Bayesian Learning - Computer Lab 1

Stefano Toffol (steto820) and Nahid Farazmand (nahfa911)

March 28, 2019

Q1. Bernoulli ... again

Let $y_1, ..., y_n \mid Bern(\theta)$ and assume that you have obtained a sample with s = 14 successes in n = 20 trials. Assume a $Beta(\alpha_0, \beta_0)$ prior for θ and let $\alpha_0 = \beta_0 = 2$.

a. Drawing and analysing random numbers from the posterior

Likelihood:

$$p(y \mid \theta) \sim \binom{n}{s} \theta^s (1-\theta)^{n-s}$$

Prior:

$$p(\theta) \sim \frac{\theta^{\alpha_0 - 1}(1-\theta)^{\beta_0 - 1}}{B(\alpha_0, \beta_0)}$$

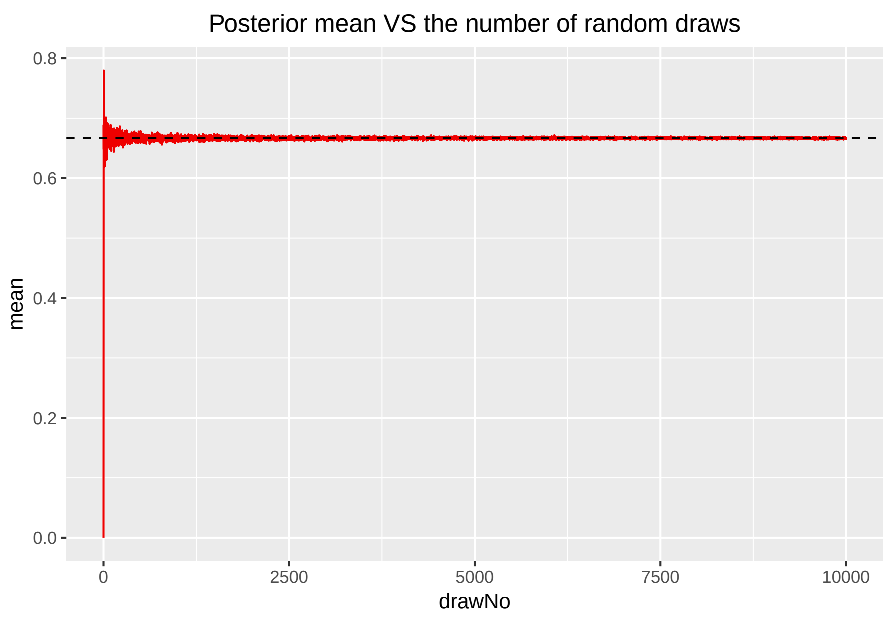
Posterior:

$$p(\theta \mid y) \propto \theta^{s+\alpha_0-1}(1-\theta)^{n-s+\beta_0-1}$$

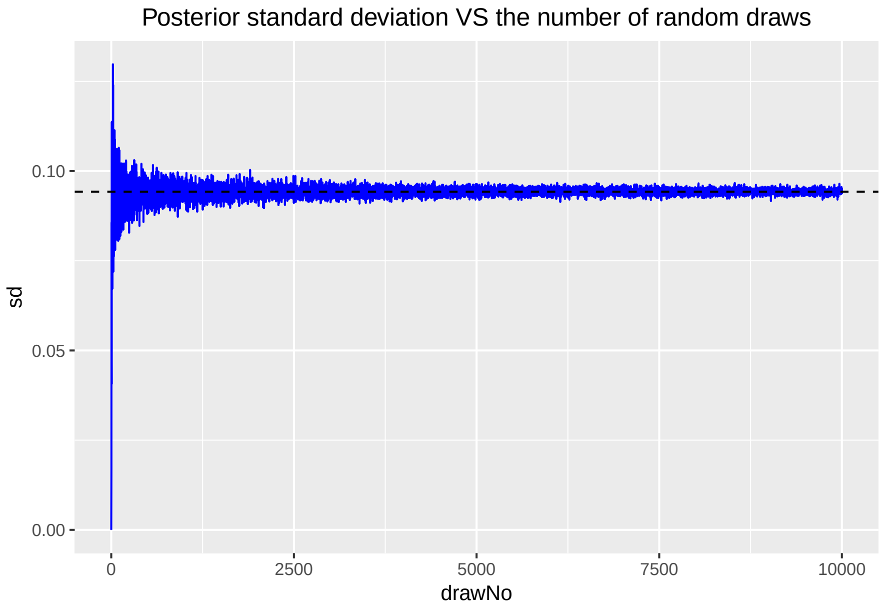
$$n - s = f$$

$$p(\theta \mid y) \sim Beta(\alpha_0 + s, \beta_0 + f) \sim Beta(16, 8)$$

Plot of posterior mean as the number of random draws grows



Plot of posterior standard deviation as the number of random draws grows



As we now based on the the formulas of mean and the variance of beta distribution, the true values for mean and standard deviation will be as bellow:

