Homework 4

Jessica Nguyen and Tristan Chen

Due on February 27, 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 the "random facts calibration game" 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 is

$$p(\mu|y, \sigma^2) = L(\mu) * p(\mu)$$

$$\propto e^{-\frac{(\bar{y}-\mu)^2}{2\sigma^2/n}} * e^{-\frac{(\mu-\mu_0)^2}{2\tau^2}}$$

$$\propto exp[-\frac{1}{2}(\frac{n}{\sigma^2} + \frac{1}{\tau^2})\mu^2 - 2(\frac{n\bar{y}}{\sigma^2} + \frac{\mu_0}{\tau^2})\mu]$$

$$\propto exp\left[-\frac{1}{2}\left(\frac{n}{\sigma^2} + \frac{1}{\tau^2}\right)\left(\mu - \frac{\frac{n\bar{y}}{\sigma^2} + \frac{\mu_0}{\tau^2}}{\frac{n}{\sigma^2} + \frac{1}{\tau^2}}\right)^2\right]$$

$$\tau^2 = \frac{1}{k_0}, \sigma^2 = 1, \mu_o = 0$$

$$\propto exp\left[-\frac{1}{2}(n + k_0)(\mu - \frac{n\bar{y}}{n + k_0})^2\right]$$

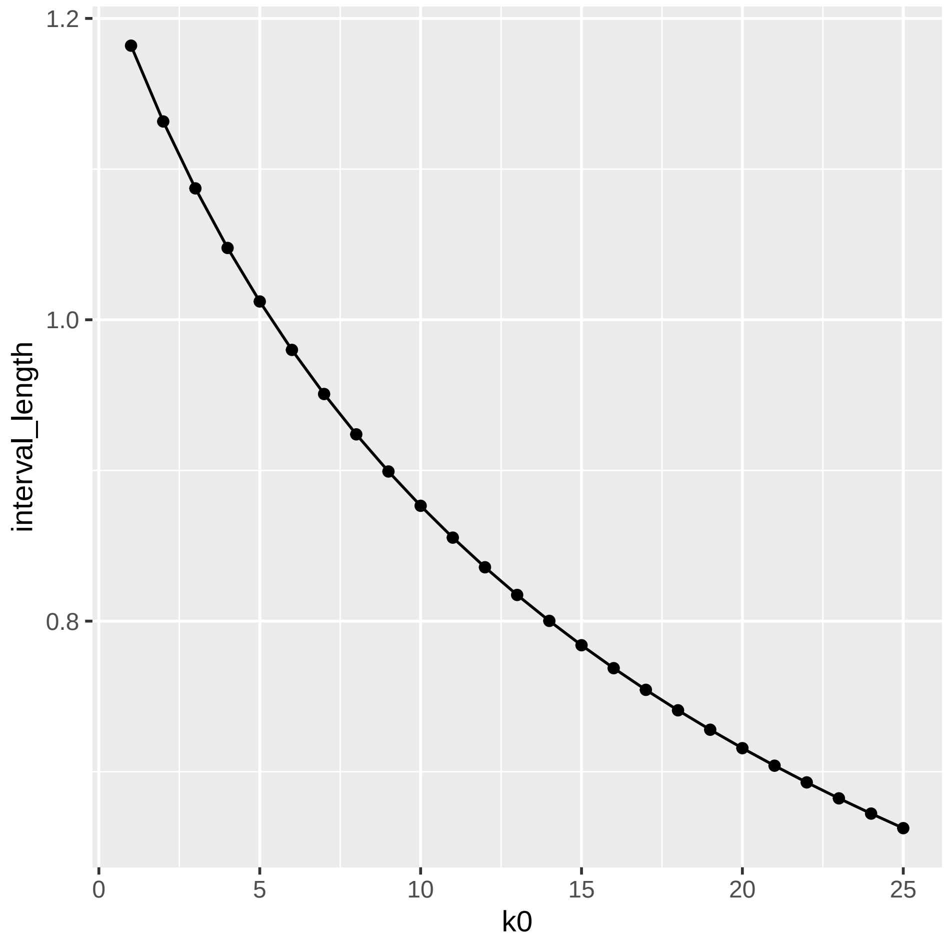
$$p(\mu|y, \sigma^2) = N\left(\frac{n\bar{y}}{n + k_0}, \frac{1}{n + k_0}\right)$$

