Bayesian Learning - Lab 01

Lakshidaa Saigiridharan (laksa656) and Maximilian Pfundstein (maxpf364)

2019-03-29

Contents

1 Bernoulli

Exercise: Let $y_1, ..., y_n|\theta \sim \text{Bern}(\theta)$, and assume that you have obtained a sample with $s = 14$ successes in $n = 20$ trials. Assuma a $\text{Beta}(\alpha_0, \beta_0)$ prior for θ and let $\alpha_0 = \beta_0 = 2$.

a) Draw random numbers from the posterior distribution $\theta|y \sim \text{Beta}(\alpha_0 + s, \beta_0 + f), y = (y_1, ..., y_n)$, and verify that the posterior mean and standard deviation converges to the true values as the number of random draws grows large.

b) Use simulation (`nDraws = 10000`) to compute the posterior propability $\Pr(\theta < 0.4|y)$ and compare with the exact value [Hint: `pbeta()`].

c) Compute the posterior distribution of the log-odds $\phi = log\left(\frac{\theta}{1-\theta}\right)$ by simulation (`nDraws = 10000`). [Hint: `hist()` and `density` might come in handy]

1.1 Drawing from the Posterior

First we define the parameters as we need them later.

```
################################################################################
# Exercise 1.a)
################################################################################

# Parameters
n = 20
s = 14
f = n - s

# Prior
alpha_z = 2
```

```r
beta_z = 2

# Posterior
alpha_post = alpha_z + s
beta_post = beta_z + f
```

The posterior is given as Beta(α_n, β_n) where $\alpha_n = \alpha_0 + s$ and $\beta_n = \beta_0 + f$. Therefore the theoretical mean is given by:

$$\mathrm{E}[X] = \frac{\alpha_n}{\alpha_n + \beta_n}$$

And the standard deviation by:

$$\mathrm{sd}[X] = \sqrt{\frac{\alpha_n \beta_n}{(\alpha_n + \beta_n)^2(\alpha_n + \beta_n + 1)}}$$

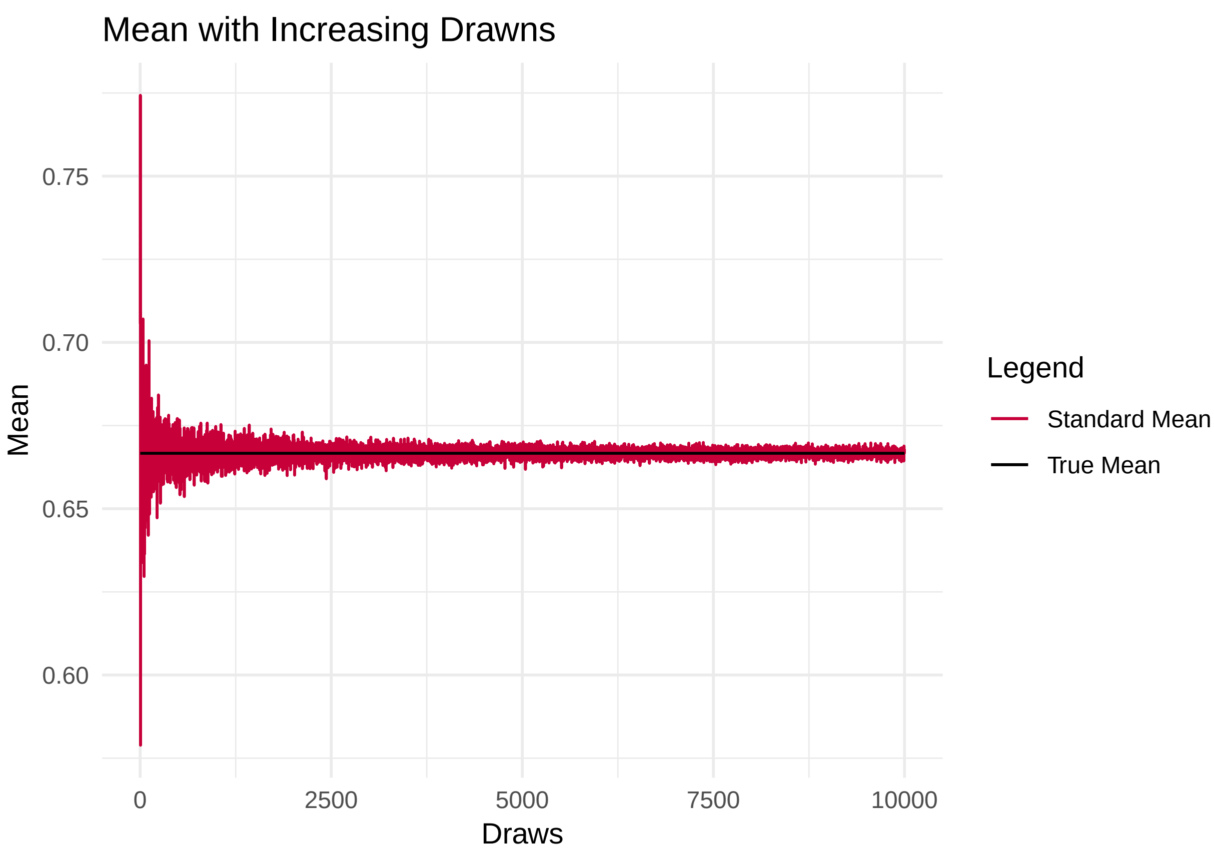
So let's calculate this.

