Computer Lab 1

phiho267 & zijfe244

29-3-2019

Contents

1. Bernoulli . . . again.

Let $y_1, ..., y_n \mid \theta \sim \text{Bern}(\theta)$, and assume that you have obtained a sample with s = 14 successes in n = 20 trials. Assume a Beta(α_0, β_0) prior for θ and let $\alpha_0 = \beta_0 = 2$.

(a) Draw random numbers from the posterior

$\theta \mid y \sim Beta(\alpha_0 + s, \beta_0 + f)$, $y = (y_1, ..., y_n)$, and verify graphically that the posterior mean and standard deviation converges to the true values as the number of random draws grows large.

- *True mean & standard deviation*

First of all do we have to calcualte the true mean and true standard deviation of the Beta distribution.

$$E[\theta] = \frac{\alpha}{\alpha + \beta}$$

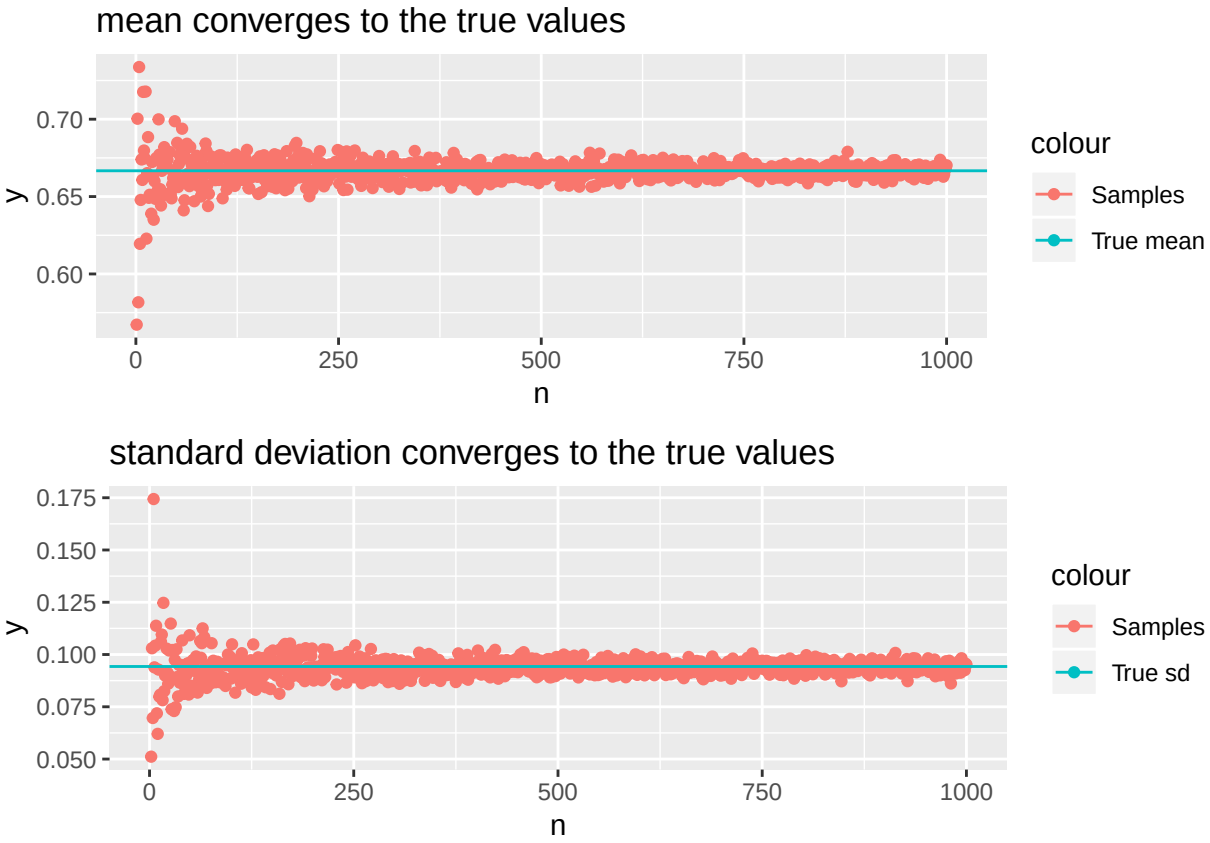
$$Var[\theta] = \frac{\alpha\beta}{(\alpha + \beta)^2(\alpha + \beta + 1)}$$

```
# calculate the true mean and standard deviation
true_mean = alpha_new/(alpha_new + beta_new)
true_var = (alpha_new*beta_new) / ((alpha_new + beta_new)^2 * (alpha_new + beta_new +1))
true_sd = sqrt(true_var)
```

