Homework 5

Yanjie Qi, Jianing (Julia) Chen

Due on November 29, 2020 at 11:59 pm

Note: If you are working with a partner, please submit only one homework per group with both names and whether you are taking the course for graduate credit or not. Submit your Rmarkdown (.Rmd) and the compiled pdf on Gauchospace.

Problem 1. Frequentist Coverage of The Bayesian Posterior Interval.

In quiz 1 we explored the importance and difficulty of well-calibrated prior distributions by examining the calibration of subjective intervals. Suppose that $y_1, .., y_n$ is an IID sample from a $Normal(\mu, 1)$. We wish to estimate μ.

1a. For Bayesian inference, we will assume the prior distribution $\mu \sim Normal(0, \frac{1}{\kappa_0})$ for all parts below. Remember, from lecture that we can interpret κ_0 as the pseudo-number of prior observations with sample mean $\mu_0 = 0$. State the posterior distribution of μ given $y_1, .., y_n$. Report the lower and upper bounds of the 95% quantile-based posterior credible interval for μ, using the fact that for a normal distribution with standard eviation σ, approximately 95% of the mass is between $\pm 1.96\sigma$.

The posterior distribution of μ given $y_1, .., y_n$ is normal distributed with $Normal(\mu_n, \tau^2)$, where $\mu_n = \frac{\frac{1}{\tau^2}\mu_0 + \frac{n}{\sigma^2}\bar{y}}{\frac{1}{\tau^2} + \frac{n}{\mu^2}}$, $\tau_n^2 = \frac{1}{\frac{1}{\tau^2} + \frac{n}{\sigma^2}}$.

Now we are given that $y_1, .., y_n \sim Normal(\mu, 1)$ and $\mu \sim Normal(0, \frac{1}{\kappa_0})$, hence:

$$\mu_n = \frac{\frac{1}{\tau^2}\mu_0 + \frac{n}{\sigma^2}\bar{y}}{\frac{1}{\tau^2} + \frac{n}{\mu^2}} = \frac{k_0 0 + \frac{n}{1}\bar{y}}{k_0 + \frac{n}{1}} = \frac{n\bar{y}}{k_0 + n}$$

$$\tau_n^2 = \frac{1}{\frac{1}{\tau^2} + \frac{n}{\sigma^2}} = \frac{1}{k_0 + \frac{n}{1}} = \frac{1}{k_0 + n}$$

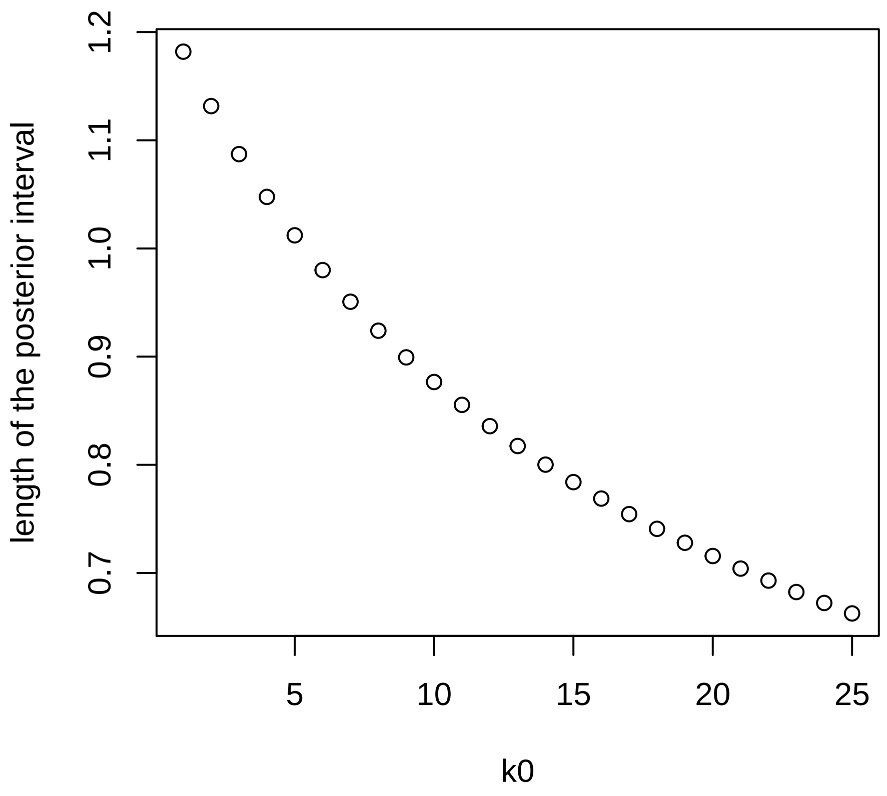
The 95% quantile-based posterior credible interval would be:

$$(\frac{n\bar{y}}{k_0 + n} - 1.96\sqrt{\frac{1}{k_0 + n}}, \frac{n\bar{y}}{k_0 + n} + 1.96\sqrt{\frac{1}{k_0 + n}})$$