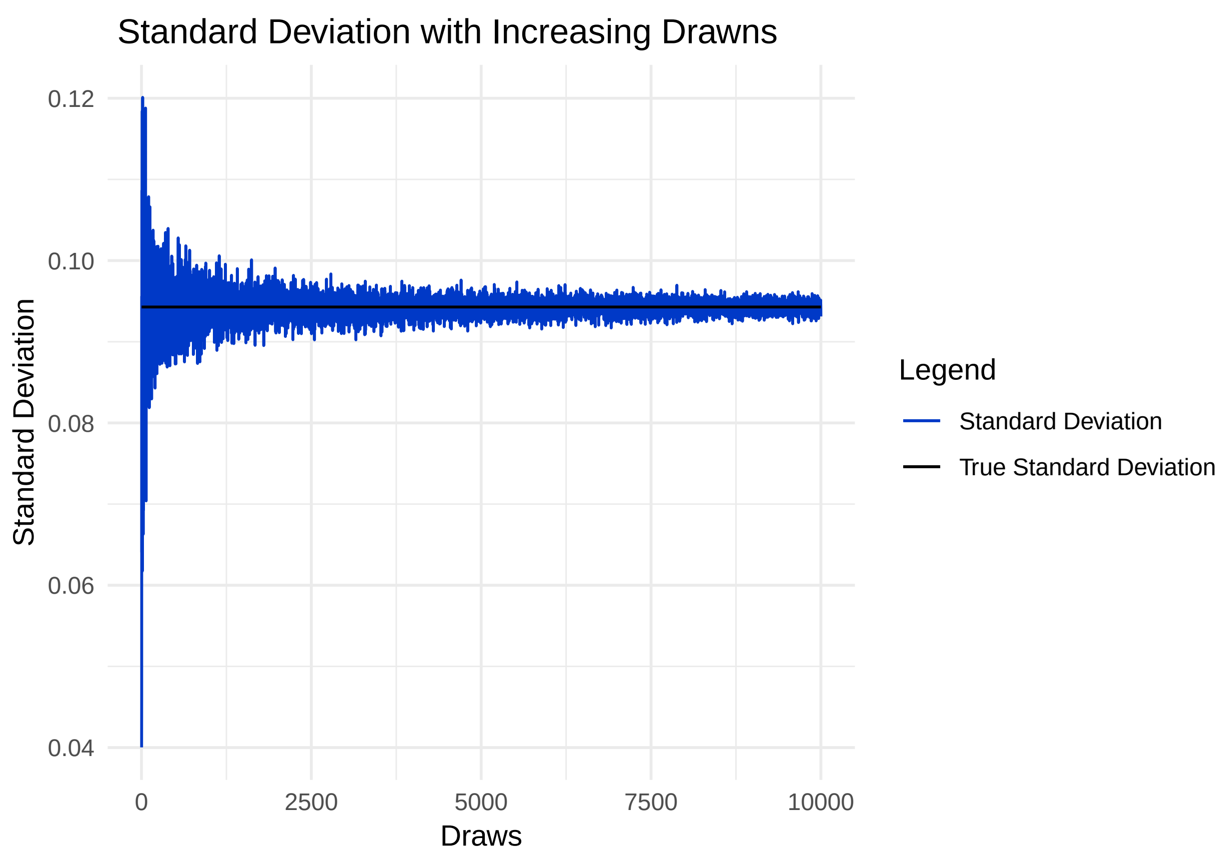
```r
mean_posterior = alpha_post / (alpha_post + beta_post)
sd_posterior = sqrt((alpha_post * beta_post) /
                  ((alpha_post + beta_post)^2 * (alpha_post + beta_post + 1)))
```

Therefore the mean of the prior is given by 0.6666667 and the standard deviation by 0.0942809.

Now we will create a function that calculates the mean and standard deviation for a given number of trials to plot it later on.

```r
get_stats = function(n, alpha, beta) {
  samples = rbeta(n, alpha, beta)
  return(c(count = n, sample_mean = mean(samples), sample_sd = sd(samples)))
}

df = data.frame(t(sapply(2:10000, get_stats, alpha_post, beta_post)))
```


Mean with Increasing Drawns
0.75
0.70
0.65
0.60
Mean
0
2500
5000
7500
10000
Draws
Legend
Standard Mean
True Mean


Standard Deviation with Increasing Drawns
0.12
0.10
0.08
0.06
0.04
Standard Deviation
0
2500
5000
7500
10000
Draws
Legend
Standard Deviation
True Standard Deviation

We can see that the posterior mean and standard deviation converge as the nimber of random draws grows.

