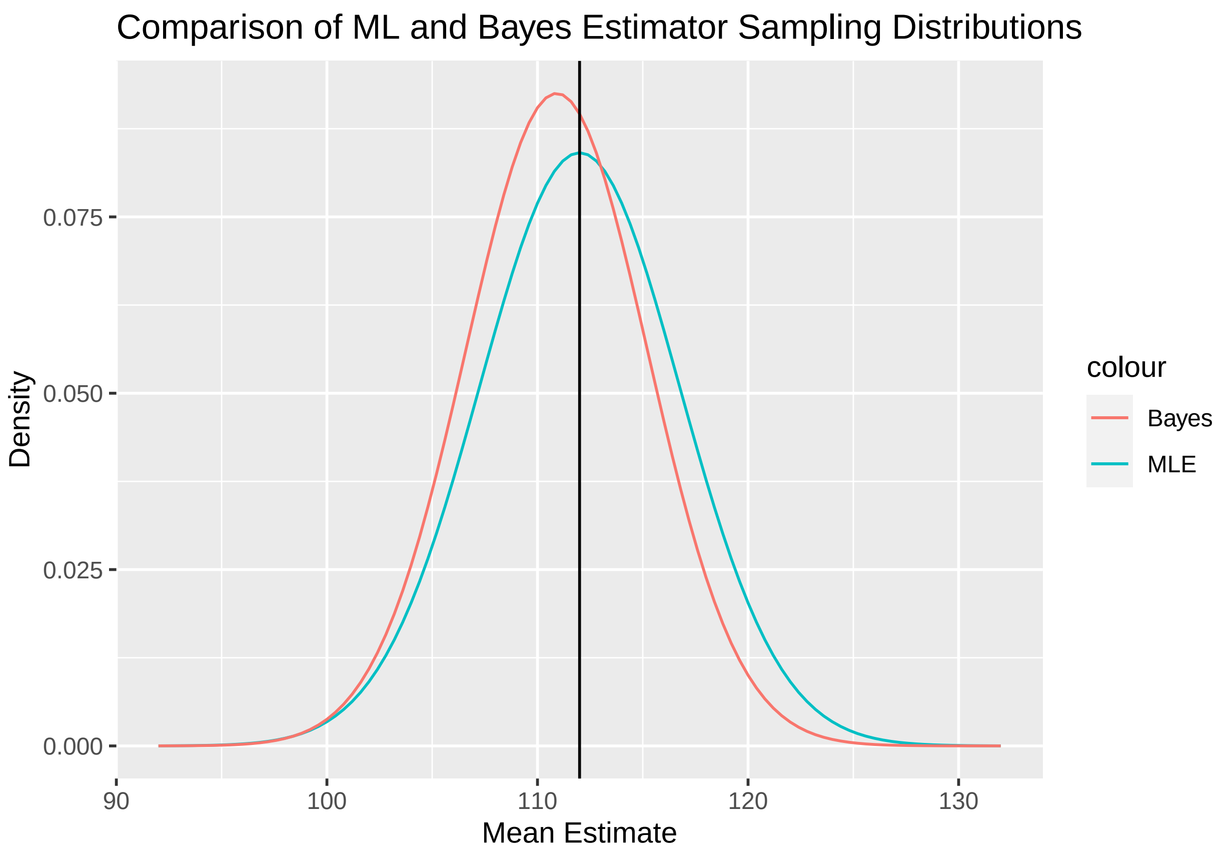
So, we see that while the Bayes estimator has a higher bias, the ML estimator has a higher MSE.

Part d

Both the ML and Bayes estimators have normal sampling distributions. For the ML estimator, $\hat{\mu}_{MLE} \sim N\left(\mu^*, \frac{\sigma^2}{n}\right) = N(112, 22.5)$ and for the Bayes estimator, $\hat{\mu}_{Bayes} \sim N\left(\frac{\mu_0 + n\mu^*}{n+1}, \frac{n\sigma^2}{(n+1)^2}\right) = N\left(\frac{1220}{11}, \frac{2250}{121}\right)$.

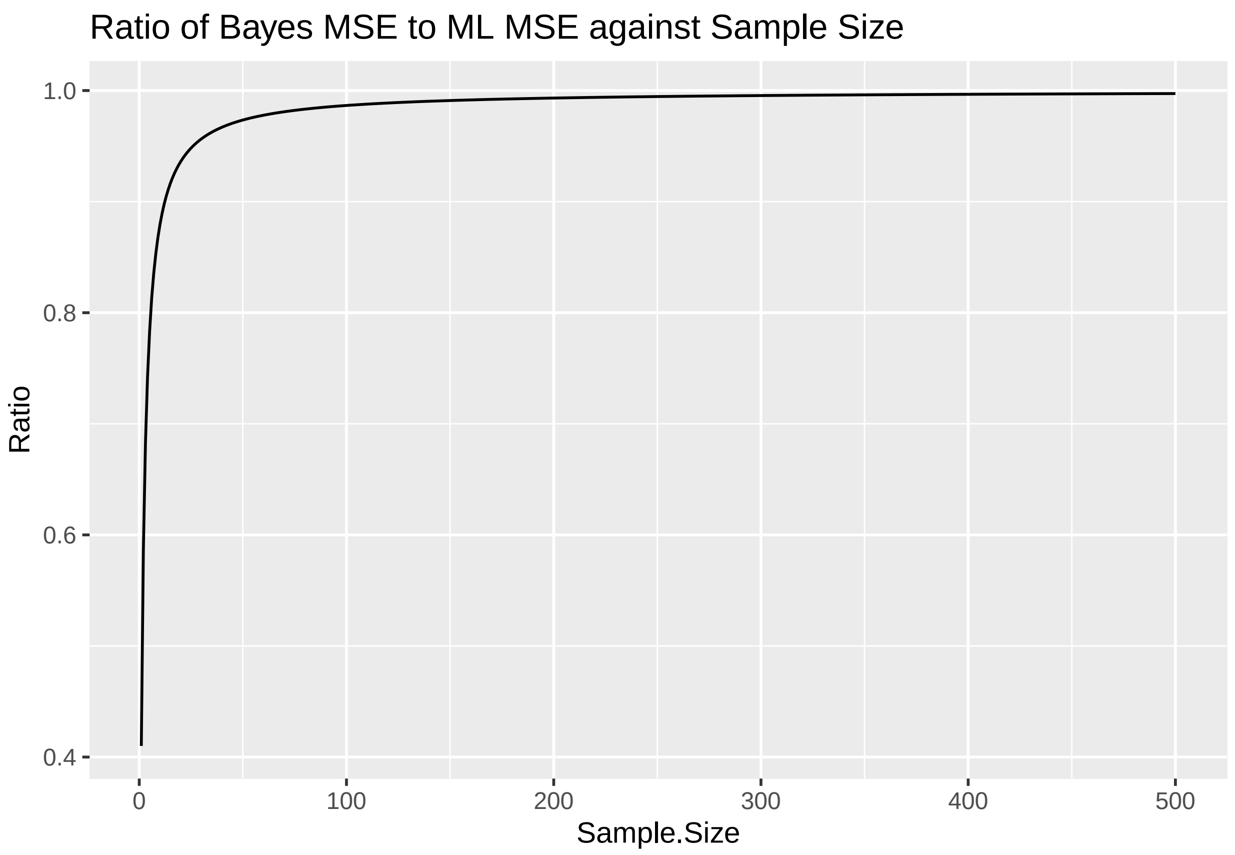
The two distributions are then,

```
ggplot(data=data.frame(x=c(92,132)), aes(x=x)) +
  stat_function(fun=dnorm, args=list(mean=112, sd=(15/sqrt(10))),
                aes(colour="MLE")) +
  stat_function(fun=dnorm, args=list(mean=(1220/11), sd=sqrt(2250/121)),
                aes(colour="Bayes")) +
  geom_vline(xintercept=112) +
  xlab("Mean Estimate") +
  ylab("Density") +
  ggtitle("Comparison of ML and Bayes Estimator Sampling Distributions")
```



In order to aide with interpretation, the black line on the above plot is the true value of $\mu^* = 112$. As expected, the ML estimator is unbiased and centered at this value. Notice that the Bayes estimator is not centered there, reflecting its bias. It is worth noting that it is shifted left, closer to the μ_0 hyperparameter. While the ML estimator performs better with respect to the bias, notice that it is not as sharply peaked as the Bayes estimator, indicating that it has a higher variance than the Bayes estimator. As the bias of the Bayes estimator is visibly quite small, the Bayes MSE is smaller than that of the ML estimator as the larger variance outpaces the lack of bias.

```
k <- 500
n <- c(1:k)

MLbias <- rep(0,k)
MLvar <- (15^2)/n
MLMSE <- MLvar + MLbias^2

Bayesbias <- (100+n*112)/(n+1) - rep(112,k)
Bayesvar <- (n*15^2)/((n+1)^2)
BayesMSE <- Bayesvar + Bayesbias^2

d <- data.frame(Sample.Size = n, Ratio=BayesMSE/MLMSE)

d %>% ggplot(aes(x=Sample.Size, y=Ratio)) +
  geom_line() +
  ggtitle("Ratio of Bayes MSE to ML MSE against Sample Size")
```



The above plot shows the ratio between the Bayes MSE and the ML MSE against sample size. Notice that the ratio is strictly less than or equal to 1 reflecting that the Bayes estimator always has a smaller mean squared error. However, after a sample size of 200 or 250, the difference is negligible. Before 50, the difference is quite dramatic with reflects the well known issue that ML estimators struggle in low data environments.

Gompertz

Gompertz hazards are of the form

$$\mu_x = \alpha e^{\beta x}$$

for $x \in [0, \infty)$ with $\alpha, \beta > 0$. It is named after Benjamin Gompertz, who suggested a similar form to capture a 'law of human mortality' in 1825.