Bounds: $[\mu_n - 1.96 * sqrt(\frac{1}{n+k_0}), \mu_n + 1.96 * sqrt(\frac{1}{n+k_0})]$

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.

```
# Use 'interval_length' to store lengths of credible intervals
interval_length <- numeric(25)
n <- 10
```

```
for (k0 in 1:25){
  interval_length[k0] = 2 * 1.96 * sqrt(1/(k0+n))
}
## PLOT SOLUTION
# YOUR CODE HERE
ggplot(data = data.frame(interval_length), aes(x = 1:25, y = interval_length)) + geom_line() + xlab("k0
```



```
. = ottr::check("tests/q1b.R")
```

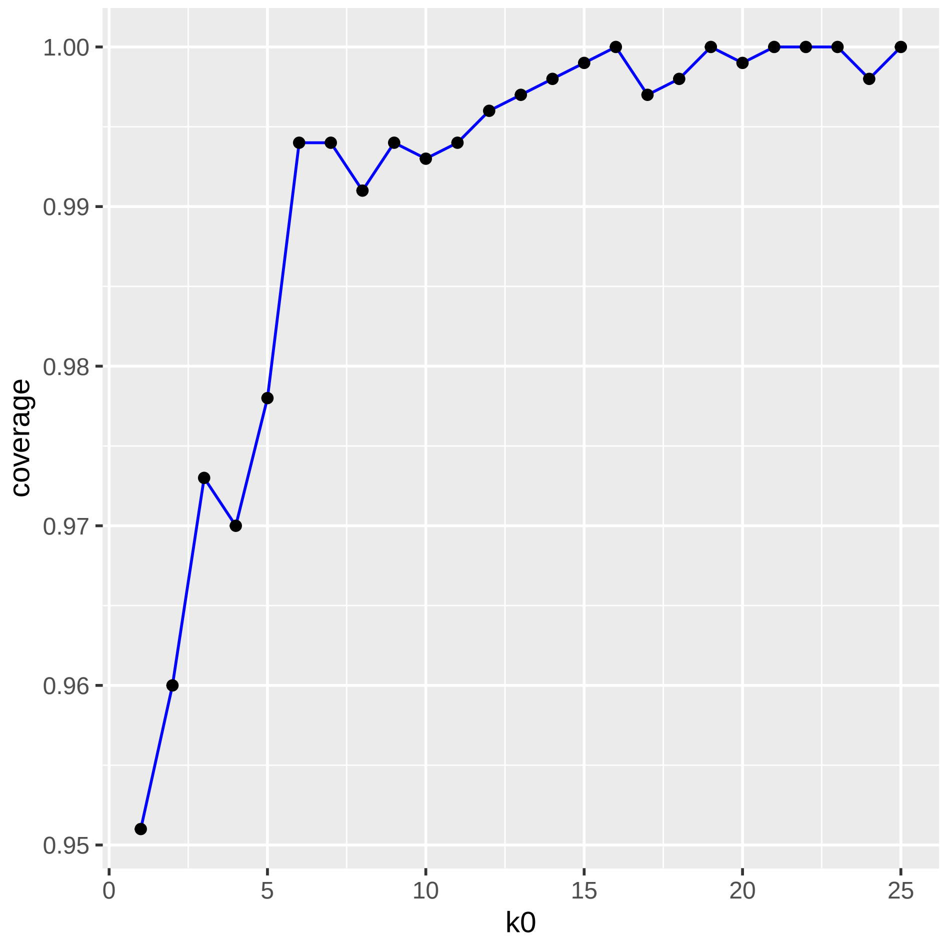
```
##
## All tests passed!
```

As k_0 increases our interval length decreases. This makes sense because the confidence interval is the $\mu_n \pm 1.96 * \sqrt{\frac{1}{n+k_0}}$, and k_0 is in the denominator of $\sqrt{\frac{1}{n+k_0}}$, so as k_0 increases, $\sqrt{\frac{1}{n+k_0}}$ becomes smaller and smaller, therefore the interval length decreases.