$$mean = \frac{\alpha_0}{\alpha_0 + \beta_0} = \frac{16}{16+8} = 0.\overline{6}$$

$$variance = \frac{\alpha_0 \beta_0}{(\alpha_0+\beta_0)^2 \cdot (\alpha_0+\beta_0+1)} = \frac{16 \cdot 8}{(16+8)^2 \cdot (16+8+1)} = \frac{2}{3^2 \cdot 5^2} = 0.00\overline{8}$$

$$sd = 0.094281$$

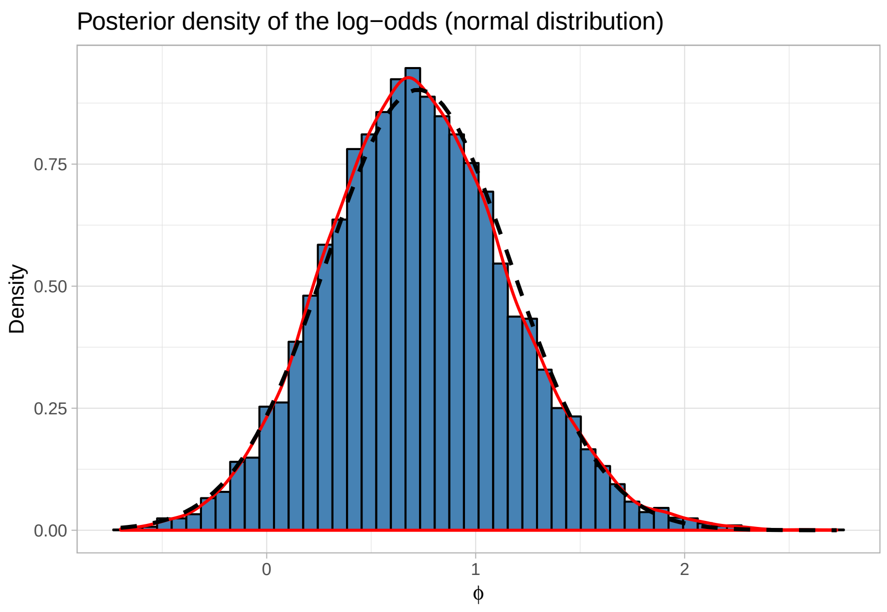
Comparing the theoretical values and the generated estimates that the posterior distribution actually converges towards the real values when the number of samples increase.

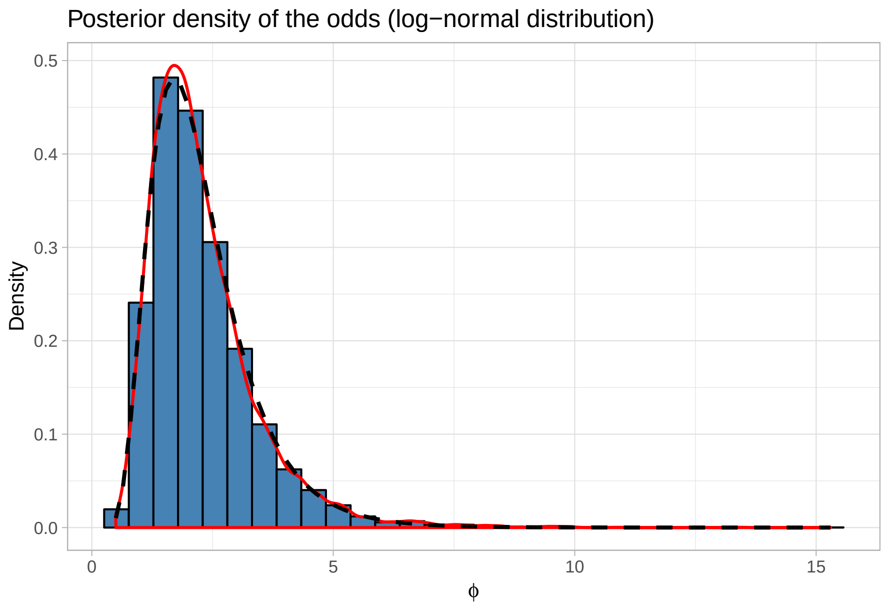
b. Computing the posterior probability $p(\theta < 0.4 \mid y)$ using simulation