1.2 $Pr(\theta < 0.4|y)$

The true probability is calculated by calling `pbeta(0.4, alpha_post, beta_post)` which is 0.0039727.

We will simulate by taking samples and counting how many of them are < 0.4.

```
################################################################################
# Exercise 1.b)
################################################################################

mean(rbeta(100000, alpha_post, beta_post) < 0.4)
```

```
## [1] 0.00388
```

As we can see both values are quite close to each other.

1.3 Log-Odds

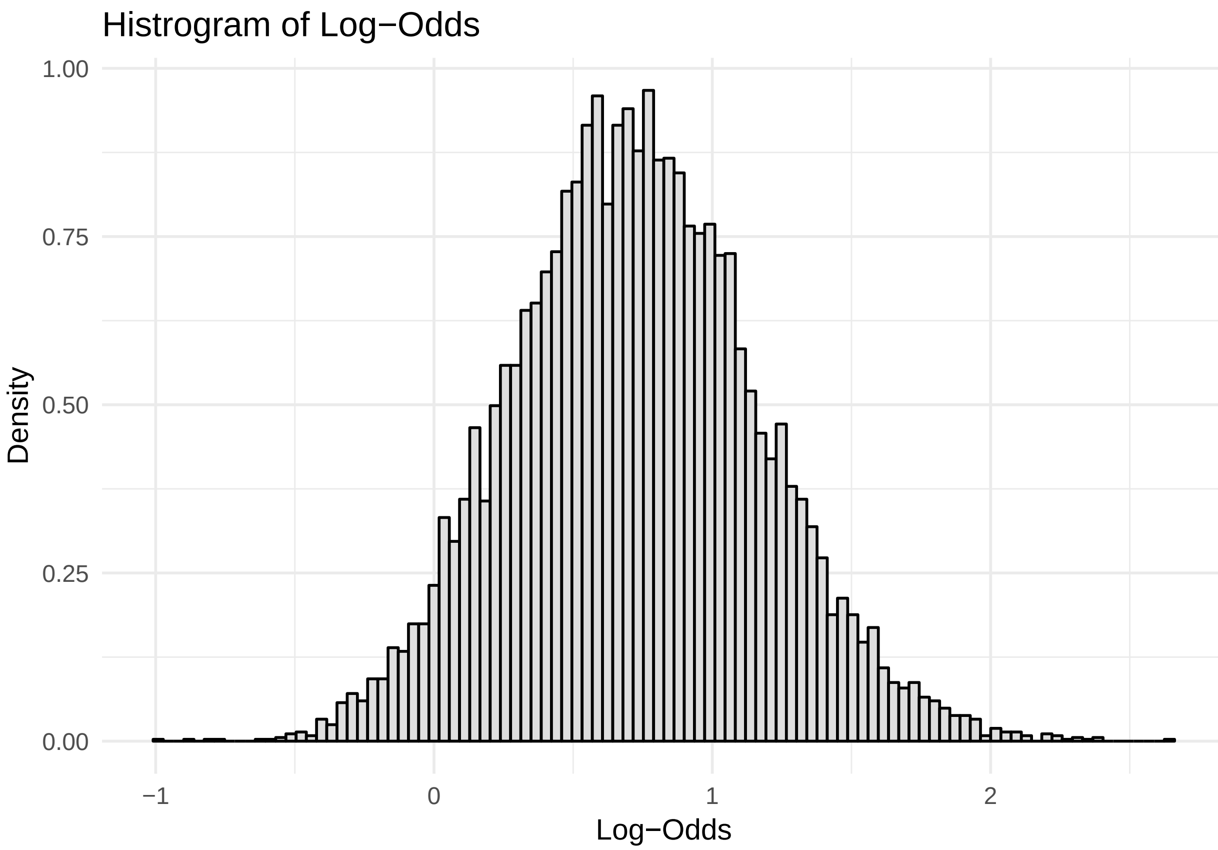
The log-odds are given by

$$\Phi = \log\left(\frac{\theta}{1-\theta}\right)$$

where θ are samples drawn from the posterior. We can therefore easily calculate the value by:

```
################################################################################
# Exercise 1.c)
################################################################################

draws = 10000

samples = rbeta(draws, alpha_post, beta_post)
phi = log(samples / (1 - samples))
```

The distribution looks as follows:



2 Log-Normal Distribution and the Gini Coefficient

Assume that you have asked 10 randomly selected persons about their monthly in- come (in thousands Swedish Krona) and obtained the following ten observations: 14, 25, 45, 25, 30, 33, 19, 50, 34 and 67. A common model for non-negative continuous variables is the log-normal distribution. The log-normal distribution log $\mathcal{N}(\mu, \sigma^2)$ has density function

$$p(y|\mu, \sigma^2) = \frac{1}{y\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2\sigma^2}(\log(y) - \mu)^2\right]$$

for $y > 0$, $\mu > 0$, and $\sigma^2 > 0$. The log-normal distribution is related to the normal distribution as follows: if $y \sim \mathcal{N}(\mu, \sigma^2)$ then $\log y \sim \mathcal{N}(\mu, \sigma^2)$. Let $y_1, ..., y_n|\mu, \sigma^2$, where $\mu = 3.5$ is assumed to be known but σ^2 unknown with non-informative prior $p(\sigma^2) \propto 1/\sigma^2$. The posterior for σ^2 is the $Inv - \chi^2(n, \tau^2)$ distribution where

$$\tau^2 = \frac{\sum_{i=1}^{n}(\log y_i - \mu)^2}{n}.$$

a) Simulate 10, 000 draws from the posterior of σ^2 (assuming $\mu = 3.5$) and compare with the theoretical $Inv - \chi^2(n, \tau^2)$ posterior distribution.

b) The most common measure of income inequality is the Gini coefficient, G, where $0 \leq G \leq 1$. $G = 0$ means a completely equal income distribution, whereas $G = 1$ means complete income inequality. See Wikipedia for more information. It can be shown that $G = 2\Phi(\sigma/\sqrt{n}) - 1$ when incomes follow a $\log \mathcal{N}(\mu, \sigma^2)$ distribution. $\Phi(z)$ is the cumulative distribution function (CDF) for the standard normal distribution with mean zero and unit variance. Use the posterior draws in a) to compute the posterior distribution of the Gini coefficient G for the current data set.

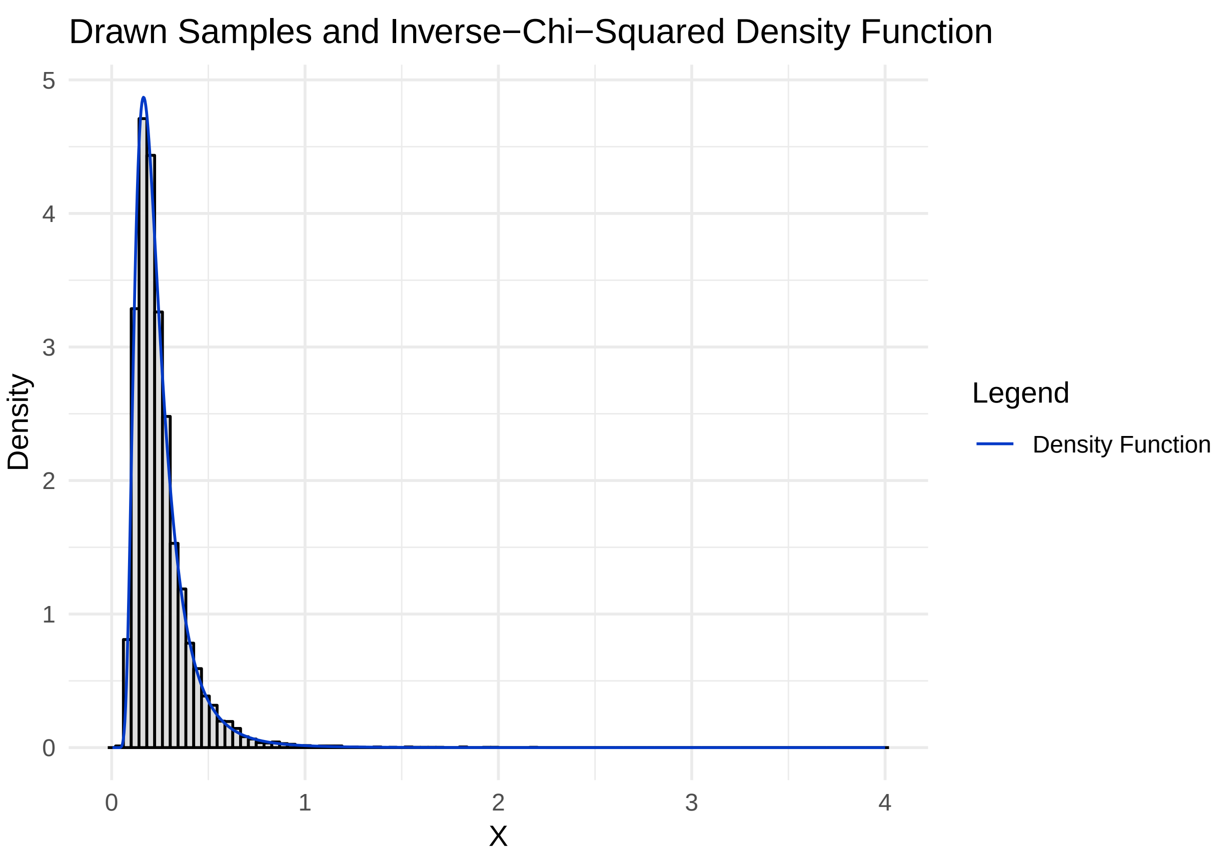
c) Use the posterior draws from b) to compute a 95% equal tail credible interval for G. An 95% equal tail interval (a, b) cuts off 2.5% percent of the posterior probability mass to the left of a, and 97.5% to the right of b. Also, do a kernel density estimate of the posterior of G using the `density` function in R with default settings, and use that kernel density estimate to compute a 95% Highest Posterior Density interval for G. Compare the two intervals.