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. Store these 25 coverage values in vector called `coverage`.

```r
set.seed(402)
## Fill in the vector called "coverage", which stores the fraction of intervals containing \mu = 0 for
coverage <- numeric(25)
n <- 10

# YOUR CODE HERE
for (k_0 in 1:25){
  count <- 0
  for(data in 1:1000){
    y <- rnorm(n, mean = 0, 1)
    posterior_mean <- ((mean(y) * n) / (n + k_0))
    credit_interval <- qnorm(c(0.025, 0.975), mean = posterior_mean, sqrt(1/(n+k_0)))
    if((credit_interval[1] < 0) & (credit_interval[2] > 0))
    {
      count <- count + 1
    }
  }
  coverage[k_0] <- count / 1000
}
ggplot(data = data.frame(coverage), aes(x = 1:25, y = coverage)) + geom_line(color = "blue") + xlab("k0
```



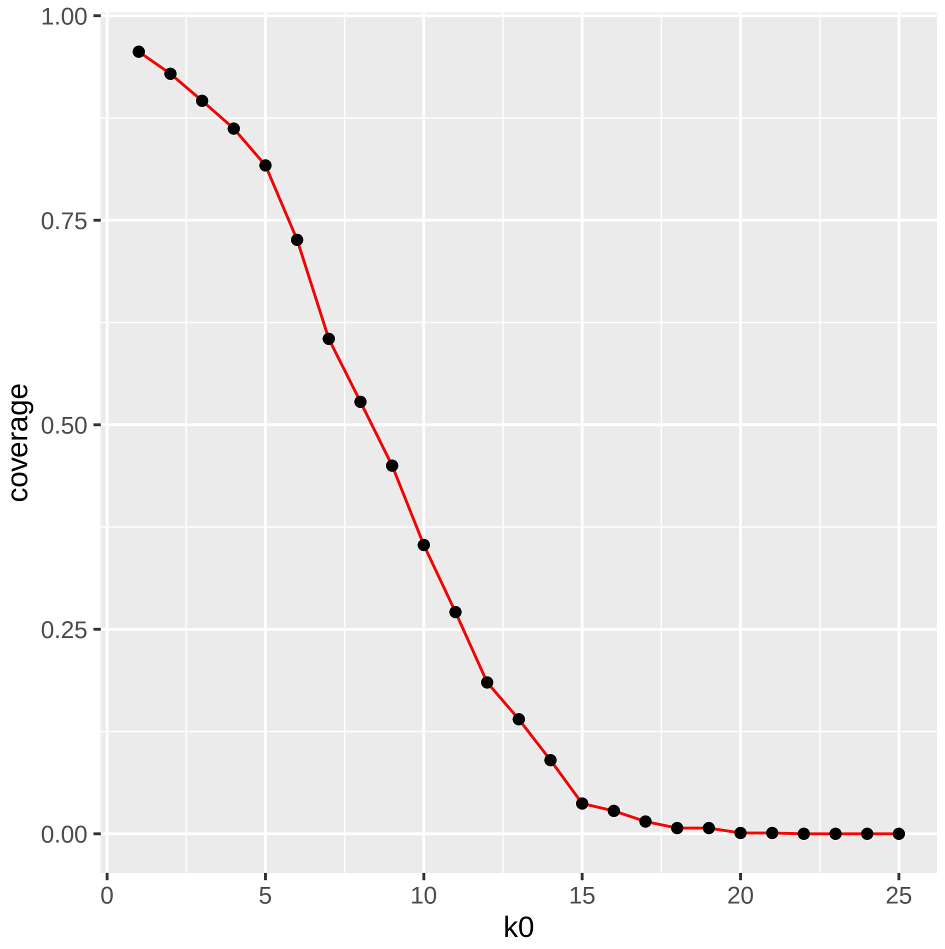
```r
. = ottr::check("tests/q1c.R")
```

```
##
## All tests passed!
```