1b. Plot the length of the posterior credible interval as a function of κ_0, for $\kappa_0 = 1, 2, ..., 25$ assuming $n = 10$. Report how this prior parameter effects the length of the posterior interval and why this makes intuitive sense.

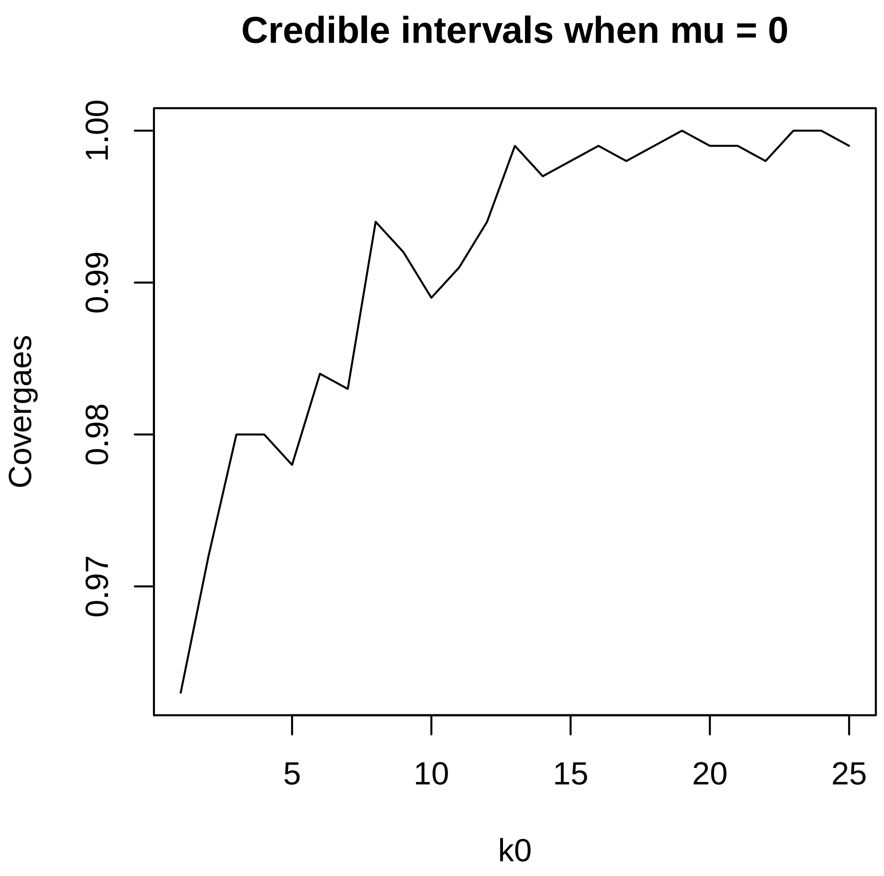
```
n=10
ybar=100

result = c()
mun = c()
sd = c()
```

```r
for (k0 in 1:25){
    mun[k0] = n*ybar / (k0 + n)
    sd[k0] = 1 / (k0 + n)
    lower = mun - 1.96*sqrt(sd)
    upper = mun + 1.96*sqrt(sd)
    result= upper-lower}

plot(x= c(1:25),y=result,
     ylab="length of the posterior interval",
     xlab="k0")
```