2.1 Simulate Draws from the Posterior Distribution

We calculate τ^2, then we sample from `rinvchisq()` and compare the drawn samples to the desnity function given by `dinvchisq()`.

```
################################################################################
# Exercise 2.a)
################################################################################

obs = c(14, 25, 45, 25, 30, 33, 19, 50, 34, 67)
n = length(obs)
mu = 3.5
tau_sq = (sum((log(obs) - mu)^2)) / (n)

samples = rinvchisq(10000, n, tau_sq)
X = seq(from = 0, to = 4, length.out = 1000)
Y = dinvchisq(X, n, tau_sq)
```

We can see hat the drawn samples for σ^2 fit the $Inv - \chi^2(n, \tau^2)$.

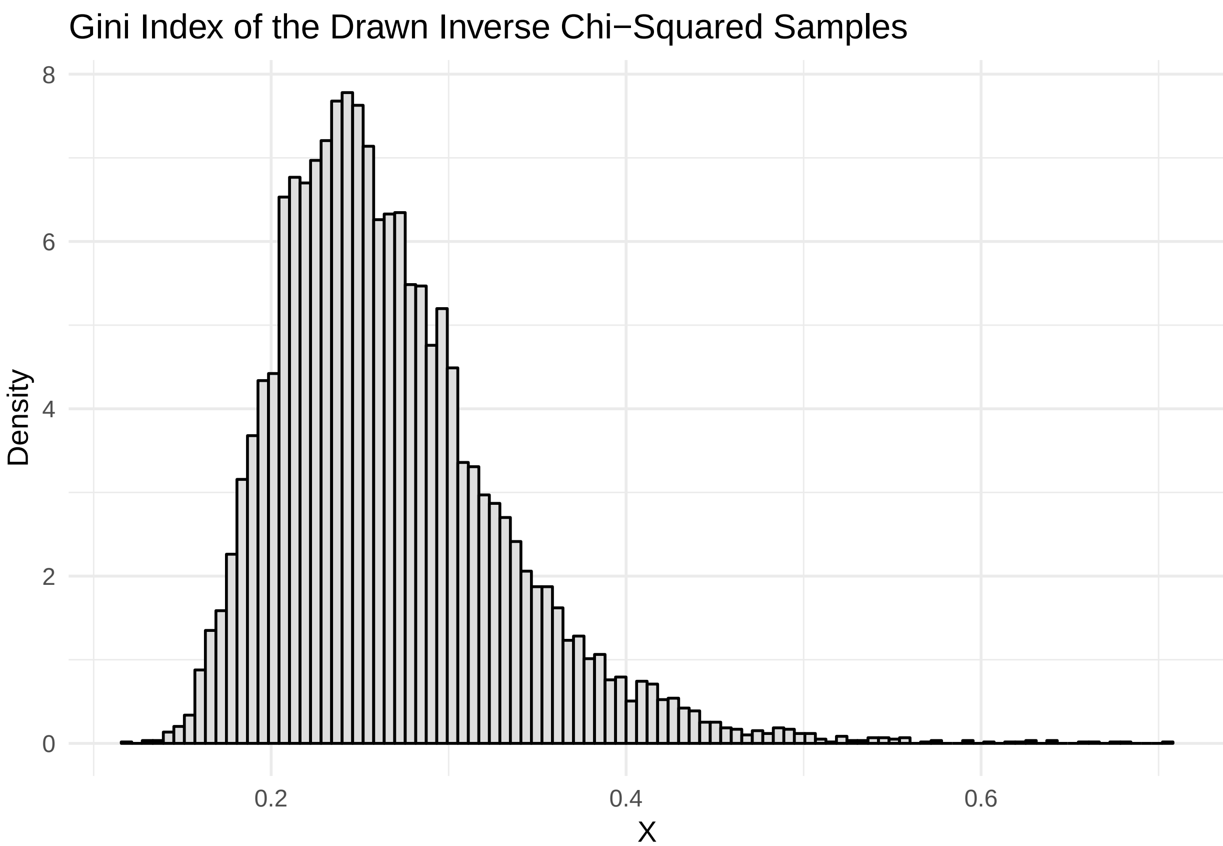


2.2 Gini-Index

The gini index is calculated by the following R code. As our samples represent σ^2, we have to include it in the square root.

```
################################################################################
# Exercise 2.b)
################################################################################

G = 2 * pnorm(sqrt(samples/2)) - 1
```