1d. Repeat 1c but now generate data assuming the true $\mu = 1$. Again, store these 25 coverage values in vector called `coverage`.

```r
## Fill in the vector called "coverage", which stores the fraction of intervals containing \mu = 1 for 
coverage <- numeric(25)
n <- 10

# YOUR CODE HERE
for (k in 1:25){
  count <- 0
  for(data in 1:1000){
    y <- rnorm(n, 1, 1)
    posterior_mean <- 10 * mean(y) / (n + k)
    credit_interval <- qnorm(c(0.025, 0.975), mean = posterior_mean, sqrt(1/(n+k)))
    if((credit_interval[1] < 1) & (credit_interval[2] > 1))
    {
      count <- count + 1
    }
  }
  coverage[k] <- count / 1000
}
ggplot(data = data.frame(coverage), aes(x = 1:25, y = coverage)) + geom_line(color = "red") + xlab("k0"
```



```
. = ottr::check("tests/q1d.R")
```

```
##
## All tests passed!
```

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?

When the true μ is 0, the coverage gets higher when k_0, but we see the exact opposite when $\mu = 1$: the coverage starts off high, and gets lower and lower as k_0 increases. For both $\mu = 0$ and $\mu = 1$, I see the lower κ_0 values (when $\kappa_0 = 1$, 2) having close to nominal coverage. The posterior interval tends to overcover for larger values of κ_0 when the true $\mu = 0$, while the posterior interval tends to undercover for larger values of κ_0 when the true $\mu = 1$. This makes sense because as κ_0 gets larger, the strength of the prior increases and our credible interval becomes more accurate. The strength of the prior increases, which should make the posterior μ closer to 0, which is why as κ_0 increases there is an overcover when the true $\mu = 1$ and an undercover when true $\mu = 1$.

Problem 2. Goal Scoring in the Women's World Cup

The Chinese Women's soccer team recently won the AFC Women's Asian Cup. Suppose you are interested in sutyding the World Cup performance of this soccer team. Let λ be the be the average number of goals scored by the team. We will analyze λ using the Gamma-Poisson model where data Y_i is the observed number of goals scored in the ith World Cup game, ie. we have $Y_i|\lambda \sim Pois(\lambda)$. *A priori*, we expect the rate of goal scoring to be $\lambda \sim Gamma(a, b)$. According to a sports analyst, they believe that λ follows a Gamma distribution with $a = 1$ and $b = 0.25$.

2a. Compute the theoretical posterior parameters a, b, and also the posterior mean.

```
y <- c(4, 7, 3, 2, 3) # Number of goals in each game

post_a <- sum(y) + 1
post_b <- length(y) + 0.25
post_mu <- post_a / post_b
```

```
. = ottr::check("tests/q2a.R")
```

2b. Create a new Stan file by selecting "Stan file" and name it `women_cup.stan`, use Rstan to report and estimate the posterior mean of the scoring rate by computing the sample average of all Monte Carlo samples of λ.

```
soccer_model <- stan_model("women_cup.stan")
```

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

## Running /opt/conda/lib/R/bin/R CMD SHLIB foo.c
## x86_64-conda-linux-gnu-cc -I"/opt/conda/lib/R/include" -DNDEBUG   -I"/opt/conda/lib/R/library/Rcpp/i
## In file included from /opt/conda/lib/R/library/RcppEigen/include/Eigen/Core:88,
##                  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>:
## /opt/conda/lib/R/library/RcppEigen/include/Eigen/src/Core/util/Macros.h:628:1: error: unknown type na
##   628 | namespace Eigen {
##       | ^~~~~~~~~
## /opt/conda/lib/R/library/RcppEigen/include/Eigen/src/Core/util/Macros.h:628:17: error: expected '=',
##   628 | namespace Eigen {
##       |                 ^
## In file included 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>:
## /opt/conda/lib/R/library/RcppEigen/include/Eigen/Core:96:10: fatal error: complex: No such file or d
##    96 | #include <complex>
##       |          ^~~~~~~~~
## compilation terminated.
## make: *** [/opt/conda/lib/R/etc/Makeconf:170: foo.o] Error 1
```