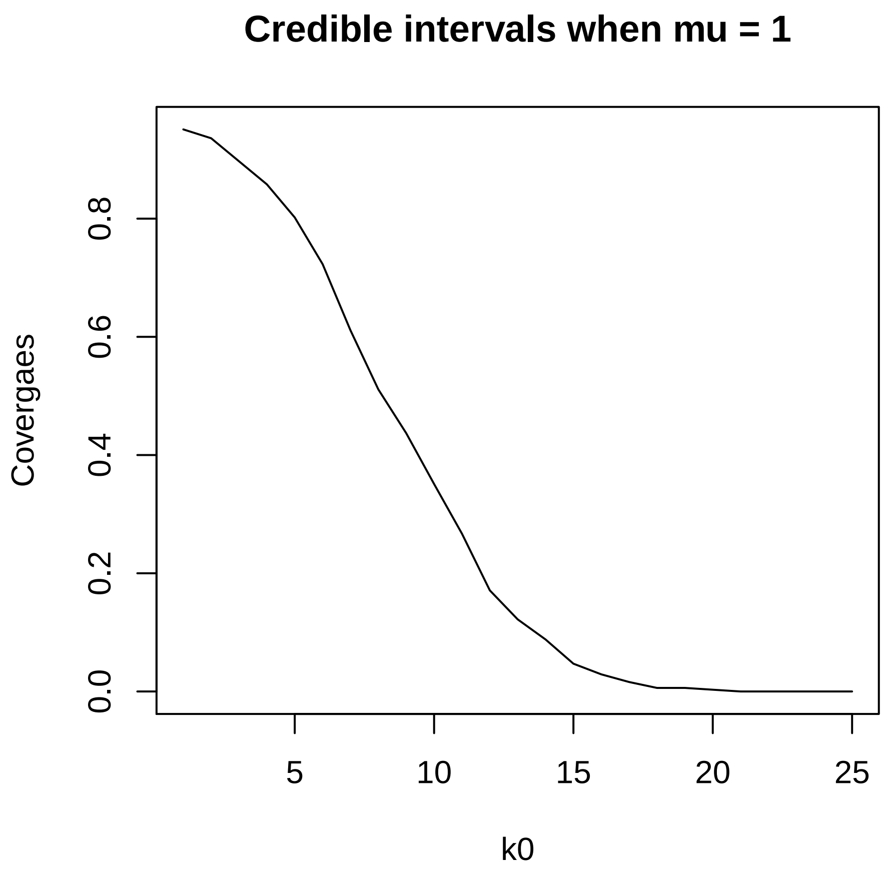
1c. Now we will evaluate the *frequentist coverage* of the posterior credible interval on simulated data. Generate 1000 data sets where the true value of $\mu = 0$ and $n = 10$. For each dataset, compute the posterior 95% interval endpoints (from the previous part) and see if it the interval covers the true value of $\mu = 0$. Compute the frequentist coverage as the fraction of these 1000 posterior 95% credible intervals that contain $\mu = 0$. Do this for each value of $\kappa_0 = 1, 2, ..., 25$. Plot the coverage as a function of κ_0.

```r
n=10

count = 0
coverages = c()
for (k in 1:25){
    for (i in 1:1000){
        samples = rnorm(10, mean=0, sd=1)
        #mun[k] = (n*sum(samples)/n) / (k + n)
        #sd[k] = 1 / (k + n)
        #lower = mun[k] - 1.96*sqrt(sd)
        #upper = mun[k] + 1.96*sqrt(sd)
        #interval = lower - upper
        interval = ((n*sum(samples)/n)/(k+10))+c(-1,1)*1.96*sqrt(1/(k+n))
        if (between (0,interval[1],interval[2])){
            count = count + 1
        }
    }
    coverages = c(coverages,count/1000)
    count = 0
}


plot(x = c(1:25), y = coverages,type = 'l',
     ylab = "Covergaes",
     xlab = "k0",
     main = "Credible intervals when mu = 0")
```



1d. Repeat the 1c but now generate data assuming the true $\mu = 1$.

```r
n=10

count = 0
coverages = c()
for (k in 1:25){
    for (i in 1:1000){
        samples = rnorm(10, mean=1, sd=1)
        interval = ((n*sum(samples)/n)/(k+10))+c(-1,1)*1.96*sqrt(1/(k+n))
        if (between (1,interval[1],interval[2])){
            count = count + 1
        }
    }
    coverages = c(coverages,count/1000)
    count = 0
}


plot(x = c(1:25), y = coverages,type = 'l',
     ylab = "Covergaes",
     xlab = "k0",
     main = "Credible intervals when mu = 1")
```



1e. Explain the differences between the coverage plots when the true $\mu = 0$ and the true $\mu = 1$. For what values of κ_0 do you see closer to nominal coverage (i.e. 95%)? For what values does your posterior interval tend to overcover (the interval covers the true value more than 95% of the time)? Undercover (the interval covers the true value less than 95% of the time)? Why does this make sense?

From two plots above, we observed that when $\mu = 0$, the relationship between coverage and k_0 is a increase function (coverage increase as k_0 increase), as $\mu = 1$, the relationship between coverage and k_0 becomes a decrease function (coverage decrease as k_0 increase).