- *Simulated mean & standard deviation*

```
# calculate how the mean changes with increasing number of n - until 10000 values
set.seed(123456)
iterations = 1:1000
mean_of_n = c()
for (i in 1:length(iterations)) {
  mean_of_n[i] = mean(rbeta(n = i, shape1 = alpha_new, shape2 = beta_new))
}

# calculate how the sd changes with increasing number of n
sd_of_n = c()
for (i in 1:length(iterations)) {
  sd_of_n[i] = sd(rbeta(n = i, shape1 = alpha_new, shape2 = beta_new))
}
```

- *Plot of:*



(b) Use simulation

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

```
set.seed(123456)
# simulation to compute the posterior probability - of beta theata < 0.4
nDraws_b = 10000
sample_b = rbeta(n = nDraws_b, shape1 = 16, shape2 = 8)
# if value in sample is smaler than 0, than 1 else 0
sample_b_binary = ifelse(sample_b < 0.4, 1, 0)
prob_sample_b = sum(sample_b_binary)/nDraws_b

# exact value theta < 0.4
exact_value_1b  = pbeta(q = 0.4, shape1 = alpha_new, shape2 = beta_new)
exact_value_1b = round(exact_value_1b, 4)
```

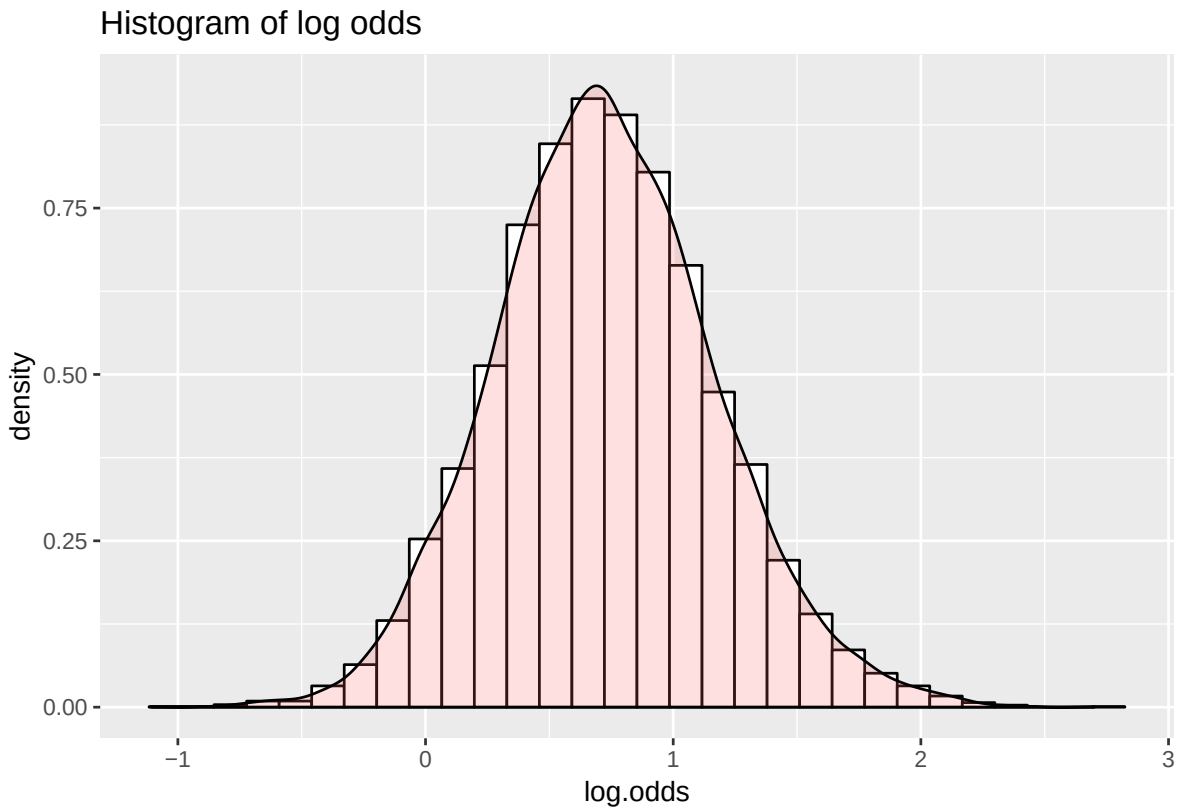
Expected.value	Simulated.vaule
0.004	0.0043

The posterior probabilities from simulation and theoritical formula (exact value) are quite similar.

(c) Compute the posterior distribution of the log-odds

$\phi = log(\frac{\theta}{1-\theta})$(nDraws = 10000). [Hint: hist() and density() might come in handy]

```
set.seed(12345)
nDraws_1c = 10000
sample_1c = rbeta(n = nDraws_b, shape1 = 16, shape2 = 8)
log_odds = log(sample_1c/(1-sample_1c))
```



2. Log-normal distribution and the Gini coefficient.

Assume that you have asked 10 randomly selected persons about their monthly income (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

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

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

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

(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.