```
# YOUR CODE HERE
stan_fit <- rstan::sampling(soccer_model, data = list(N=5, y=y), refresh = 0)
samples <- rstan::extract(stan_fit)

post_mean <- mean(samples$lambda)
post_mean
```

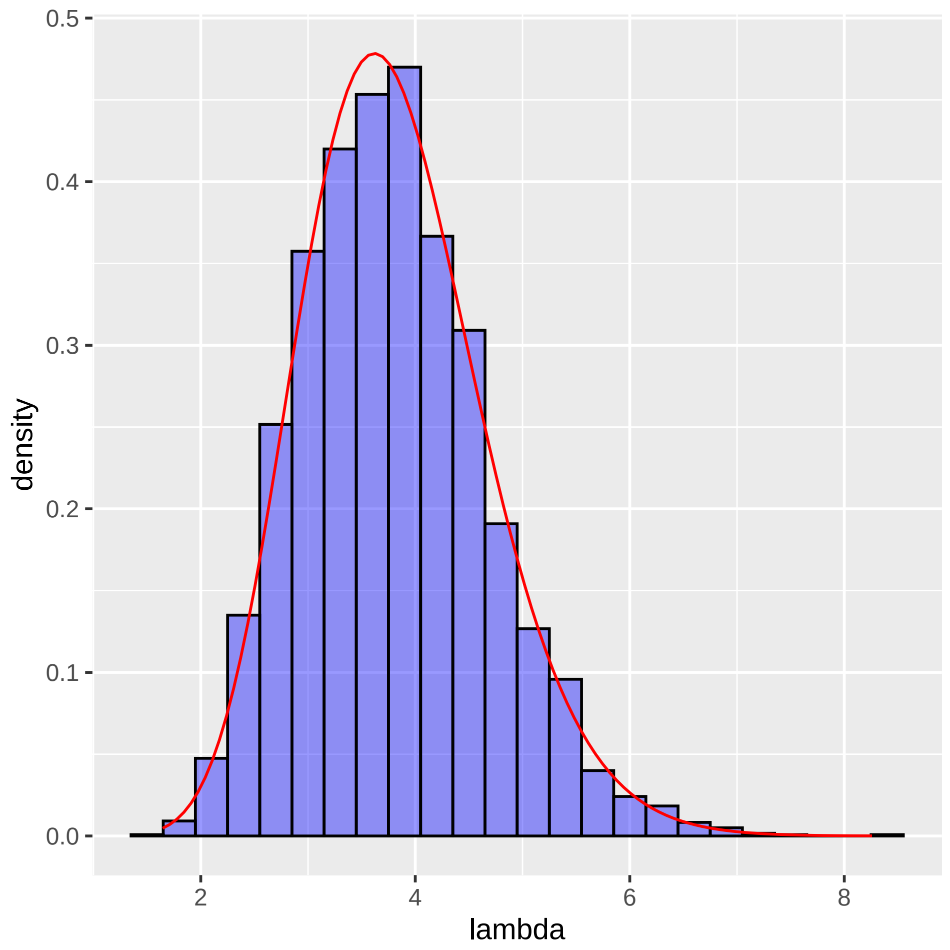
```
## [1] 3.796735
```

```
. = ottr::check("tests/q2b.R")
```

```
##
## All tests passed!
```

2c. Create a histogram of the Monte Carlo samples of λ and add a line showing the theoretical posterior of density of λ. Do the Monte Carlo samples coincide with the theoretical density?