In $\mu = 0$ plot, when k_0 is around 1 to have 95% coverage, in $\mu = 1$ plot, it is hard to tell what value of k0 will have 95% coverage, but lower k_0 tends to give closer coverage to 95%.

When $\mu = 0$, higher value of k_0 leads to overcoverage, when $\mu = 1$, higher values of k_0 lead to undercoverage.

This makes sense because when k_0 increases, at time $\mu = 0$, the real value of $\mu = 0$ meets our prior belief, so coverage will increase, but when $\mu = 1$, the real value of $\mu = 1$ does not match our prior belief, hence it will go to 0.

Bayesian inference for the normal distribution in Stan.

Create a new Stan file by selecting "Stan file" in the Rstudio menu. Save it as `IQ_model.stan`. We will make some basic modifications to the template example in the default Stan file for this problem. Consider the IQ example used from class. Scoring on IQ tests is designed to yield a N(100, 15) distribution for the general population. We observe IQ scores for a sample of n individuals from a particular town, $y_1, \ldots y_n \sim N(\mu, \sigma^2)$. Our goal is to estimate the population mean in the town. Assume the $p(\mu, \sigma) = p(\mu \mid \sigma)p(\sigma)$, where $p(\mu \mid \sigma)$ is $N(\mu_0, \sigma/\sqrt{\kappa_0})$ and $p(\sigma)$ is Gamma(a, b). Before you administer the IQ test you believe the town is no different than the rest of the population, so you assume a prior mean for μ of $\mu_0 = 100$, but you aren't to sure about this a priori and so you set $\kappa_0 = 1$ (the effective number of pseudo-observations). Similarly, a priori you assume σ has a mean of 15 (to match the intended standard deviation of the IQ test) and so you decide on setting $a = 15$ and $b = 1$ (remember, the mean of a Gamma is a/b). Assume the following IQ scores are observed:

```r
y <- c(70, 85, 111, 111, 115, 120, 123)
n <- length(y)
```

2a. Make a scatter plot of the posterior distribution of the median, μ, and the precision, $1/\sigma^2$. Put μ on the x-axis and $1/\sigma^2$ on the y-axis. What is the posterior relationship between μ and $1/\sigma^2$? Why does this make sense? *Hint:* review the lecture notes.