This question uses data on deaths by age in Sweden over time. The data are in the `sweden` file in the class repo. I grabbed the data from the Human Mortality Database.

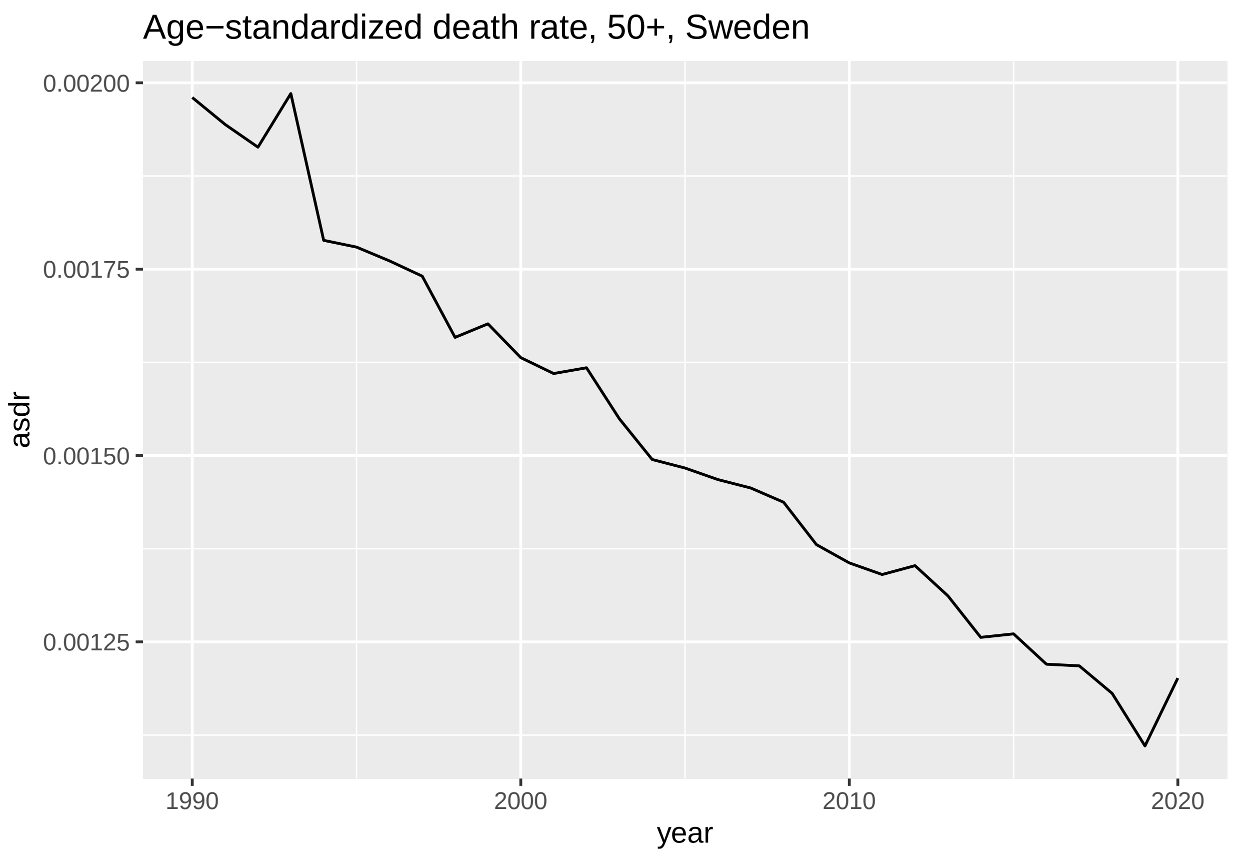
We will assume that the deaths we observe in a particular age group are Poisson distributed with a rate equal to the mortality rate multiplied by the population, i.e.

$$D_x \sim \text{Poisson}(\mu_x P_x)$$

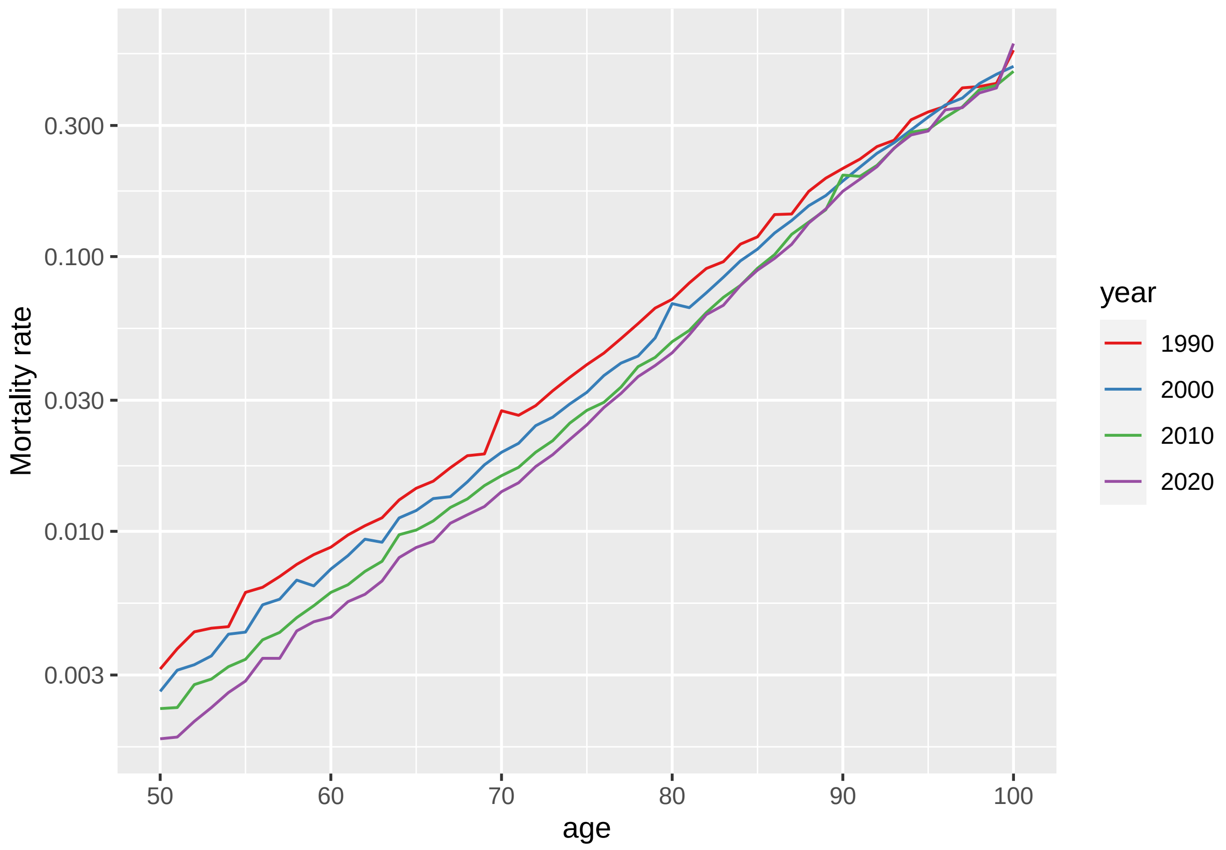
where x refers to age. In this question we will be estimating mortality rates using the Gompertz model as described above.

a) Describe, with the aid of a couple of graphs, some key observations of how mortality above age 50 in Sweden has changed over time.

```
df_red <- read_csv(here("data/sweden.csv"))

df_red %>%
  group_by(year) %>%
  summarise(asdr = sum(deaths/pop)/sum(pop)*1000) %>%
  ggplot(aes(year, asdr)) + geom_line() +
  labs(title = "Age-standardized death rate, 50+, Sweden")
```



```r
df_red %>%
  filter(year %in% seq(1990, 2020, by = 10)) %>%
  mutate(mortality_rate = deaths/pop) %>%
  ggplot(aes(age, mortality_rate, color = factor(year))) +
  geom_line() +
  scale_y_log10() +
  labs(y = "Mortality rate") +
  scale_color_brewer(palette = "Set1", name = "year")
```



b) Carry out prior predictive checks for α and β, based on populations by age in Sweden in 2020. Summarize what you find and what you decide to be weakly informative priors for these parameters.