```
# YOUR CODE HERE
dfs <- data.frame(samples)
ggplot(data=dfs, aes(lambda)) + geom_histogram(binwidth = 0.3, alpha = 0.4, fill="blue", color ="black"
```



The theoretical density follows the Monte Carlo samples. Looking at the graph, the curve closely follows the shape of the distribution of the Monte Carlo samples.

2d. Use the Monte Carlo samples from Stan to compute the mean of predicative posterior distribution to estimate the distribution of expected goals scored for next game played by the Chinese women's soccer team.

```
pred_samples <- rpois(4000, lambda = samples$lambda)


pred_mean <- mean(pred_samples)
pred_mean
```

```
## [1] 3.79325
```

```
. = ottr::check("tests/q2d.R")
```

```
##
## All tests passed!
```

Problem 3. Bayesian inference for the normal distribution in Stan.

Create a new Stan file and name it `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:

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

```
a <- 15
b <- 1
mu0 <- 100
k0 <-1
```

3a. Make a scatter plot of the posterior distribution of the mean, 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.

```
normal_stan_model <- stan_model("IQ_model.stan")
```

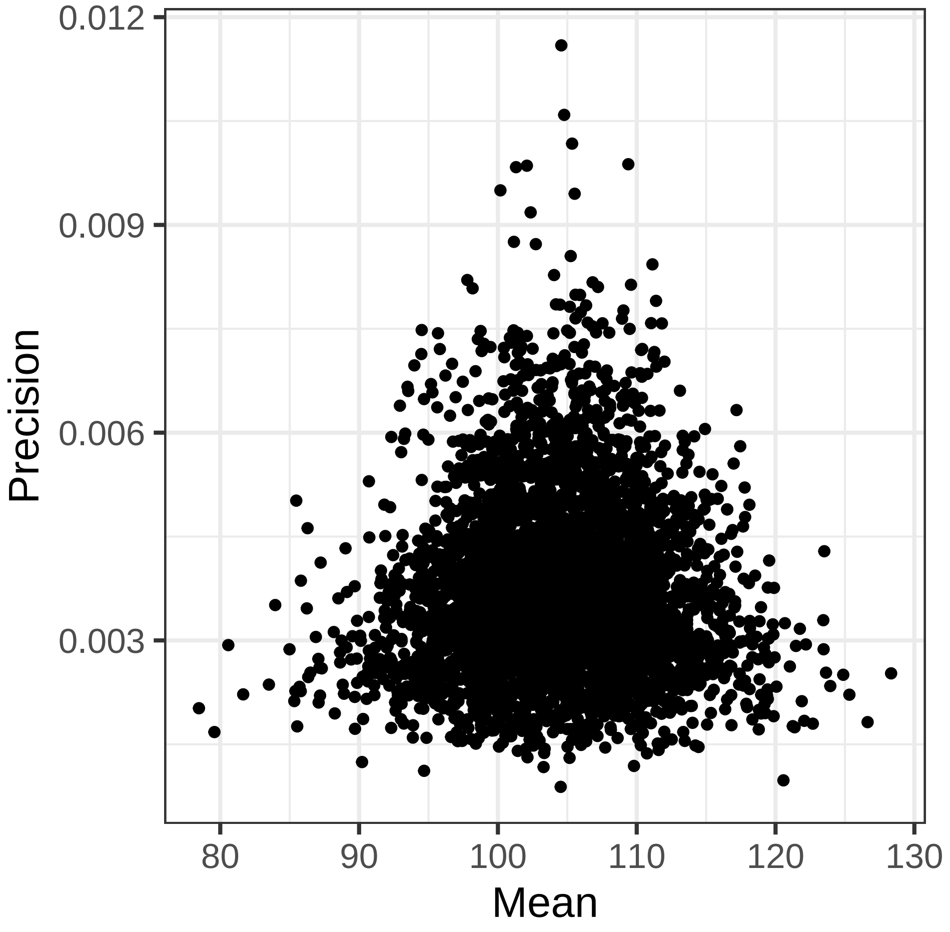
```
## Trying to compile a simple C file