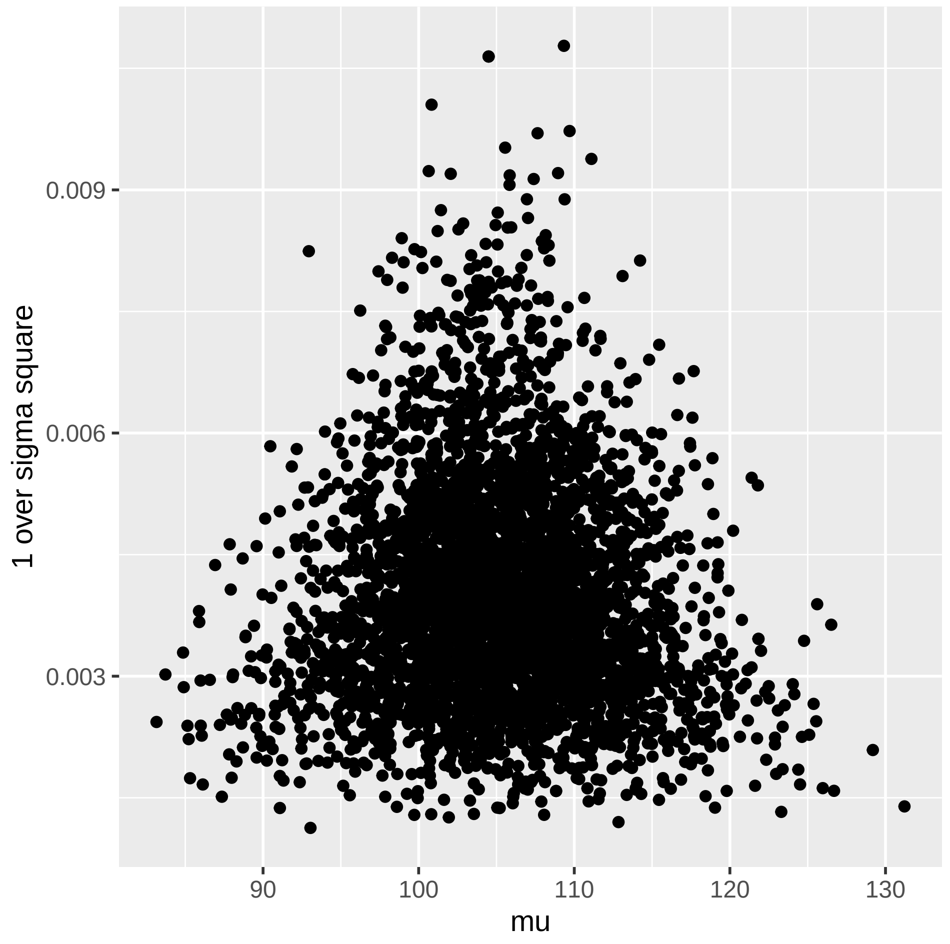
Type your answer here, replacing this text.

```r
y <- c(70, 85, 111, 111, 115, 120, 123)
n <- length(y)
q2_stan_model = stan_model(file = "IQ_model.stan")
```

```
## Trying to compile a simple C file

## Running /opt/conda/lib/R/bin/R CMD SHLIB foo.c
## x86_64-conda_cos6-linux-gnu-cc -I"/opt/conda/lib/R/include" -DNDEBUG   -I"/opt/conda/lib/R/library/R
## In file included from /opt/conda/lib/R/library/RcppEigen/include/Eigen/Core:88:0,
##                  from /opt/conda/lib/R/library/RcppEigen/include/Eigen/Dense:1,
##                  from /opt/conda/lib/R/library/StanHeaders/include/stan/math/prim/mat/fun/Eigen.hpp:
##                  from <command-line>:0:
## /opt/conda/lib/R/library/RcppEigen/include/Eigen/src/Core/util/Macros.h:613:1: error: unknown type na
##  namespace Eigen {
##  ^~~~~~~~~
## /opt/conda/lib/R/library/RcppEigen/include/Eigen/src/Core/util/Macros.h:613:17: error: expected '=',
##  namespace Eigen {
##                  ^
## In file included from /opt/conda/lib/R/library/RcppEigen/include/Eigen/Dense:1:0,
```

```
##                     from /opt/conda/lib/R/library/StanHeaders/include/stan/math/prim/mat/fun/Eigen.hpp:
##                     from <command-line>:0:
## /opt/conda/lib/R/library/RcppEigen/include/Eigen/Core:96:10: fatal error: complex: No such file or d
##  #include <complex>
##           ^~~~~~~~~
## compilation terminated.
## make: *** [/opt/conda/lib/R/etc/Makeconf:168: foo.o] Error 1
```

```
q2_stan_fit <- rstan::sampling(q2_stan_model,
                           data = list(N = n, k0 = 1, y = y),
                           refresh = 0)
q2_df <- data.frame(
    mu = as.numeric(rstan::extract(q2_stan_fit)$mu),
    sigma = as.numeric(1/(rstan::extract(q2_stan_fit)$sigma)^2)
    )

ggplot(q2_df, aes(x=mu, y=sigma)) + geom_point() +
    ylab("1 over sigma square")
```



2b. You are interested in whether the mean IQ in the town is greater than the mean IQ in the overall population. Use Stan to find the posterior probability that μ is greater than 100.

```
library(rstan)
y <- c(70, 85, 111, 111, 115, 120, 123)
```

```r
n <- length(y)

total = length(rstan::extract(q2_stan_fit)$mu)
all = as.numeric(rstan::extract(q2_stan_fit)$mu)
Lg100 = sum(all > 100)
Lg100/total
```

```
## [1] 0.79275
```