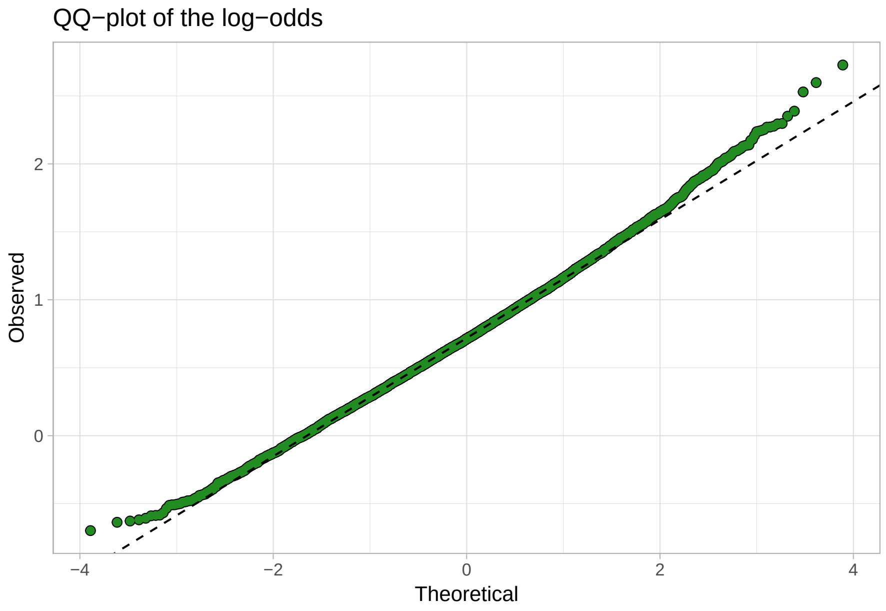
```
# Probability calculated using proportion of generated values
posterior_prob <- sum(post<0.4) / length(post)
# Probability according to theoretical distribution
theoretical_prob <- pbeta(0.4, 16, 8)
```

The posterior probability estimated from the random generation is equal to 0.004. On the other hand the theoretical probability obtained from the density of a *Beta* distribution is equal to 0.0003972. The two quantities are very similar, especially if we consider the small magnitude of the value.

c. Computing the posterior probability of the log-odds $\phi = \ln(\frac{\theta}{1-\theta})$ using simulation







Drawing the density of the generated log-odds, we observed how close the distribution was to the Gaussian one, with parameters equal to the ML estimations of the data (black dashed line). We tried to demonstrate it also analytically, but we were not able to find an explicit solution. Observing the odds (supposely log-normal) and the quantile plot, both plots hint towards this direction. However the actual distribution of the log-odds does not correspond to any of the known distributions, despite its similariity with the Gaussian one.

Q2. Log-normal distribution and the Gini coefficient

a. Simulate 10000 draws from the posterior of σ^2 (assuming $\mu = 3.5$) and compare with the theoretical $Inv - \chi^2(n, \tau^2)$ posterior distribution.

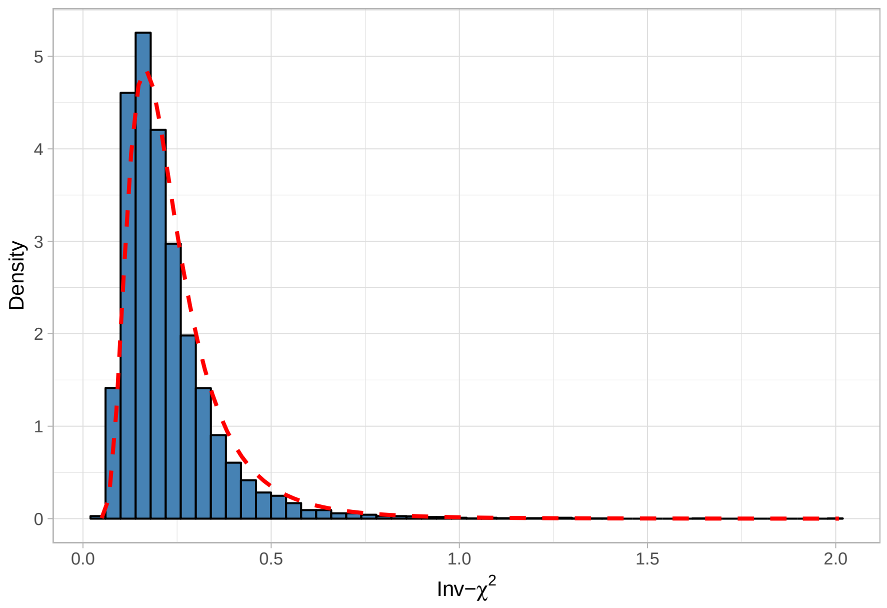
```
data <- c(14, 25, 45, 25, 30, 33, 19, 50, 34, 67)

n <- length(data)
mu <- 3.5
tau <- sum((log(data)-mu)^2)/n

numDraws <- 10000
# Random generation from a scaled inverse chisquare
rinvchisq <- function(draws, n, tau) {
  chi_square <- rchisq(draws, n)
  return( tau*(n-1)/chi_square )
}

# Density of a scaled inverse chisquare
dinvchisq <- function(data, df, tau) {
    return( (tau*df/2)^(df/2)/gamma(df/2) * exp(-df*tau/(2*data)) / data^(1+df/2) )
}

post_invchisq <- rinvchisq(numDraws, n, tau)
```