The histogram for the gini indeces looks like follows:

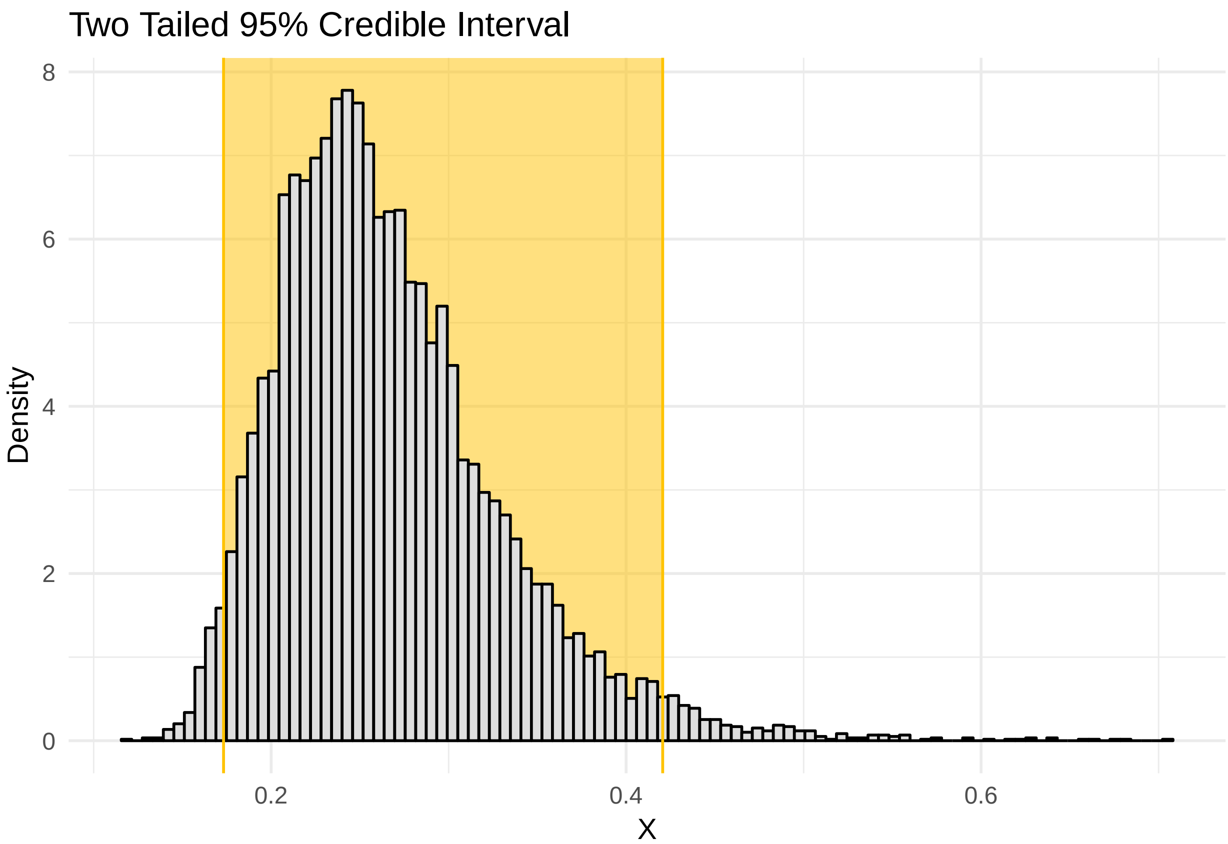


2.3 Credible and Density Interval

We us the `quantile()` function to receive the quantile values for the credible interval.

```
################################################################################
# Exercise 2.c)
################################################################################

quantiles = quantile(G, c(0.025, 0.975))
```