c) Fit a model in Stan to estimate α and β for the year 2020. Note that it may be easier to specify the likelihood on the log scale (you can do this in Stan using the `poisson_log` function). Priors should be informed by your prior predictive checks and any other information available. Ensure that the model has converged and other diagnostics are good. Interpret your estimates for α and β.

```
df_red_2020 <- df_red %>% filter(year==2020)

data <- list(y = round(df_red_2020$deaths),
             log_pop = log(df_red_2020$pop),
             N = nrow(df_red_2020),
             x = df_red_2020$age)

mod <- stan(data = data, file = here("assignments/code/models/", "gomp.stan"))
```

```
## Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
## clang -I"/Library/Frameworks/R.framework/Resources/include" -DNDEBUG   -I"/Library/Frameworks/R.frame
## In file included from <built-in>:1:
## In file included from /Library/Frameworks/R.framework/Versions/4.0/Resources/library/StanHeaders/incl
## In file included from /Library/Frameworks/R.framework/Versions/4.0/Resources/library/RcppEigen/inclu
## In file included from /Library/Frameworks/R.framework/Versions/4.0/Resources/library/RcppEigen/inclu
## /Library/Frameworks/R.framework/Versions/4.0/Resources/library/RcppEigen/include/Eigen/src/Core/util/
## namespace Eigen {
## ^
## /Library/Frameworks/R.framework/Versions/4.0/Resources/library/RcppEigen/include/Eigen/src/Core/util/
## namespace Eigen {
##                ^
##                ;
## In file included from <built-in>:1:
## In file included from /Library/Frameworks/R.framework/Versions/4.0/Resources/library/StanHeaders/incl
## In file included from /Library/Frameworks/R.framework/Versions/4.0/Resources/library/RcppEigen/inclu
## /Library/Frameworks/R.framework/Versions/4.0/Resources/library/RcppEigen/include/Eigen/Core:96:10: fa
## #include <complex>
##          ^~~~~~~~~
## 3 errors generated.
## make: *** [foo.o] Error 1
##
## SAMPLING FOR MODEL 'gomp' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 3.2e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.32 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.086626 seconds (Warm-up)
## Chain 1:                0.115263 seconds (Sampling)
## Chain 1:                0.201889 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'gomp' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 8e-06 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.08 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
```

```
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.100852 seconds (Warm-up)
## Chain 2:                0.358005 seconds (Sampling)
## Chain 2:                0.458857 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'gomp' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 1.1e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.11 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.108517 seconds (Warm-up)
## Chain 3:                0.368214 seconds (Sampling)
## Chain 3:                0.476731 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'gomp' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 1e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.1 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
```

```
## Chain 4:  Elapsed Time: 0.109953 seconds (Warm-up)
## Chain 4:                0.414144 seconds (Sampling)
## Chain 4:                0.524097 seconds (Total)
## Chain 4:
```

```
mod
```

```
## Inference for Stan model: gomp.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##                 mean se_mean   sd      2.5%       25%       50%       75%
## log_alpha     -12.60    0.00 0.03    -12.65    -12.61    -12.60    -12.58
## beta            0.12    0.00 0.00      0.12      0.12      0.12      0.12
## lp__       636601.67    0.04 1.01 636599.08 636601.27 636601.97 636602.40
##                97.5% n_eff Rhat
## log_alpha     -12.54   448 1.01
## beta            0.12   453 1.01
## lp__       636602.66   557 1.01
##
## Samples were drawn using NUTS(diag_e) at Tue Mar 22 09:36:21 2022.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

d) Carry out some posterior predictive checks to assess model performance.

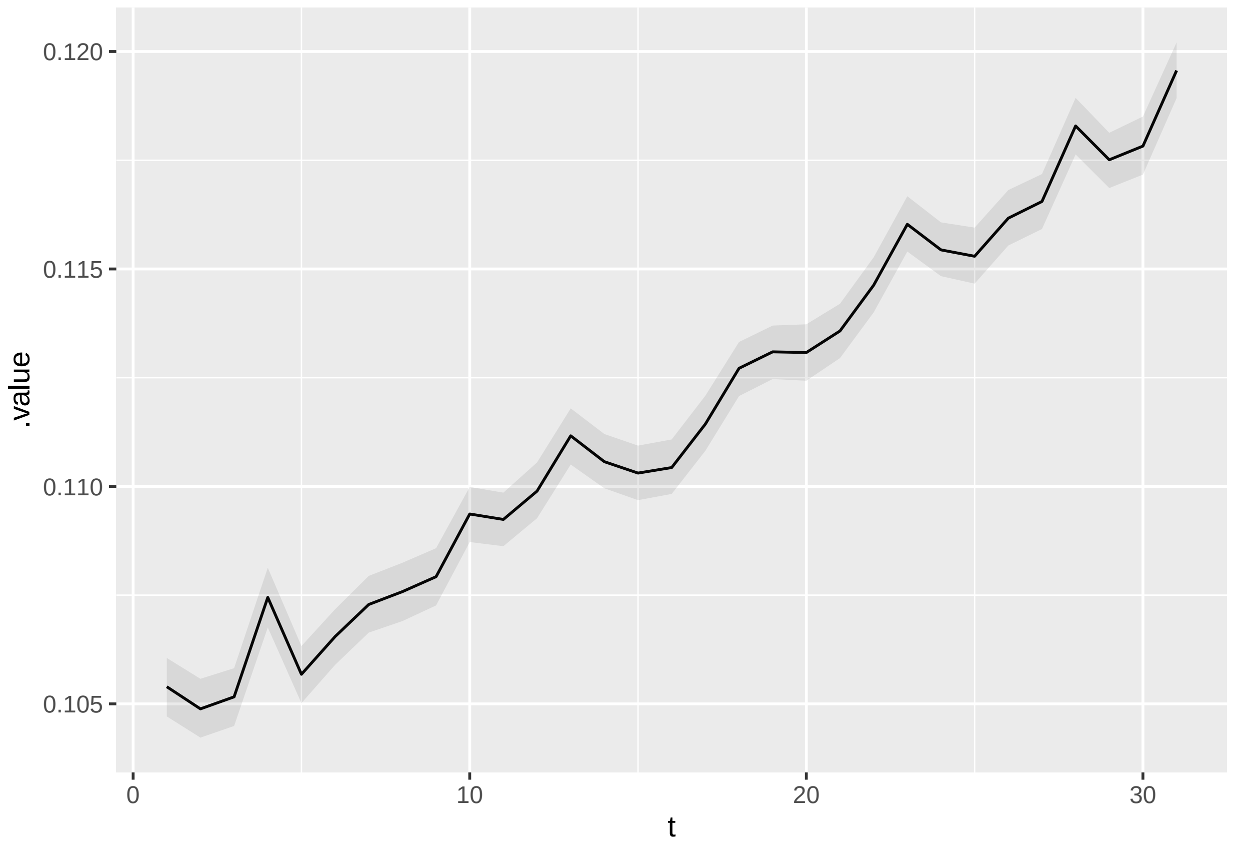
e) Now extend your model to estimate α and β in every year over the interval 1990-2020. Plot the resulting point estimates and 95% credible intervals for your estimates of α and β over time. Comment briefly on what you observe.

```
y <- df_red %>%
  select(age, year, deaths) %>%
  pivot_wider(names_from = "year", values_from = "deaths") %>%
  select(-age) %>%
  as.matrix()

pop <- df_red %>%
  select(age, year, pop) %>%
  pivot_wider(names_from = "year", values_from = "pop") %>%
  select(-age) %>%
  as.matrix()

data2 <- list(y = round(y),
             log_pop = log(pop),
             N = nrow(y),
             x = df_red_2020$age - 75,
             T = ncol(y))

mod <- stan(data = data2,
            file = here("assignments/code/models/", "gomp4.stan"))
```

```
## Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
## clang -I"/Library/Frameworks/R.framework/Resources/include" -DNDEBUG   -I"/Library/Frameworks/R.frame
## In file included from <built-in>:1:
## In file included from /Library/Frameworks/R.framework/Versions/4.0/Resources/library/StanHeaders/incl
## In file included from /Library/Frameworks/R.framework/Versions/4.0/Resources/library/RcppEigen/inclu
## In file included from /Library/Frameworks/R.framework/Versions/4.0/Resources/library/RcppEigen/inclu
## /Library/Frameworks/R.framework/Versions/4.0/Resources/library/RcppEigen/include/Eigen/src/Core/util/
## namespace Eigen {
## ^
## /Library/Frameworks/R.framework/Versions/4.0/Resources/library/RcppEigen/include/Eigen/src/Core/util/
## namespace Eigen {
##                ^
##                ;
## In file included from <built-in>:1:
## In file included from /Library/Frameworks/R.framework/Versions/4.0/Resources/library/StanHeaders/incl
## In file included from /Library/Frameworks/R.framework/Versions/4.0/Resources/library/RcppEigen/inclu
## /Library/Frameworks/R.framework/Versions/4.0/Resources/library/RcppEigen/include/Eigen/Core:96:10: f
## #include <complex>
##          ^~~~~~~~~
## 3 errors generated.
## make: *** [foo.o] Error 1
##
## SAMPLING FOR MODEL 'gomp4' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.000294 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 2.94 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 4.50362 seconds (Warm-up)
## Chain 1:                2.64209 seconds (Sampling)
## Chain 1:                7.1457 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'gomp4' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0.000342 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 3.42 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
```

```
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 4.19564 seconds (Warm-up)
## Chain 2:                2.27599 seconds (Sampling)
## Chain 2:                6.47162 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'gomp4' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 0.000214 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 2.14 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 5.1043 seconds (Warm-up)
## Chain 3:                2.23911 seconds (Sampling)
## Chain 3:                7.3434 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'gomp4' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 0.000161 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 1.61 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 3.20277 seconds (Warm-up)
## Chain 4:                1.87759 seconds (Sampling)
## Chain 4:                5.08036 seconds (Total)
## Chain 4:
```