b. Log-normal distribution and the Gini coefficient

y = (14, 25, 45, 25, 30, 33, 19, 50, 34, 67).

Likelihood:

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

for $y > 0$, $\mu > 0$ and $\sigma^2 > 0$. Besides, $\mu = 3.5$, but σ^2 is unknown with non-informative prior $p(\sigma^2) \propto \frac{1}{\sigma^2}$ So the posterior for σ^2 is:

$$Inv - \chi^2(n, \tau^2)$$

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

and we know that the relation between log-normal distribution and normal distribution is as follow:

If $y \sim logN(\mu, \sigma^2)$ then $\log(y) \sim N(\mu, \sigma^2)$

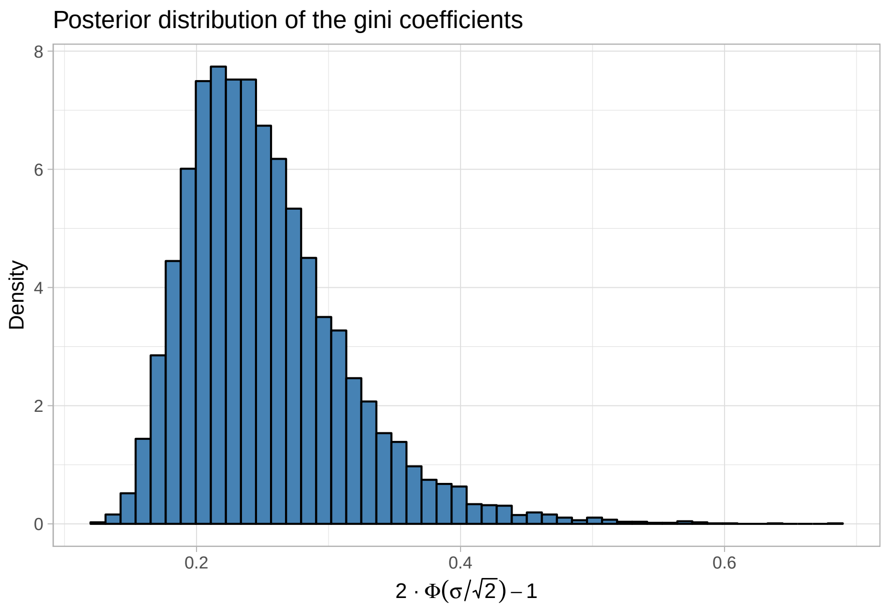
```r
# Compute the gini coefficient estimate from the data
gini_data <- sum(sapply(data, function(x) sum(abs(x-data)))) / (2*n*sum(data))

# Compute the distribution of the gini coefficient according to the log-normal model
gini_distr_lognorm <- function(x) {

  return( 2*pnorm(sqrt(x)/sqrt(2))-1 )

}

gini_distr <- gini_distr_lognorm(post_invchisq)
```