## Running /opt/conda/lib/R/bin/R CMD SHLIB foo.c
## x86_64-conda-linux-gnu-cc -I"/opt/conda/lib/R/include" -DNDEBUG   -I"/opt/conda/lib/R/library/Rcpp/i
## In file included from /opt/conda/lib/R/library/RcppEigen/include/Eigen/Core:88,
##                  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>:
## /opt/conda/lib/R/library/RcppEigen/include/Eigen/src/Core/util/Macros.h:628:1: error: unknown type na
##   628 | namespace Eigen {
##       | ^~~~~~~~~
## /opt/conda/lib/R/library/RcppEigen/include/Eigen/src/Core/util/Macros.h:628:17: error: expected '=',
##   628 | namespace Eigen {
##       |                 ^
## In file included 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>:
## /opt/conda/lib/R/library/RcppEigen/include/Eigen/Core:96:10: fatal error: complex: No such file or d
##    96 | #include <complex>
##       |          ^~~~~~~~~
## compilation terminated.
## make: *** [/opt/conda/lib/R/etc/Makeconf:170: foo.o] Error 1
```

```
# Run rstan and extract the samples
# YOUR CODE HERE
set.seed(333)
stan_fit <- rstan::sampling(normal_stan_model, data=list(N=n, y=y, a=a, b=b, mu0=mu0, k0=k0), refresh=0
samples <- rstan::extract(stan_fit)

mu_samples <- samples$mu
```

```r
sigma_samples <- samples$sigma
precision_samples <- 1 / sigma_samples^2
## Make the plot
# YOUR CODE HERE
tibble(Mean = mu_samples, Precision=precision_samples)%>%ggplot() + geom_point(aes(x=Mean, y=Precision)
```



```r
. = ottr::check("tests/q3a.R")
```

```
##
## All tests passed!
```

As the precision increases, the variance decreases, so the samples get closer and closer to the mean. This is why there is a triangular shape.

3b. 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.

```r
# YOUR CODE HERE
set.seed(333)
mean(mu_samples > 100)
```

```
## [1] 0.773
```

Under the normal model, the posterior probability μ is greater than 100 is 0.773.

3c. 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 $y_1, \dots, y_n \sim Laplace(\mu, \sigma)$. 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 Laplace 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
sm_t <- stan_model("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-linux-gnu-cc -I"/opt/conda/lib/R/include" -DNDEBUG   -I"/opt/conda/lib/R/library/Rcpp/i
## In file included from /opt/conda/lib/R/library/RcppEigen/include/Eigen/Core:88,
##                  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>:
## /opt/conda/lib/R/library/RcppEigen/include/Eigen/src/Core/util/Macros.h:628:1: error: unknown type na
##   628 | namespace Eigen {
##       | ^~~~~~~~~
## /opt/conda/lib/R/library/RcppEigen/include/Eigen/src/Core/util/Macros.h:628:17: error: expected '=',
##   628 | namespace Eigen {
##       |                 ^
## In file included 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>:
## /opt/conda/lib/R/library/RcppEigen/include/Eigen/Core:96:10: fatal error: complex: No such file or d
##    96 | #include <complex>
##       |          ^~~~~~~~~
## compilation terminated.
## make: *** [/opt/conda/lib/R/etc/Makeconf:170: foo.o] Error 1
```

```r
## Run rstan and extract samples
# YOUR CODE HERE
set.seed(333)
laplace_stan_fit <- rstan::sampling(sm_t, data=list(N=n, y=y, a=a, b=b, mu0=mu0, k0=k0), refresh=0)
laplace_samples <- rstan::extract(laplace_stan_fit)

mu_samples <- laplace_samples$mu
sigma_samples <- laplace_samples$sigma

post_prob_laplace <- mean(mu_samples > 100)
post_prob_laplace
```

```
## [1] 0.925
```

```r
. = ottr::check("tests/q3c.R")
```

```
##
## All tests passed!
```

The probability under the model assuming that the IQ scores follow a Laplace distribution is 0.925. This is higher than the probability under the normal model, which was 0.773. This makes sense because the Laplace distribution is more robust towards outliers. The probability when using the Laplace distribution is thus less effected by the two significantly lower scores.