```
mod %>%
  gather_draws(log_alpha[t], beta[t]) %>%
  median_qi() %>%
  filter(.variable=="beta") %>%
  ggplot(aes(t, .value)) +
  geom_line()+
  geom_ribbon(aes(ymin = .lower, ymax = .upper), alpha = 0.1)
```

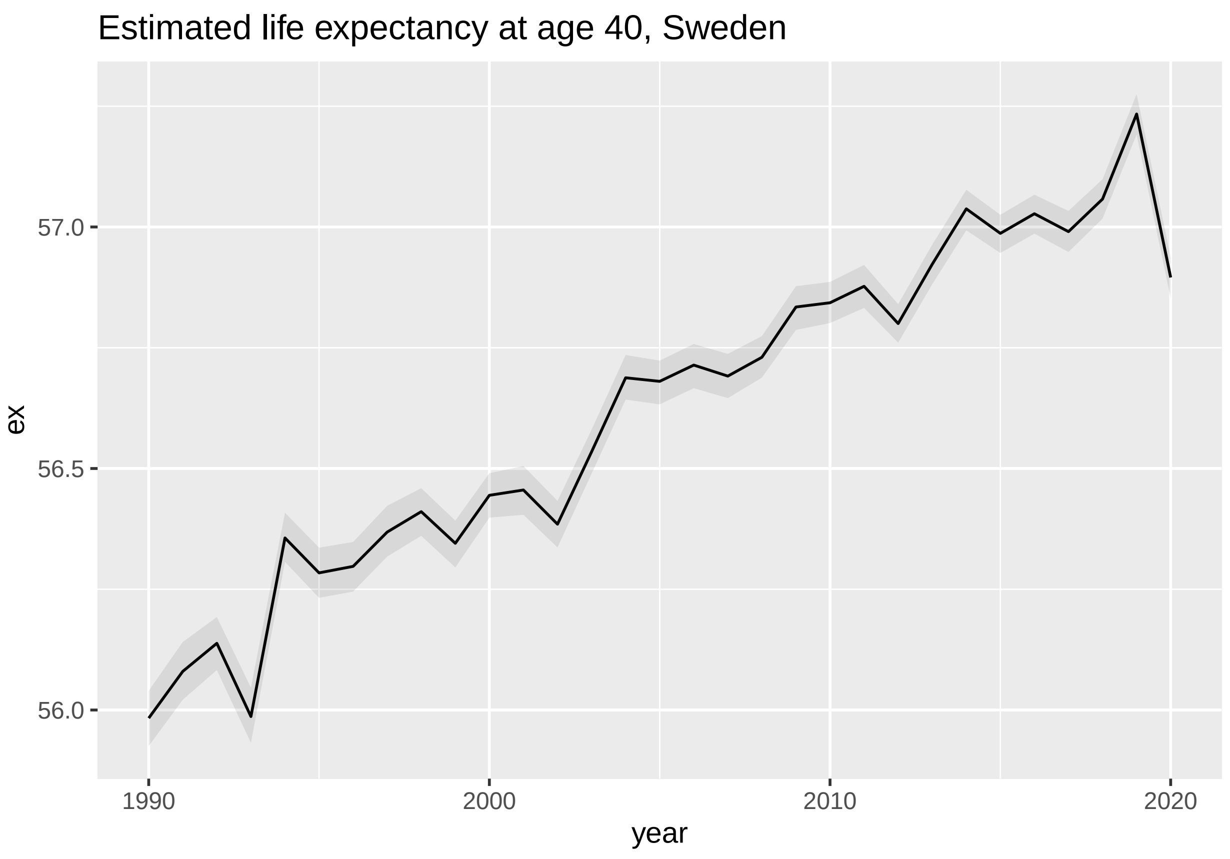


f) Life expectancy at age x is defined as

$$\int_x^\omega e^{-\mu_x} dx$$

where ω is the oldest age group (you may assume this is age 100). Life expectancy is the expected number of years of life left at age x. The integral can be approximated by summing over discrete age groups. Based on your estimates in the previous question, estimate life expectancy at age 40 (note starting age!) for every year from 1990-2020. Plot your resulting point estimates and 95% credible intervals over time and comment briefly.

```
tibble(age = rep(40:100, times = ncol(y)),
       t = rep(1:ncol(y), each = length(40:100))) %>%
  inner_join(mod %>%
               spread_draws(log_alpha[t], beta[t])) %>%
  mutate(log_mu = log_alpha+ beta*(age-75)) %>%
  mutate(lx = exp(-exp(log_mu))) %>%
  group_by(t, .chain, .iteration) %>%
  summarize(ex = sum(lx)) %>%
  group_by(t) %>%
  median_qi(.width = 0.99) %>%
  mutate(year = 1990:2020) %>%
  ggplot(aes(year, ex)) + geom_line() + geom_ribbon(aes(ymin = .lower, ymax = .upper), alpha = 0.1) +
  labs(title = "Estimated life expectancy at age 40, Sweden")
```



Wells

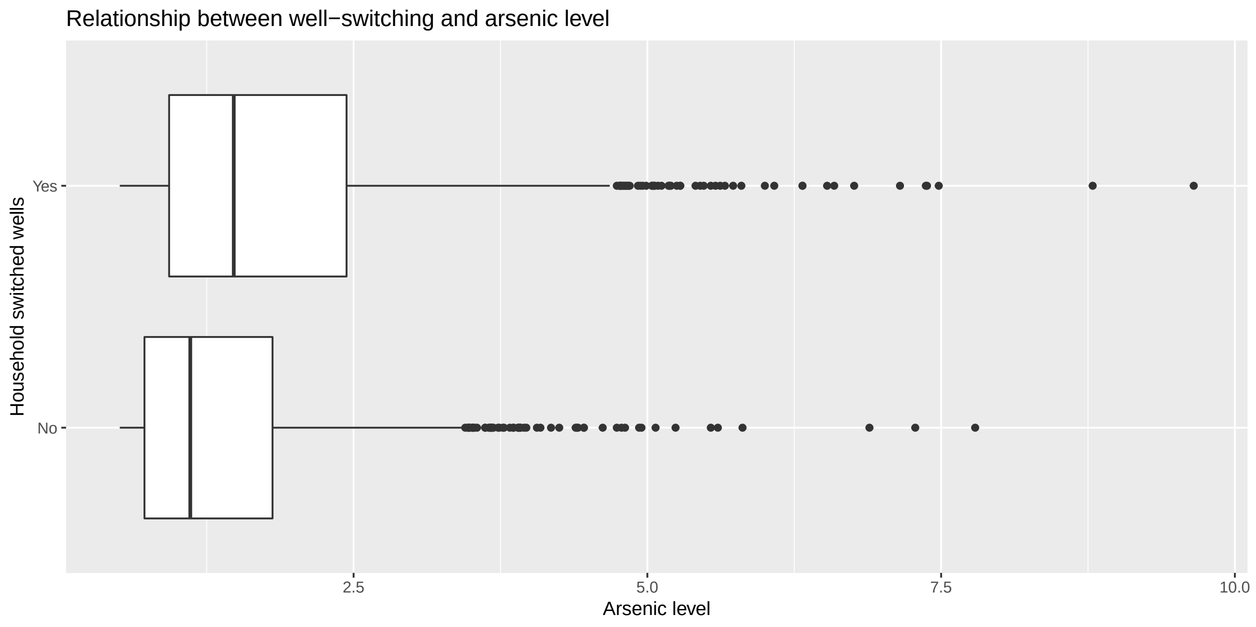
This question uses data looking at the decision of households in Bangladesh to switch drinking water wells in response to their well being marked as unsafe or not. The outcome of interest is whether or not household i switched wells: $y_i = 1$(household i switched to a new well). The variables of interest for this question are `switch`, which is y_i above; `arsenic`, the level of arsenic in the respondent's well; and `dist`, the distance (in metres) of the closest known safe well.

a)

(SAMPLE SOLUTION)

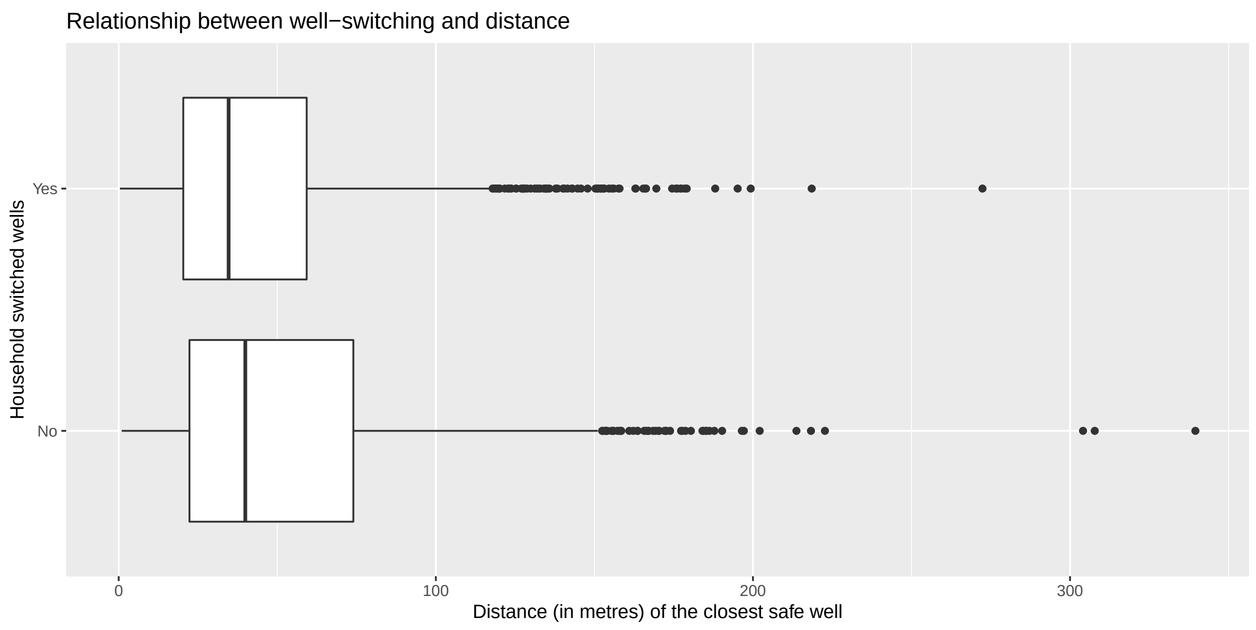
There are no missing values or duplicates in the dataset. There were 3020 total respondents, of whom 1737 (57.5%) switched wells, while the remaining respondents did not. Since the outcome is binary, we will first construct a set of boxplots comparing well-switching versus arsenic.