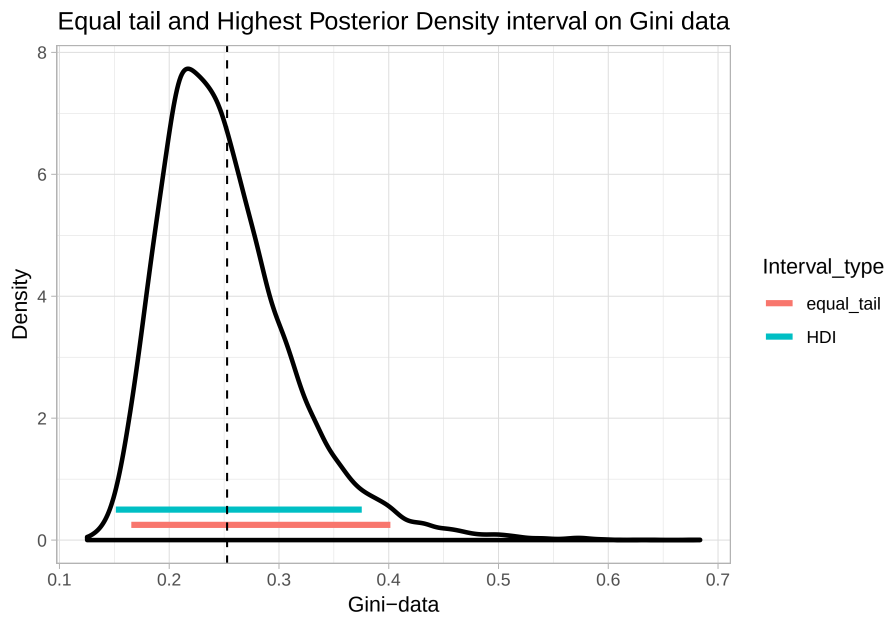
C. Credible intervals

```
equal_tail_interval <- c(quantile(gini_distr, 0.025), quantile(gini_distr, 0.975))

# We estimate the density of the gini coefficients
kernel_density_gini <- density(gini_distr)
df_kernel_density <- data.frame(x = kernel_density_gini$x,
                                y = kernel_density_gini$y)
# We order the data_frame according to the density
df_kernel_density <- df_kernel_density[order(df_kernel_density$y, decreasing = T),]
# We compute and save the cumulative sum of the density
df_kernel_density$cumsum <- cumsum(df_kernel_density$y)

# We get the threshold for the top 95% of the density
alpha_density <- tail(df_kernel_density$cumsum, 1) * 0.95
# We check which points are inside the interval
df_kernel_density$inside_interval <- df_kernel_density$cumsum < alpha_density
# The function is unimodal: just getting the extreme
# of the points included will be enough
HD_interval <- range(df_kernel_density$x[df_kernel_density$inside_interval])
```

In the chuck above we computed the two types of interval requested. Observing the plot below the results are quite similar, due to the small degree of skewness of the distribution. However as a general rule the *HQ* interval is more trustworthy and reliable than the other methods. In this case in particular the equal tail interval excludes some relativelly likely points on the left of the distribution.



Q3. Bayesian inference for the concentration parameter in the von Mises distribution

Prior:

$$p(\kappa) \sim Exp(\lambda = 1) = \lambda \cdot e^{-\lambda\kappa}$$

Likelihood:

$$p(y_i \mid \mu, \kappa) = \frac{\exp[\kappa \cdot cos(y_i - \mu)]}{2\pi I_0(\kappa)}$$

$$\prod_{i=1}^{n} \frac{\exp[\kappa \cdot cos(y_i - \mu)]}{2\pi I_0(\kappa)} = \left[\frac{1}{I_0(\kappa)}\right]^n \cdot \exp\left[\sum_{i=1}^{n} \kappa \cdot cos(y_i - \mu)\right]$$

Posterior:

$$p(\kappa \mid y_1, y_2, ..., y_n) \propto p(y_1, y_2, ..., y_n \mid \kappa) \cdot p(\kappa)$$

$$p(\kappa \mid y_1, y_2, ..., y_n) \propto \left[\frac{1}{I_0(\kappa)}\right]^n \cdot \exp\left[\sum_{i=1}^{n} \kappa \cdot cos(y_i - \mu) - \lambda\kappa\right]$$

We know that $\lambda = 1$ and $\mu = 2.39$; so the posterior distribution is:

$$p(\kappa \mid y_1, y_2, ..., y_n) \propto \left[\frac{1}{I_0(\kappa)}\right]^n \cdot \exp\left[\sum_{i=1}^{n} \kappa \cdot cos(y_i - 2.39) - \kappa\right]$$