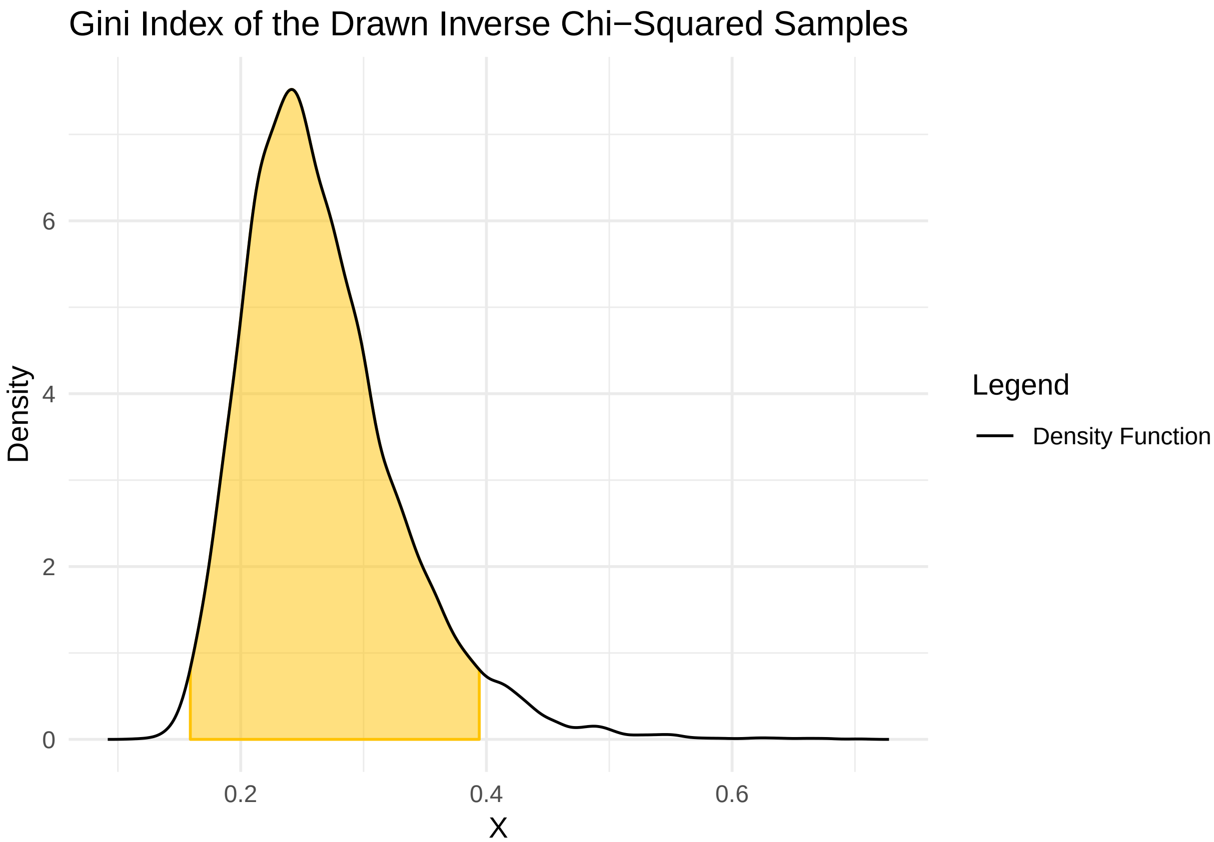
The following plot visualizes the credible interval:



For the density interval we first order our x and y values and then keep on adding more until we have reached more than 5 percent of the area (is faster then looking for the 95 percentile). We then save the index of that data point. The y value of this data point defines the threshold. We then filter for the values above the threshold, the left values all reside insde of the density interval. As we have just one interval in this case, it's enough to simply take the first and last value (of the unordered set after filtering) to visualize the density interval.

```
dg = density(G)

dg_unordered = data.frame(dg$x, dg$y)
dg_ordered = data.frame(dg$x[order(dg$y)], dg$y[order(dg$y)])
colnames(dg_ordered) = c("x_ord", "y_ord")

index = NaN

for (i in 1:nrow(dg_ordered)) {
  if (sum(dg_ordered$y_ord[1:i])/sum(dg_ordered$y_ord) > 0.05) {
    index = i
    break
  }
}

theta = dg_ordered[index,2]

selected = dg_unordered[dg_unordered$dg.y > theta,]

interval_a = selected[1,]
interval_b = selected[nrow(selected),]
```

The value of θ is 0.7881264. The interval is given by [0.1589535, 0.3941731].