```
d <- read.table(url("http://www.stat.columbia.edu/~gelman/arm/examples/arsenic/wells.dat"))
wells <- tibble(switch=d$switch,arsenic=d$arsenic,dist=d$dist) %>%
  mutate(ars_c = arsenic-mean(arsenic), dist_c = dist-mean(dist),
         log_ars_c = log(arsenic)-mean(log(arsenic)),
         switch_cat=ifelse(switch==0,"No","Yes"))

wells %>% ggplot(aes(x=switch_cat)) +
  geom_boxplot(aes(y=arsenic)) + coord_flip() +
  labs(x="Household switched wells",
       y="Arsenic level",
       title="Relationship between well-switching and arsenic level")
```



We immediately notice that respondents who decided to switch had a higher median arsenic level in their well; there is also a wider variation of arsenic across respondents. Both respondents who switched and those who did not had a large number of outliers, although the outlying values for those who did tend to be larger.

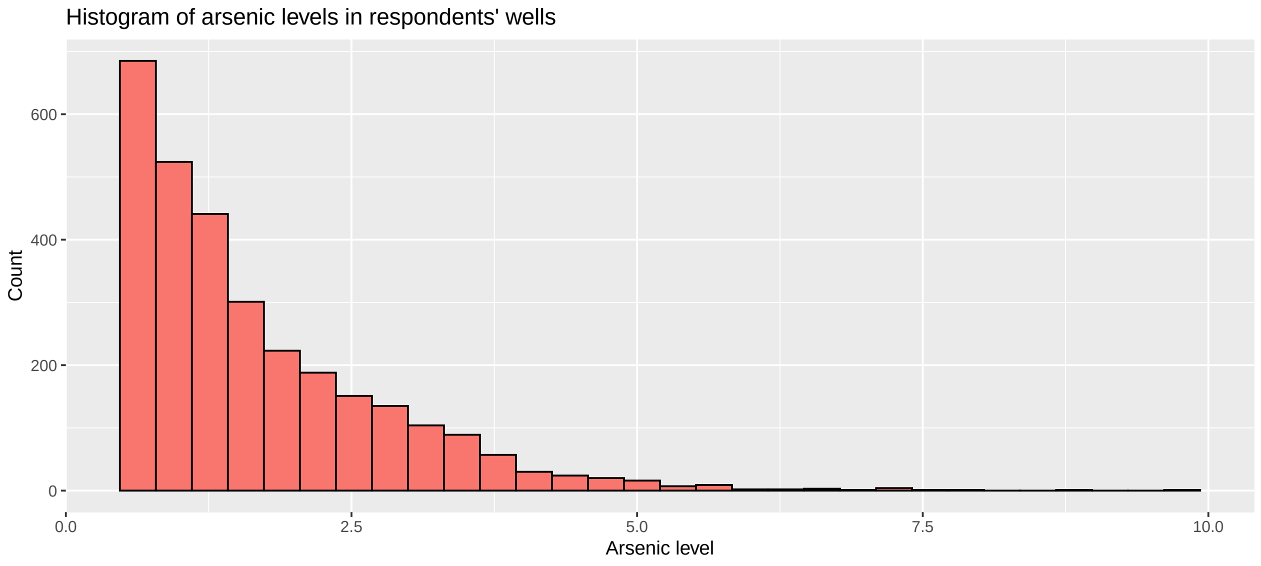
```
wells %>% ggplot(aes(x=switch_cat)) +
  geom_boxplot(aes(y=dist)) + coord_flip() +
  labs(x="Household switched wells",
       y="Distance (in metres) of the closest safe well",
       title="Relationship between well-switching and distance")
```



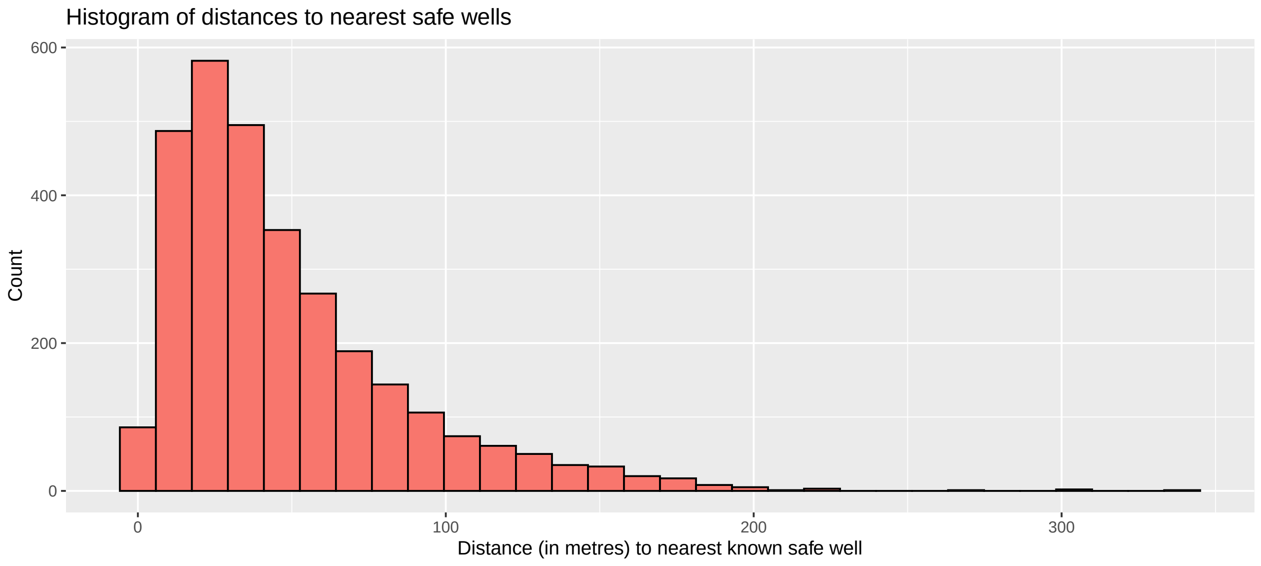
It appears that there was a shorter median distance to the closest known safe well for respondents who decided to switch than for those who did not, and a smaller variation across respondents. At first glance, this variable appears to be slightly less influential than arsenic, since the different between categories is less pronounced. However, note that we have a very large sample size (3020) and thus, even a small difference in median could prove to be significant. As before, there are many outliers for both categories, with generally larger values for respondents who did not switch.

We will now display histograms for the arsenic levels and the distances to the nearest known safe wells.

```
wells %>% ggplot(aes(x=arsenic)) +
  geom_histogram(col='black',fill='#F8766D') +
  labs(x="Arsenic level",
       y="Count", title="Histogram of arsenic levels in respondents' wells")
```

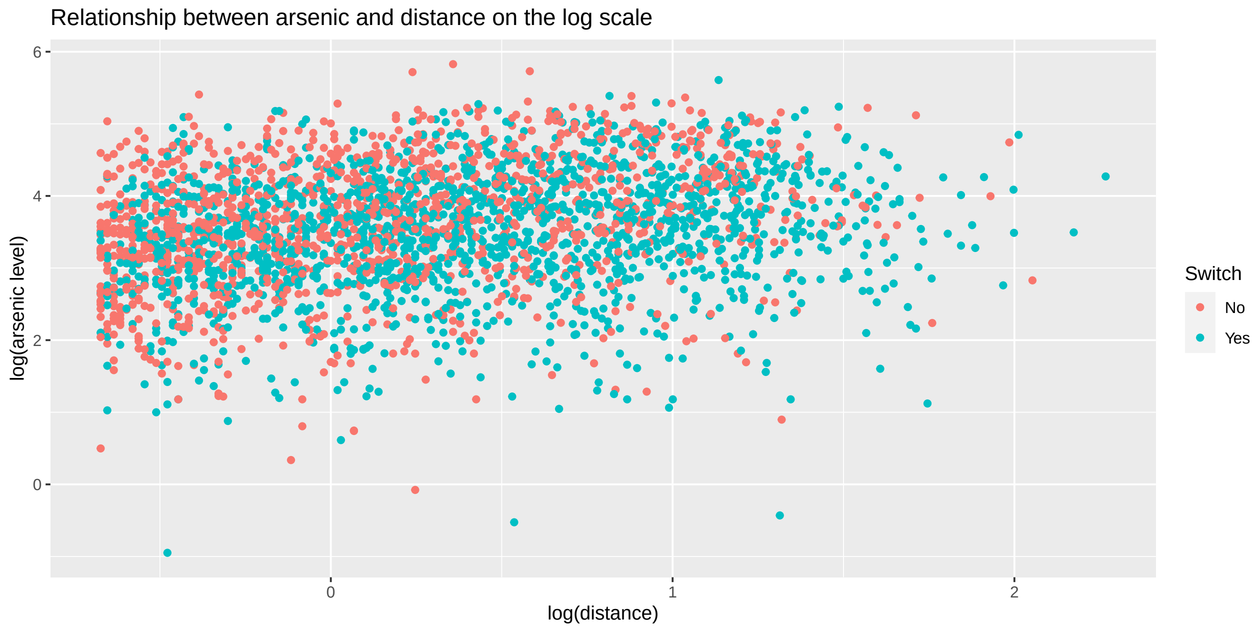


```
wells %>% ggplot(aes(x=dist)) +
  geom_histogram(col='black',fill='#F8766D') +
  labs(x="Distance (in metres) to nearest known safe well",
       y="Count", title="Histogram of distances to nearest safe wells")
```



This confirms what we initially observed when inspecting the boxplots: both the distributions of arsenic levels and distances are heavily right-skewed. Note that there were no wells with an arsenic level below 0.5 by design. It might be interesting to fit a model which treats arsenic as a categorical variable with safe vs. unsafe levels, i.e. below 0.5 versus above 0.5. We will now plot the relationship between arsenic and distance, taking into account whether each responded switched wells or not.