2b. You notice that two of the seven scores are significantly lower than the other five. You think that the normal distribution may not be the most appropriate model, in particular because you believe some people in this town are likely have extreme low and extreme high scores. One solution to this is to use a model that is more robust to these kinds of outliers. The Student's t distribution and the Laplace distribution are two so called "heavy-tailed distribution" which have higher probabilities of outliers (i.e. observations further from the mean). Heavy-tailed distributions are useful in modeling because they are more robust to outliers. Fit the model assuming now that the IQ scores in the town have a Laplace distribution, that is $y_1, \ldots, y_n \sim Laplace(\mu, \sigma)$. Create a copy of the previous stan file, and name it "IQ_laplace_model.stan". *Hint:* In the Stan file you can replace `normal` with `double_exponential` in the model section, another name for the Laplce distribution. Like the normal distribution it has two arguments, μ and σ. Keep the same prior distribution, $p(\mu, \sigma)$ as used in the normal model. Under the Laplace model, what is the posterior probability that the median IQ in the town is greater than 100? How does this compare to the probability under the normal model? Why does this make sense?

```r
y <- c(70, 85, 111, 111, 115, 120, 123)
n <- length(y)

q2_stan_model2 = stan_model(file = "IQ_laplace_model.stan")
```

```
## Trying to compile a simple C file

## Running /opt/conda/lib/R/bin/R CMD SHLIB foo.c
## x86_64-conda_cos6-linux-gnu-cc -I"/opt/conda/lib/R/include" -DNDEBUG   -I"/opt/conda/lib/R/library/R
## In file included from /opt/conda/lib/R/library/RcppEigen/include/Eigen/Core:88:0,
##                  from /opt/conda/lib/R/library/RcppEigen/include/Eigen/Dense:1,
##                  from /opt/conda/lib/R/library/StanHeaders/include/stan/math/prim/mat/fun/Eigen.hpp:
##                  from <command-line>:0:
## /opt/conda/lib/R/library/RcppEigen/include/Eigen/src/Core/util/Macros.h:613:1: error: unknown type na
##  namespace Eigen {
##  ^~~~~~~~~
## /opt/conda/lib/R/library/RcppEigen/include/Eigen/src/Core/util/Macros.h:613:17: error: expected '=',
##  namespace Eigen {
##                  ^
## In file included from /opt/conda/lib/R/library/RcppEigen/include/Eigen/Dense:1:0,
##                  from /opt/conda/lib/R/library/StanHeaders/include/stan/math/prim/mat/fun/Eigen.hpp:
##                  from <command-line>:0:
## /opt/conda/lib/R/library/RcppEigen/include/Eigen/Core:96:10: fatal error: complex: No such file or di
##  #include <complex>
##           ^~~~~~~~~
## compilation terminated.
## make: *** [/opt/conda/lib/R/etc/Makeconf:168: foo.o] Error 1
```

```r
q2_stan_fit2 <- rstan::sampling(q2_stan_model2,
                                data = list(N = n, k0 = 1, y = y),
                                refresh = 0)

total2 = length(rstan::extract(q2_stan_fit2)$mu)
```

```r
all2 = as.numeric(rstan::extract(q2_stan_fit2)$mu)
Lg100_2 = sum(all2 > 100)
Lg100_2/total2
```

```
## [1] 0.9605
```

The probability under the Laplace model is larger than the probability under the normal model, since 0.9605 > 0.79275 . This makes sense because the Laplace distribution is a heavy-tailed distribution, it more robust to extremely high or low compared to the normal distribution.

Logistic regression for pesticide toxicity data.

A environmental agency is testing the effects of a pesticide that can cause acute poisoning in bees, the world's most important pollinator of food crops. The environmental agency collects data on exposure to different levels of the pestidicide in parts per million (ppm). The agency also identifies collapsed beehives, which they expect could be due to acute pesticide poisoning. In the data they collect, each observation is pair (x_i, y_i), where x_i represents the dosage of the pollutant and y_i represents whether or not the hive survived. Take $y_i = 1$ means that the beehive has collapsed from poisoning and $y_i = 0$ means the beehive survived. The agency collects data at several different sites, each of which was exposed to a different dosages. The resulting data can be seen below:

```r
x <- c(1.06, 1.41, 1.85, 1.5, 0.46, 1.21, 1.25, 1.09,
       1.76, 1.75, 1.47, 1.03, 1.1, 1.41, 1.83, 1.17,
       1.5, 1.64, 1.34, 1.31)

y <- c(0, 1, 1, 1, 0, 1, 1, 1, 1, 1,
       1, 0, 0, 1, 1, 0, 0, 1, 1, 0)
```

Assume that beehiv collapse, y_i, given pollutant exposure level x_i, is $Y_i \sim \text{Bernoulli}(\theta(x_i))$, where $\theta(x_i)$ is the probability of death given dosage x_i. We will assume that $\text{logit}(\theta_i(x_i)) = \alpha + \beta x_i$ where $\text{logit}(\theta)$ is defined as $\log(\theta/(1-\theta))$. This model is known as *logistic regression* and is one of the most common methods for modeling probabilities of binary events.

3a. Solve for $\theta_i(x_i)$ as a function of α and β by inverting the logit function. If you haven't seen logistic regression before (it is covered in more detail in PSTAT 127 and PSTAT131), it is essentially a generalization of linear regression for binary outcomes. The inverse-logit function maps the linear part, $\alpha + \beta x_i$, which can be any real-valued number into the interval [0, 1] (since we are modeling probabilities of binary outcome, we need the mean outcome to be confined to this range).