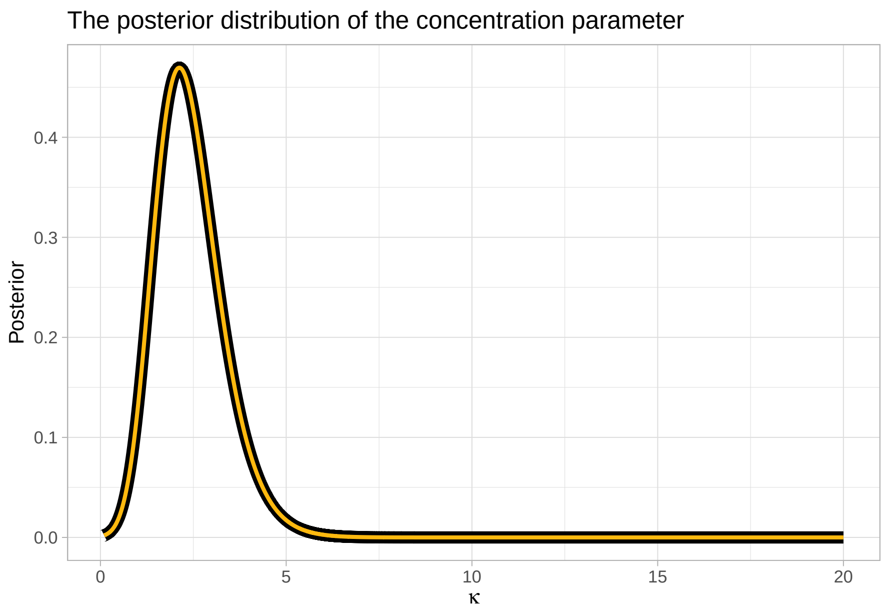
In order to plot the posterior distribution of the concentration parameter we decided to test it over an interval of possible values, starting from 0 and going until 20, with steps of 0.1. This decision was made after a few plotting attempts, since the probability mass is concentrated mainly between 0 and 5, the distribution appears unimodal and for values of $\kappa > 45$ the program cannot compute the results, returning `NaNs` (the Bessel function gives too big values).

The final result is the figure below. As we can see we plotted the posterior distribution both as the product of the likelihood and the prior distribution as well as the posterior distribution computed by hand ourselves. The two lines perfectly overlap, demonstrating the correctness of our computations. The black line corresponds to the product of the likelihood and the prior; the golden line corresponds to our analytical solution.
The distribution appears unimodal and with one mode at approximately 2.0.

```r
# Data (radians are the only necessary ones)
degrees <- c(40, 303, 326, 285, 296, 314, 20, 308, 299, 296)
radians <- c(-2.44, 2.14, 2.54, 1.83, 2.02, 2.33, -2.79, 2.23, 2.07, 2.02)

# Prior: exponential with rate=1
prior_k <- function(k, lambda = 1) dexp(k, lambda)
# Likelihoo: von Mises distribution with fixed mean (2.39)
likelihood_k <- function(k, mu = 2.39, data = radians) {
  n <- length(data)
  return( (2*pi*besselI(k, 0))^(-n) * exp(k * sum(cos(data-mu))) )
}
# Posterior distribution solved analytically above
posterior_k <- function(k, mu = 2.39, lambda = 1, data = radians) {
  n <- length(data)
  return( (2*pi*besselI(k, 0))^(-n) * exp(k * (sum(cos(data-mu))-1)) )
}

# Values on which we will test our function
k_seq <- seq(0.1, 20, by = 0.001)
```



Find the (approximate) posterior mode of from the information in a).

```r
posterior_mode <- df_plot$K[which.max(df_plot$Posterior)]
```

In order to get a more precise estimate of the mode of the distribution we find the maximum value of the posterior density among the values we have tested (between 0 and 20). The actual maximum is 2.125.