```
wells %>% ggplot(aes(x=log(arsenic),y=log(dist),col=switch_cat)) +
  geom_point() +
  labs(x="log(distance)",
       y="log(arsenic level)",
       title="Relationship between arsenic and distance on the log scale") +
  labs(col="Switch")
```



There does not appear to be a notable relationship between arsenic levels and distance to nearest safe well, regardless of whether the respondent switched wells or not. Both relationships are slightly positive, with the "Yes" category having somewhat higher variability. The correlation between the `arsenic` and `distance` variables has been computed to be 0.178.

b)

Assume $y_i \sim \text{Bernoulli}(p_i)$, where p_i refers to the probability of switching. We consider two candidate models: - Model 1:

$$\text{logit}(p_i) = \beta_0 + \beta_1 \cdot (d_i - \bar{d}) + \beta_2 \cdot (a_i - \bar{a}) + \beta_3 \cdot (d_i - \bar{d})(a_i - \bar{a})$$

- Model 2:

$$\text{logit}(p_i) = \beta_0 + \beta_1 \cdot (d_i - \bar{d}) + \beta_2 \cdot (\log(a_i) - \overline{\log(a)}) + \beta_3 \cdot (d_i - \bar{d})(\log(a_i) - \overline{\log(a)})$$

where d_i is distance and a_i is arsenic level. We will fit both of these models using Stan and put $N(0,1)$ priors on all βs. We first display the β coefficients for Model 1.

```
summary(mod1)$summary[1:4,]
```

```
##                 mean      se_mean          sd         2.5%          25%
## beta[1]  0.353171704 1.529093e-03 0.040064606  0.273393825  0.326132403
## beta[2] -0.008766878 2.207652e-05 0.001076673 -0.010824882 -0.009510400
## beta[3]  0.468923358 1.632943e-03 0.042997912  0.381824207  0.440434435
## beta[4] -0.001820327 2.162607e-05 0.001011120 -0.003753701 -0.002514405
##                  50%          75%         97.5%     n_eff      Rhat
## beta[1]  0.353880959  0.380243866  0.4284257320  686.5212 1.0013585
## beta[2] -0.008778128 -0.008033989 -0.0066743341 2378.5202 1.0004244
## beta[3]  0.468704674  0.495859591  0.5539747953  693.3503 1.0035452
## beta[4] -0.001838680 -0.001105240  0.0001646955 2186.0029 0.9994497
```

We have that $\exp(\hat{\beta}_0) = \exp(0.352) = 1.424$, which represents the odds of a switch for a subject with average arsenic levels and average distance to the nearest safe well. Then, we find that for a subject with average arsenic levels, the odds of a switch decrease by a factor of $\exp(\hat{\beta}_2) = 1/\exp(-0.009) = 1.009$ for a one-unit increase in distance. We also find that for a subject with average distance, the odds of a switch increase by a factor of $\exp(\hat{\beta}_3) = \exp(0.473) = 1.600$ for a one-unit increase in arsenic level. We now indirectly interpret $\hat{\beta}_4$. If neither arsenic nor distance are set to their average values, the relative effect of arsenic on the odds of a switch depends on distance in the following fashion: $\exp\{a(0.473 - 0.002d)\}$. Similarly, the relative effect of distance on the odds of a switch depends on the arsenic level in the following fashion: $\exp\{d(-0.009 - 0.002a)\}$.

The β coefficients for Model 2 are:

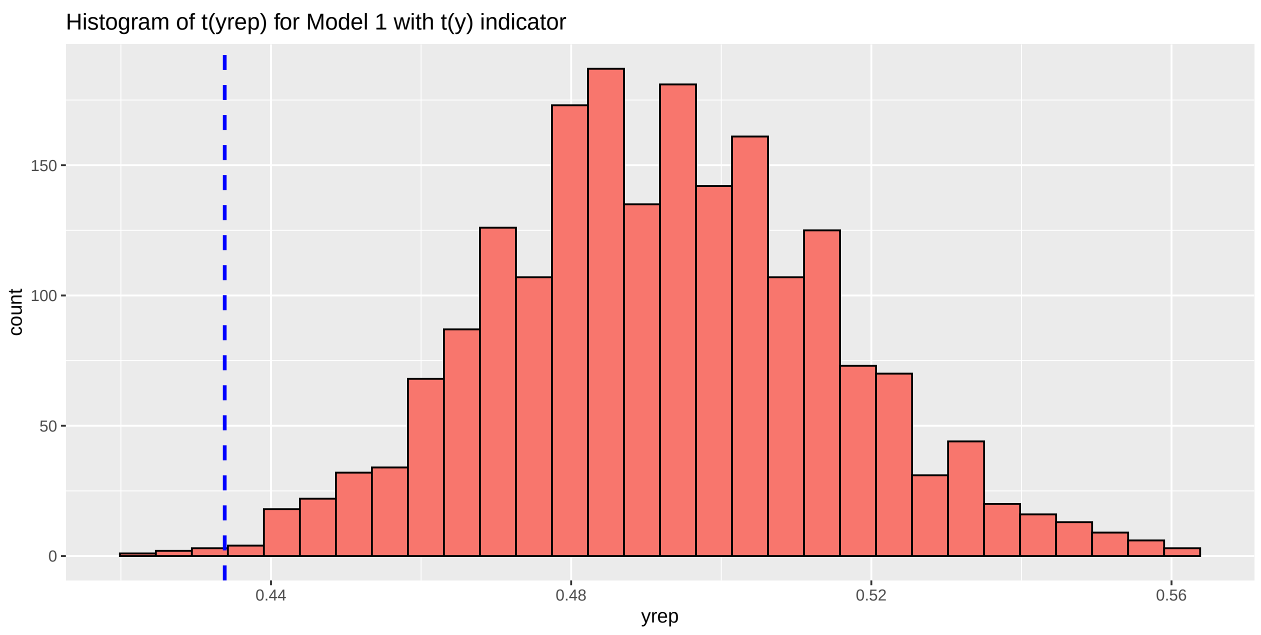
```r
summary(mod2)$summary[1:4,]
```

```
##                 mean      se_mean          sd         2.5%          25%
## beta[1]  0.341267507 1.501087e-03 0.039924300  0.264945144  0.315304308
## beta[2] -0.009492900 2.531866e-05 0.001077293 -0.011628936 -0.010201842
## beta[3]  0.871124721 2.442851e-03 0.067366851  0.733883346  0.826971154
## beta[4] -0.002254956 3.977996e-05 0.001836073 -0.005839001 -0.003457936
##                  50%          75%        97.5%     n_eff      Rhat
## beta[1]  0.341192561  0.367930991  0.422096986  707.3962 1.0052586
## beta[2] -0.009495939 -0.008764504 -0.007373504 1810.4488 1.0012007
## beta[3]  0.870729591  0.917105630  1.003271617  760.4990 1.0015197
## beta[4] -0.002259430 -0.001094580  0.001332309 2130.3507 0.9999599
```

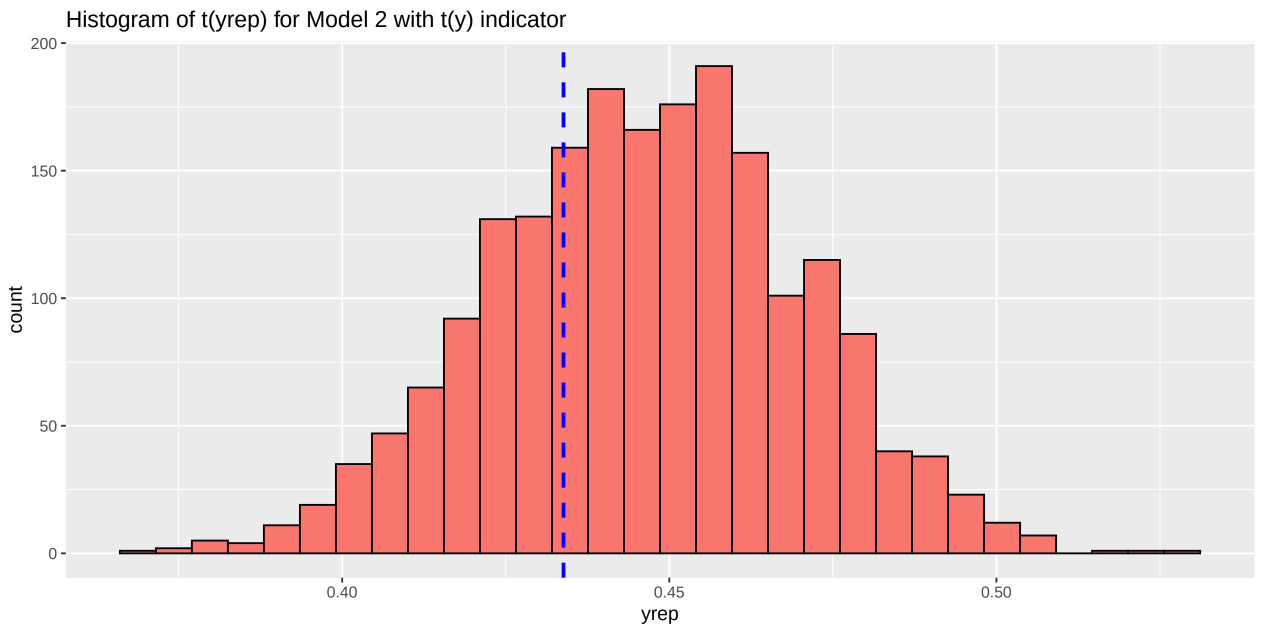
c)

Let $t(\mathbf{y}) = \sum_{i=1}^{n} 1(y_i = 1, a_i < 0.82) / \sum_{i=1}^{n} 1(a_i < 0.82)$, i.e. the proportion of households that switch with arsenic levels less than 0.82. We will calculate $t(\mathbf{y}^{rep})$ for each replicated dataset for each model, plot the resulting histogram for each model, and compare to the observed value of $t(\mathbf{y})$. In each case, the observed value is indicated by a vertical dashed blue line.