Type your answer here, replacing this text.

$$\begin{aligned}
\text{logit}(\theta_i(x_i)) &= \alpha + \beta x_i \\
\log\left(\frac{\theta_i(x_i)}{1-\theta_i(x_i)}\right) &= \alpha + \beta x_i \\
\frac{\theta_i(x_i)}{1-\theta_i(x_i)} &= e^{\alpha+\beta x_i} \\
\theta_i(x_i) &= \frac{e^{\alpha+\beta x_i}}{1+e^{\alpha+\beta x_i}}
\end{aligned}$$

3b The dose at which there is a 50% chance of beehvive collapse, $\theta(x_i) = 0.5$, is known as LD50 ("letha dose 50%"), and is often of interest in toxicology studies. Solve for LD50 as a function of α and β.

Type your answer here, replacing this text.

$$\begin{aligned}\theta(x_i) &= 0.5\\ \frac{e^{\alpha+\beta x_i}}{1+e^{\alpha+\beta x_i}} &= 0.5\\ e^{\alpha+\beta x_i} &= 0.5(1+e^{\alpha+\beta x_i})\\ 0.5e^{\alpha+\beta x_i} &= 0.5\\ \alpha+\beta x_i &= ln(1) = 0\\ \beta x_i &= -\alpha\end{aligned}$$

Therefore $x_i = -\frac{\alpha}{\beta}$

3c Implement the logistic regression model in stan by reproducing the stan model described here: https://mc-stan.org/docs/2_18/stan-users-guide/logistic-probit-regression-section.html. Run the stan model on the beehive data to get Monte Carlo samples. Compute Monte Carlo samples of the LD50 by applying the function derived in the previous part to your α and β samples. Report and estimate of the posterior mean of the LD50 by computing the sample average of all Monte Carlo samples of LD50.

```
x <- c(1.06, 1.41, 1.85, 1.5, 0.46, 1.21, 1.25, 1.09,
       1.76, 1.75, 1.47, 1.03, 1.1, 1.41, 1.83, 1.17,
       1.5, 1.64, 1.34, 1.31)

y <- c(0, 1, 1, 1, 0, 1, 1, 1, 1, 1,
       1, 0, 0, 1, 1, 0, 0, 1, 1, 0)

n <- length(y)


q3_stan_model <- stan_model(file = "log.stan")
```

```
## Trying to compile a simple C file

## Running /opt/conda/lib/R/bin/R CMD SHLIB foo.c
## x86_64-conda_cos6-linux-gnu-cc -I"/opt/conda/lib/R/include" -DNDEBUG   -I"/opt/conda/lib/R/library/R
## In file included from /opt/conda/lib/R/library/RcppEigen/include/Eigen/Core:88:0,
##                  from /opt/conda/lib/R/library/RcppEigen/include/Eigen/Dense:1,
##                  from /opt/conda/lib/R/library/StanHeaders/include/stan/math/prim/mat/fun/Eigen.hpp:
##                  from <command-line>:0:
## /opt/conda/lib/R/library/RcppEigen/include/Eigen/src/Core/util/Macros.h:613:1: error: unknown type n
##  namespace Eigen {
##  ^~~~~~~~~
## /opt/conda/lib/R/library/RcppEigen/include/Eigen/src/Core/util/Macros.h:613:17: error: expected '=',
##  namespace Eigen {
##                  ^
## In file included from /opt/conda/lib/R/library/RcppEigen/include/Eigen/Dense:1:0,
##                  from /opt/conda/lib/R/library/StanHeaders/include/stan/math/prim/mat/fun/Eigen.hpp:
##                  from <command-line>:0:
## /opt/conda/lib/R/library/RcppEigen/include/Eigen/Core:96:10: fatal error: complex: No such file or d
##  #include <complex>
##           ^~~~~~~~~
## compilation terminated.
## make: *** [/opt/conda/lib/R/etc/Makeconf:168: foo.o] Error 1
```