3 Bayesian Inference

4 Source Code

```
knitr::opts_chunk$set(echo = TRUE)
library(ggplot2)
library(geoR)

################################################################################
# Exercise 1.a)
################################################################################

# Parameters
n = 20
s = 14
f = n - s

# Prior
alpha_z = 2
beta_z = 2
```

```r
# Posterior
alpha_post = alpha_z + s
beta_post = beta_z + f


mean_posterior = alpha_post / (alpha_post + beta_post)
sd_posterior = sqrt((alpha_post * beta_post) /
                  ((alpha_post + beta_post)^2 * (alpha_post + beta_post + 1)))


get_stats = function(n, alpha, beta) {
  samples = rbeta(n, alpha, beta)
  return(c(count = n, sample_mean = mean(samples), sample_sd = sd(samples)))
}

df = data.frame(t(sapply(2:10000, get_stats, alpha_post, beta_post)))


ggplot(df) +
  geom_line(aes(x = count, y = sample_mean, color = "Standard Mean")) +
  geom_line(aes(x = count, y = mean_posterior, color = "True Mean")) +
  labs(title = "Mean with Increasing Drawns", y = "Mean", x = "Draws") +
  scale_color_manual("Legend", values = c("#C70039", "#000000")) +
  theme_minimal()


ggplot(df) +
  geom_line(aes(x = count, y = sample_sd, colour = "Standard Deviation")) +
  geom_line(aes(x = count, y = sd_posterior, colour = "True Standard Deviation")) +
  labs(title = " Standard Deviation with Increasing Drawns",
       y = "Standard Deviation", x = "Draws") +
  scale_color_manual("Legend", values = c("#0039C7", "#000000")) +
  theme_minimal()


################################################################################
# Exercise 1.b)
################################################################################

mean(rbeta(100000, alpha_post, beta_post) < 0.4)


################################################################################
# Exercise 1.c)
################################################################################

draws = 10000

samples = rbeta(draws, alpha_post, beta_post)
phi = log(samples / (1 - samples))


phi = data.frame(phi)
```

```r
colnames(phi) = "phi"

ggplot(phi) +
  geom_histogram(aes(x = phi, y=..density..), color = "black",
                 fill = "#dedede", bins = sqrt(draws)) +
  labs(title = "Histrogram of Log-Odds",
  y = "Density",
  x = "Log-Odds", color = "Legend") +
  theme_minimal()


################################################################################
# Exercise 2.a)
################################################################################

obs = c(14, 25, 45, 25, 30, 33, 19, 50, 34, 67)
n = length(obs)
mu = 3.5
tau_sq = (sum((log(obs) - mu)^2)) / (n)

samples = rinvchisq(10000, n, tau_sq)
X = seq(from = 0, to = 4, length.out = 1000)
Y = dinvchisq(X, n, tau_sq)


df = data.frame(X, Y)
samples_df = data.frame(samples)

ggplot(df) +
  geom_histogram(data = samples_df, aes(x = samples, y=..density..),
                 bins = sqrt(nrow(samples_df)), color = "black",
                 fill = "#DEDEDE") +
  geom_line(aes(x = X, y = Y, color = "Density Function")) +
  labs(title = "Drawn Samples and Inverse-Chi-Squared Density Function",
       y = "Density", x = "X") +
  scale_color_manual("Legend", values = c("#0039C7", "#000000")) +
  theme_minimal()


################################################################################
# Exercise 2.b)
################################################################################

G = 2 * pnorm(sqrt(samples/2)) - 1


G_df = data.frame(G)

ggplot(G_df) +
  geom_histogram(aes(x = G, y=..density..),
                 bins = sqrt(nrow(G_df)), color = "black", fill = "#DEDEDE") +
  labs(title = "Gini Index of the Drawn Inverse Chi-Squared Samples",
       y = "Density", x = "X") +
```

```r
  scale_color_manual("Legend", values = c("#0039C7", "#000000")) +
  theme_minimal()


################################################################################
# Exercise 2.c)
################################################################################

quantiles = quantile(G, c(0.025, 0.975))


ggplot(G_df) +
  annotate("rect", xmin=quantiles[1], xmax=quantiles[2], ymin=0, ymax=Inf,
           alpha=0.5, fill="#FFC300") +
  geom_histogram(aes(x = G, y=..density..),
                 bins = sqrt(nrow(G_df)), color = "black", fill = "#DEDEDE") +
  geom_vline(xintercept = quantiles, colour = "#FFC300") +
  labs(title = "Two Tailed 95% Credible Interval",
       y = "Density", x = "X") +
  scale_color_manual("Legend", values = c("#0039C7", "#000000")) +
  theme_minimal()

dg = density(G)

dg_unordered = data.frame(dg$x, dg$y)
dg_ordered = data.frame(dg$x[order(dg$y)], dg$y[order(dg$y)])
colnames(dg_ordered) = c("x_ord", "y_ord")

index = NaN

for (i in 1:nrow(dg_ordered)) {
  if (sum(dg_ordered$y_ord[1:i])/sum(dg_ordered$y_ord) > 0.05) {
    index = i
    break
  }
}

theta = dg_ordered[index,2]

selected = dg_unordered[dg_unordered$dg.y > theta,]

interval_a = selected[1,]
interval_b = selected[nrow(selected),]


dg_df = data.frame(dg$x, dg$y)

ggplot(dg_df) +
  geom_ribbon(data = selected, aes(x = dg.x, ymin = 0, ymax = dg.y),
              alpha = 0.5, fill = "#FFC300", color = "#FFC300") +
  geom_line(aes(x = dg.x, y = dg.y, color = "Density Function")) +
  labs(title = "Gini Index of the Drawn Inverse Chi-Squared Samples",
       y = "Density", x = "X") +
```

```
  scale_color_manual("Legend", values = c("#000000", "#000000")) +
  theme_minimal()
```