```r
yrep1 <- rstan::extract(mod1)[["y_rep"]]
yrep2 <- rstan::extract(mod2)[["y_rep"]]
propyrep1 <- numeric(nrow(yrep1))

for(i in 1:nrow(yrep1)){
  df <- tibble(y=yrep1[i,],ars=wells$arsenic) %>% filter(ars<0.82)
  propyrep1[i] <- sum(df$y)/nrow(df)
}
propyrep2 <- numeric(nrow(yrep2))
for(i in 1:nrow(yrep2)){
  df <- tibble(y=yrep2[i,],ars=wells$arsenic) %>% filter(ars<0.82)
  propyrep2[i] <- sum(df$y)/nrow(df)
}

propnum <- tibble(yrep1=propyrep1,yrep2=propyrep2)
tstat <- wells %>% filter(arsenic<0.82) %>% summarise(sum(switch)/n()) %>% pull()

propnum %>% ggplot(aes(x=yrep1)) +
  geom_histogram(col='black',fill='#F8766D') +
  geom_vline(aes(xintercept=tstat),lty=2,lwd=1,col='blue') +
  labs(title="Histogram of t(yrep) for Model 1 with t(y) indicator",x="yrep")
```



```
propnum %>% ggplot(aes(x=yrep2)) +
  geom_histogram(col='black',fill='#F8766D') +
  geom_vline(aes(xintercept=tstat),lty=2,lwd=1,col='blue') +
  labs(title="Histogram of t(yrep) for Model 2 with t(y) indicator",x="yrep")
```



Calculating $\mathrm{P}(t(\mathbf{y}^{rep}) < t(\mathbf{y}))$ for each model.

```
mean(ifelse(propnum$yrep1<tstat,1,0))
```

```
## [1] 0.002
```

```
mean(ifelse(propnum$yrep2<tstat,1,0))
```

```
## [1] 0.2915
```

Looking at the plots, we note that the observed test statistic $t(\mathbf{y}) = 0.434$ falls closer to the middle of the distribution for Model 2, indicating that Model 2 makes more realistic predictions for lower values of arsenic.

The probability for Model 1 is 0.004 and the probability for Model 2 is 0.282. We note that for Model 1, only 0.4% of test statistics in the replicated datasets are less than the observed test statistic, which does not seem reasonable. For Model 2, 28.2% of test statistics in the replicated datasets are less than $t(\mathbf{y})$, which seems far more plausible.

d)

Use the `loo` package to get estimates of the expected log pointwise predictive density for each point, $ELPD_i$.

```
loglik1 <- rstan::extract(mod1)[["log_lik"]]
loglik2 <- rstan::extract(mod2)[["log_lik"]]
loo1 <- loo(loglik1, save_psis = TRUE)
loo2 <- loo(loglik2, save_psis = TRUE)
loo_compare(loo1, loo2)
```

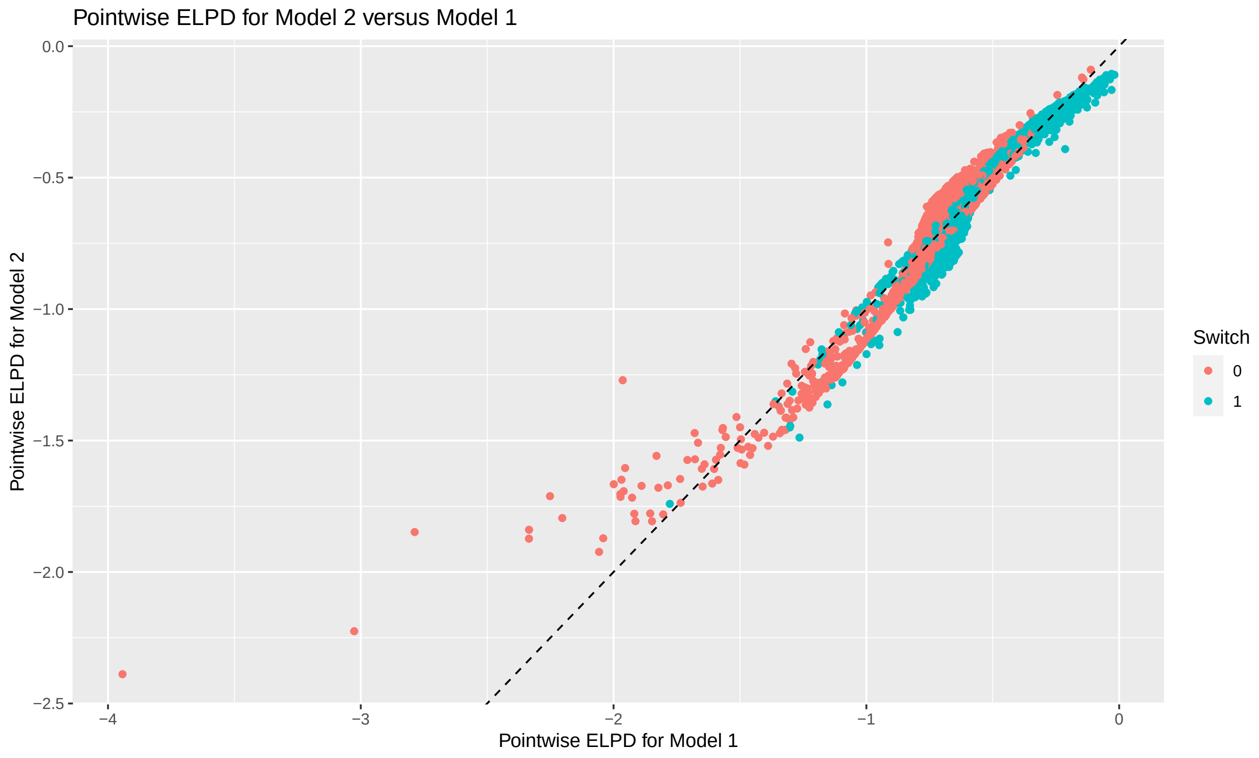
```
##        elpd_diff se_diff
## model2   0.0       0.0
## model1 -15.7       4.4
```

Clearly, Model 2 has a larger $\sum_i ELPD_i$ than Model 1 (the difference is 15.7). Therefore, based on this model-checking method, we would prefer Model 2 over Model 1.

e)

Create a scatter plot of the $ELPD_i$'s for Model 2 versus the $ELPD_i$'s for Model 1.

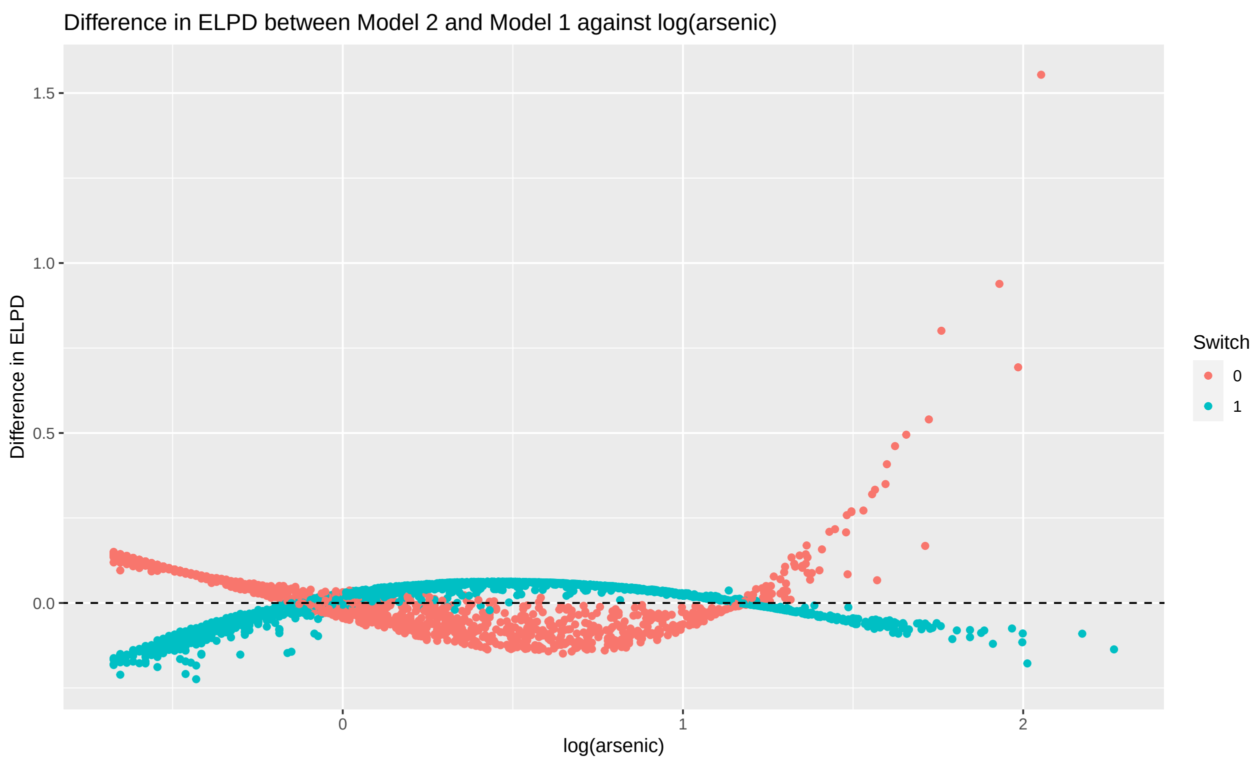
```
df_loo <- tibble(elpd1=loo1$pointwise[,1],elpd2=loo2$pointwise[,1],
                 elpd1se=loo1$pointwise[,2],elpd2se=loo2$pointwise[,2]) %>%
  mutate(switch=wells$switch,arsenic=wells$arsenic)
df_loo %>% ggplot(aes(x=elpd1,col=as.factor(switch))) +
  geom_point(aes(y=elpd2)) + labs(col="Switch",x="Pointwise ELPD for Model 1",
                                  y="Pointwise ELPD for Model 2",
                                  title="Pointwise ELPD for Model 2 versus Model 1") +
  geom_abline(intercept=0,slope=1,lty=2)
```



The $ELPD_i$'s seem to line up reasonably well between the two models at larger values, regardless of whether the individuals switched wells or not. However, the models begin to disagree for lower values of $ELPD_i$, given that the individual did not switch wells; the pattern becomes obvious for values less than -1.5. Model 2 tends to perform better than Model 1, since a significant number of those extreme values are above the 0-1 line. In other words, when predicting the probability of not switching for individuals who did not switch (while fitting a model with all other data points), Model 2 is more certain about the outcome; it is therefore less likely to have false negatives.

Create another scatter plot of the difference in $ELPD_i$'s between the models versus log arsenic.