Appendix

```
knitr::opts_chunk$set(echo = F, message = F, error = F, warning = F,
                      fig.align='center', out.width="70%")


# Q1. Bernoulli ... again
library(ggplot2)
set.seed(2468)
out <- data.frame(drawNo = 0, mean = 0, sd = 0)
for(i in 2:10000){
  post <- rbeta(i, 16, 8)
  out[i,] <- c(i,mean(post),sd(post))
}

# posterior mean VS the number of random draws
ggplot(data = out) +
  geom_line(mapping = aes(x = drawNo,y = mean), color = 'red2') +
  geom_hline(yintercept = (16/(16+8)), lty = 2, col = "black") +
  ggtitle('Posterior mean VS the number of random draws') +
  theme(plot.title = element_text(hjust = 0.5))


# posterior standard deviation VS the number of random draws
ggplot(data = out) +
  geom_line(mapping = aes(x = drawNo,y = sd), color = 'blue') +
  geom_hline(yintercept = sqrt(16*8/((16+8)^2*25)), lty = 2, col = "black") +
  ggtitle('Posterior standard deviation VS the number of random draws') +
  theme(plot.title = element_text(hjust = 0.5))


# Probability calculated using proportion of generated values
posterior_prob <- sum(post<0.4) / length(post)
# Probability according to theoretical distribution
theoretical_prob <- pbeta(0.4, 16, 8)


# Transform the data into log-odds ( log(theta/(1-theta)) )
nDraws <- 10000
logOdds <- as.data.frame(log(post/(1-post)))
colnames(logOdds) <- c("x")

# Plot it as histogram; we can observe it's close to normality (black dashed line)
ggplot(logOdds, aes(x = x)) +
  geom_histogram(aes(y = ..density..), col = "black", fill = "steelblue", bins = 50) +
  geom_density(col = "red", size = 0.75) +
  stat_function(fun = dnorm, lty = 2, size = 1,
                args = list(mean = mean(logOdds[,1]), sd = sd(logOdds[,1]))) +
  labs(x = expression(phi), y = "Density") +
  ggtitle("Posterior density of the log-odds (normal distribution)") +
  theme_light()

# Plot the odds of the distribution (log-normal)
```

```
ggplot(logOdds, aes(x = exp(x))) +
  geom_histogram(aes(y = ..density..), col = "black", fill = "steelblue") +
  geom_density(col = "red", size = 0.75) +
  stat_function(fun = dlnorm, lty = 2, size = 1,
                args = list(meanlog = mean(logOdds[,1]), sdlog = sd(logOdds[,1]))) +
  ggtitle("Posterior density of the odds (log-normal distribution)") +
  labs(x = expression(phi), y = "Density") +
  theme_light()


# Function to plot the quantile-quantile plot
ggQQ <- function(vec, col = "forestgreen", title = "QQ-plot") {

  require(ggplot2)

  y <- quantile(vec[!is.na(vec)], c(0.25, 0.75))
  x <- qnorm(c(0.25, 0.75))
  slope <- diff(y)/diff(x)
  int <- y[1L] - slope * x[1L]

  d <- data.frame(resids = vec)

  ggplot(d, aes(sample = resids)) +
    stat_qq(color="black", shape=1, size=I(2)) +
    stat_qq(color = col) +
    geom_abline(slope = slope, intercept = int, lty = 2) +
    ggtitle(label = title) +
    labs(x = "Theoretical", y = "Observed") +
    theme_light()

}

ggQQ(logOdds[,1], title = "QQ-plot of the log-odds")


data <- c(14, 25, 45, 25, 30, 33, 19, 50, 34, 67)

n <- length(data)
mu <- 3.5
tau <- sum((log(data)-mu)^2)/n

numDraws <- 10000
# Random generation from a scaled inverse chisquare
rinvchisq <- function(draws, n, tau) {
  chi_square <- rchisq(draws, n)
  return( tau*(n-1)/chi_square )
}

# Density of a scaled inverse chisquare
dinvchisq <- function(data, df, tau) {
    return( (tau*df/2)^(df/2)/gamma(df/2) * exp(-df*tau/(2*data)) / data^(1+df/2) )
}
```

```r
post_invchisq <- rinvchisq(numDraws, n, tau)


df_post_invchisq <- as.data.frame(post_invchisq)
colnames(df_post_invchisq) <- "x"

# Histogram of the generated points (scaled inverse chisquare)
ggplot(df_post_invchisq, aes(x = x)) +
  geom_histogram(aes(y = ..density..), col = "black", fill = "steelblue", bins = 50) +
  stat_function(fun = dinvchisq, lty = 2, col = "red", size = 1,
                args = list(df = n, tau = tau)) +
  labs(x = expression(paste("Inv-", chi^2)), y = "Density") +
  theme_light()


# Compute the gini coefficient estimate from the data
gini_data <- sum(sapply(data, function(x) sum(abs(x-data)))) / (2*n*sum(data))

# Compute the distribution of the gini coefficient according to the log-normal model
gini_distr_lognorm <- function(x) {