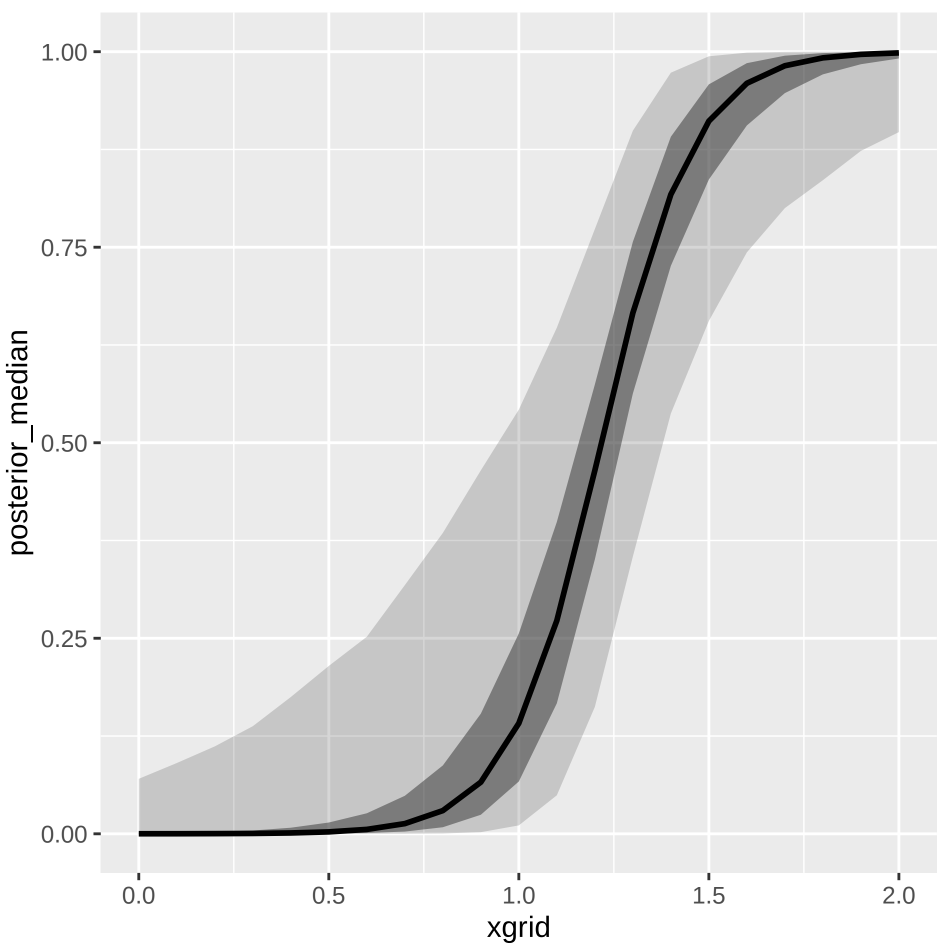
```
q3_stan_fit <-  sampling(q3_stan_model, data=list(N=n, x=x, y=y), refresh=0)
```

```r
samples3 <- extract(q3_stan_fit)
logistic_alpha_samples <- samples3$alpha
logistic_beta_samples <- samples3$beta

LD50 <- -logistic_alpha_samples/logistic_beta_samples
mean(LD50)
```

```
## [1] 1.199367
```

3d. Make a plot showing both 50% and 95% confidence band for the probability of a hive collapse as a function of pollutant exposure, $\Pr(y = 1 \mid \alpha, \beta, x)$. Plot your data on a grid of x-values from $x = 0$ to 2. *Hint:* see lab 7 for a similar example.

```r
xgrid <- seq(0, 2, by=0.1)

compute_curve <- function(sample){
    alpha <- sample[1]
    beta <- sample[2]
    y_values <- exp(alpha + beta*xgrid)/(1+exp(alpha + beta*xgrid))
}

res <- apply(cbind(logistic_alpha_samples, logistic_beta_samples), 1, compute_curve)
quantiles <- apply(res, 1, function(x) quantile(x, c(0.025, 0.25, 0.75, 0.975)))

posterior_mean <- rowMeans(res)
posterior_median <- apply(res, 1, median)

tibble(x=xgrid,
       q025 = quantiles[1,],
       q25 = quantiles[2,],
       q75 = quantiles[3,],
       q975 = quantiles[4,],
       median = posterior_median)%>%

ggplot() +
geom_ribbon(aes(x=xgrid, ymin=q25, ymax=q75), alpha=0.5)+
geom_ribbon(aes(x=xgrid, ymin=q025, ymax=q975), alpha=0.2)+
geom_line(aes(x=xgrid, y=posterior_median), size=1)
```


posterior_median
1.00
0.75
0.50
0.25
0.00
0.0
0.5
1.0
1.5
2.0
xgrid