```
df_loo %>% ggplot(aes(x=log(arsenic),col=as.factor(switch))) +
  geom_point(aes(y=elpd2-elpd1)) + labs(col="Switch",x="log(arsenic)",
                                      y="Difference in ELPD",
                                      title="Difference in ELPD between Model 2 and Model 1 against lo
  geom_abline(intercept=0,slope=0,lty=2)
```



Recall that a more desirable model has a larger $\sum_i ELPD_i$. It appears that regardless of whether the individual switched wells or not, the difference in ELPD between Model 2 and Model 1 is small for non-extreme values of log(arsenic), say between 0 and 1. In this range, Model 1 appears to perform very slightly better than Model 2 if the individuals did not switch, and Model 2 performs very slightly better if individuals switched. This pattern reverses for more extreme values, and becomes very obvious. In particular, Model 2 performs far better if the individual did not switch, for large values of log(arsenic). This is likely the main reason that Model 2 has a larger $\sum_i ELPD_i$. If the individual did switch, Model 1 seems to performs slightly better for extreme log(arsenic) values.

f)

Given the outcome in this case is discrete, we can directly interpret the $ELPD_i$'s. Note that the expected log pointwise predictive density is defined as

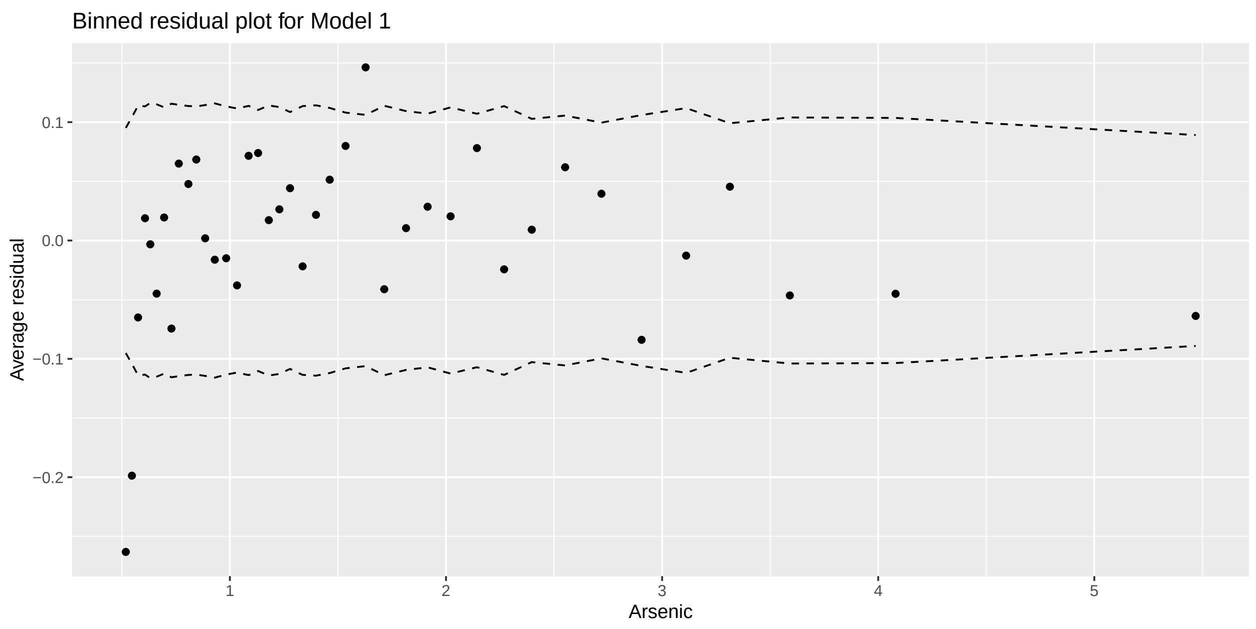
$$\text{elpd}_{\text{LOO}} = \sum_{i=1}^{n} \log p(y_i|\mathbf{y}_{-i}),$$

and hence we have that $\exp(ELPD_i) = p(y_i|\mathbf{y}_{-i})$. If the observation $y_i = 1$, this is the predicted probability of switching for individual i, given a model fit with all other observations except i. If the observation $y_i = 0$, this is the predicted probability of not switching for individual i, given a model fit with all other observations except i.

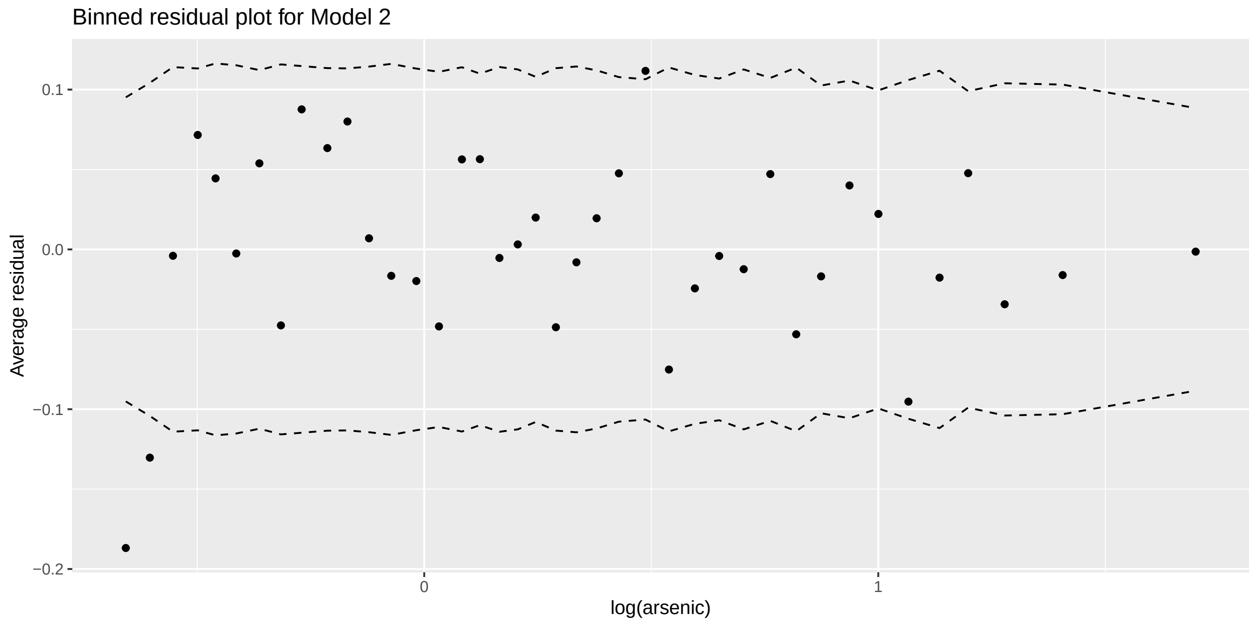
g)

For each model, recode the $ELPD_i$'s to get $\hat{y}_i = \mathbb{E}(Y_i|\mathbf{y}_{-i})$. Then create a binned residual plot, looking at the average residual $y_i - \hat{y}_i$ by arsenic for Model 1 and by log(arsenic) for Model 2, splitting the data such that there are 40 bins. The average residual is shown with a dot for each bin, and we will also add a dashed line to represent ±2 standard errors for each bin. The standard errors are computed by taking the standard deviation of the residuals per bin, and diving by the square root of the number of observations in that bin.

```r
df_loo <- df_loo %>% mutate(fit1 = ifelse(switch==1,exp(elpd1),1-exp(elpd1)),
                            fit2 = ifelse(switch==1,exp(elpd2),1-exp(elpd2)))
df_loo_bin <- df_loo %>% group_by(group=ntile(arsenic,40)) %>%
  summarise(res1=mean(switch-fit1),res2=mean(switch-fit2),
            res1se=sd(switch-fit1)/sqrt(length(fit1)),
            res2se=sd(switch-fit2)/sqrt(length(fit2)),arsenic=mean(arsenic))
df_loo_bin %>% ggplot(aes(x=arsenic)) + geom_point(aes(y=res1)) +
  geom_line(aes(y=2*res1se),lty=2) + geom_line(aes(y=-2*res1se),lty=2) +
  labs(y="Average residual",x="Arsenic",
       title="Binned residual plot for Model 1")
```



```r
df_loo_bin %>% ggplot(aes(x=log(arsenic))) + geom_point(aes(y=res2)) +
  geom_line(aes(y=2*res2se),lty=2) + geom_line(aes(y=-2*res2se),lty=2) +
  labs(y="Average residual",x="log(arsenic)",
       title="Binned residual plot for Model 2")
```



Notice that for both models, 37/40=92.5% of points lie within the confidence bounds, with the outlying values in Model 2 closer to the bounds. There does not appear to be a systematic pattern in the residuals for either model—we observe mostly constant variation—although they appear to be somewhat more clustered for lower arsenic values in Model 1. Both models seem to perform particularly poorly for very low values of arsenic, indicating that there may be a non-linear trend in the predictor at low values, followed by a more linear trend at higher values. Overall, it does appear that Model 2 has a slightly better fit: more even variation in the residuals, and the outlying points are closer to the confidence bounds.