  return( 2*pnorm(sqrt(x)/sqrt(2))-1 )

}

gini_distr <- gini_distr_lognorm(post_invchisq)


df_gini_distr <- as.data.frame(gini_distr)
colnames(df_gini_distr) <- "x"

ggplot(df_gini_distr, aes(x = x, y = ..density..)) +
  geom_histogram(aes(y = ..density..), col = "black",
                 fill = "steelblue", bins = 50) +
  labs(x = expression(paste("2 · ", Phi(sigma/sqrt(2)) -1 )), y = "Density") +
  ggtitle("Posterior distribution of the gini coefficients") +
  theme_light()


equal_tail_interval <- c(quantile(gini_distr, 0.025), quantile(gini_distr, 0.975))

# We estimate the density of the gini coefficients
kernel_density_gini <- density(gini_distr)
df_kernel_density <- data.frame(x = kernel_density_gini$x,
                                y = kernel_density_gini$y)
# We order the data_frame according to the density
df_kernel_density <- df_kernel_density[order(df_kernel_density$y, decreasing = T),]
# We compute and save the cumulative sum of the density
df_kernel_density$cumsum <- cumsum(df_kernel_density$y)

# We get the threshold for the top 95% of the density
alpha_density <- tail(df_kernel_density$cumsum, 1) * 0.95
# We check which points are inside the interval
```

```r
df_kernel_density$inside_interval <- df_kernel_density$cumsum < alpha_density
# The function is unimodal: just getting the extreme
# of the points included will be enough
HD_interval <- range(df_kernel_density$x[df_kernel_density$inside_interval])


data <- rbind(data.frame(x = equal_tail_interval, y = c(0.25, 0.25),
                         Interval_type = c('equal_tail','equal_tail')),
              data.frame(x = HD_interval, y = c(0.5, 0.5),
                         Interval_type =c('HDI','HDI')))

ggplot() +
  geom_density(aes(x = x), data = df_gini_distr, size = 1.1) +
  geom_line(data = data, mapping = aes(x = x, y = y, color = Interval_type),
            size = 1.5 )+
  labs(x = 'Gini-data', y = "Density") +
  geom_vline(aes(xintercept = mean(gini_distr)), lty = 2) +
  theme_light() +
  ggtitle('Equal tail and Highest Posterior Density interval on Gini data') +
  theme(plot.title = element_text(hjust = 0.5))

# To get the shade (p is the plot above, saved into the object p):
# df_density <- ggplot_build(p)$data[[1]]
# ind_shade <- findInterval(HD_interval, df_density$x)
# df_shade <- data.frame(x = df_density$x[ind_shade[1]:ind_shade[2]],
#                        ymin = df_density$ymin[ind_shade[1]:ind_shade[2]],
#                        ymax = df_density$ymax[ind_shade[1]:ind_shade[2]])
#
# p + geom_ribbon(aes(x = x, ymin = ymin, ymax = ymax), data = df_shade,
#                 col = "grey75", alpha = 0.5) +
#   geom_density(aes(x = x), data = df_gini_distr, size = 1.1) +


# Data (radians are the only necessary ones)
degrees <- c(40, 303, 326, 285, 296, 314, 20, 308, 299, 296)
radians <- c(-2.44, 2.14, 2.54, 1.83, 2.02, 2.33, -2.79, 2.23, 2.07, 2.02)

# Prior: exponential with rate=1
prior_k <- function(k, lambda = 1) dexp(k, lambda)
# Likelihoo: von Mises distribution with fixed mean (2.39)
likelihood_k <- function(k, mu = 2.39, data = radians) {
  n <- length(data)
  return( (2*pi*besselI(k, 0))^(-n) * exp(k * sum(cos(data-mu))) )
}
# Posterior distribution solved analytically above
posterior_k <- function(k, mu = 2.39, lambda = 1, data = radians) {
  n <- length(data)
  return( (2*pi*besselI(k, 0))^(-n) * exp(k * (sum(cos(data-mu))-1)) )
}

# Values on which we will test our function
k_seq <- seq(0.1, 20, by = 0.001)
```

```r
df_plot <- data.frame(K = k_seq, Posterior = likelihood_k(k_seq)*prior_k(k_seq),
                      Posterior2 = posterior_k(k_seq))
sum_unnor <- sum(df_plot$Posterior2)*0.001
df_plot$Posterior <- df_plot$Posterior/sum_unnor
df_plot$Posterior2 <- df_plot$Posterior2/sum_unnor

ggplot(df_plot, aes(x = K, y = Posterior)) +
  geom_line(size = 3, col = "black") +
  geom_line(aes(y = Posterior2), size = 1, col = "darkgoldenrod1") +
  labs(x = expression(kappa)) +
  ggtitle('The posterior distribution of the concentration parameter') +
  theme(plot.title = element_text(hjust = 0.5)) +
  theme_light()


posterior_mode <- df_plot$K[which.max(df_plot$Posterior)]
```