For this exercise did we decided to compare the simulated values with the theoretical mean and variance.

We calculate the mean and variance of the scaled inverse chi-squared distribution distribution as follows:

$$E(\theta) = \frac{n\tau^2}{n-2}\ for\ n > 2$$

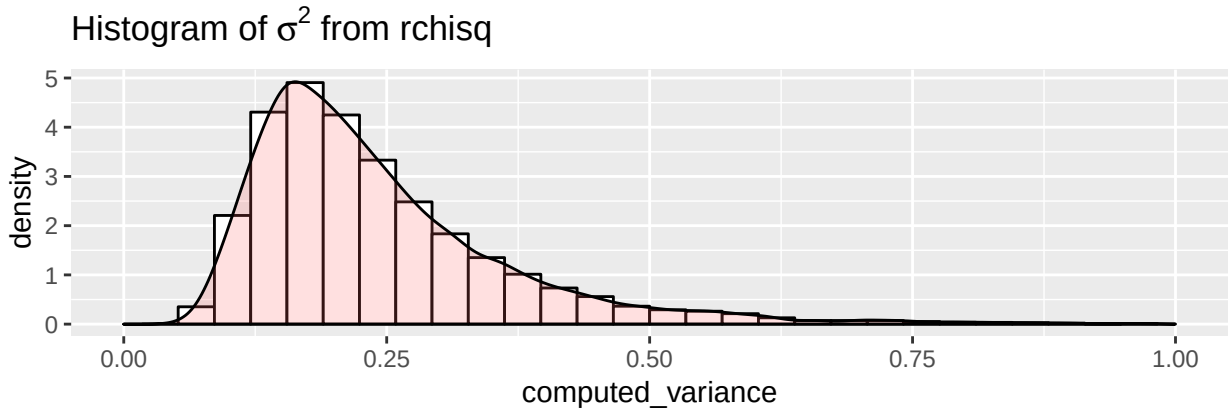
$$Var[\theta] = \frac{2n^2\tau^4}{(n-2)^2(n-4)}\ for\ n > 4$$

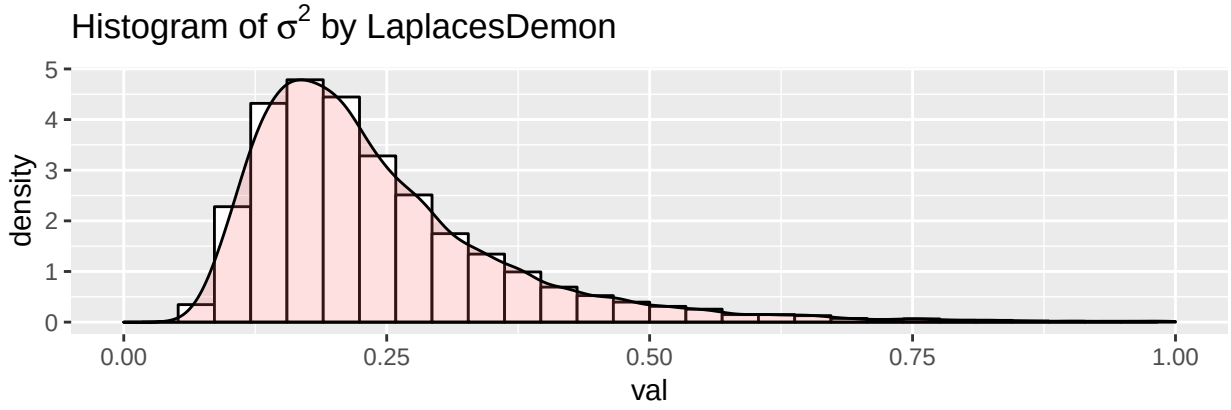
- *True mean and variance*

```
# --- theoreticle value
# calculate tau with given formula
tau = function(y,mu = 3.5){
  n = length(y)
  result = sum((log(y)-mu)^2)/n
  return(result)
}

tau_2 = tau(observations)

# theoreticle mean
theoreticle_mean = n_observations * tau_2/(n_observations-2)

# theoreticle var
theoreticle_var = 2 * n_observations^2 * tau_2^2 /
  ((n_observations -2)^2 * (n_observations - 4))
```

- *Simulated mean and variance*

```
#-------------- Method 1: Theory from the lecture
# --- simulate values from rchisq with formula in lecture 3
set.seed(12345)
nDraws_2b = 10000
computed_variance = c()
for (i in 1:nDraws_2b) {
  X = rchisq(1, n_observations)
  computed_variance[i] =  (n_observations) * tau_2 / X
}

# --- simulate values by LaplacesDemon
val <- LaplacesDemon::rinvchisq(nDraws_2b,df=10,tau_2)
```





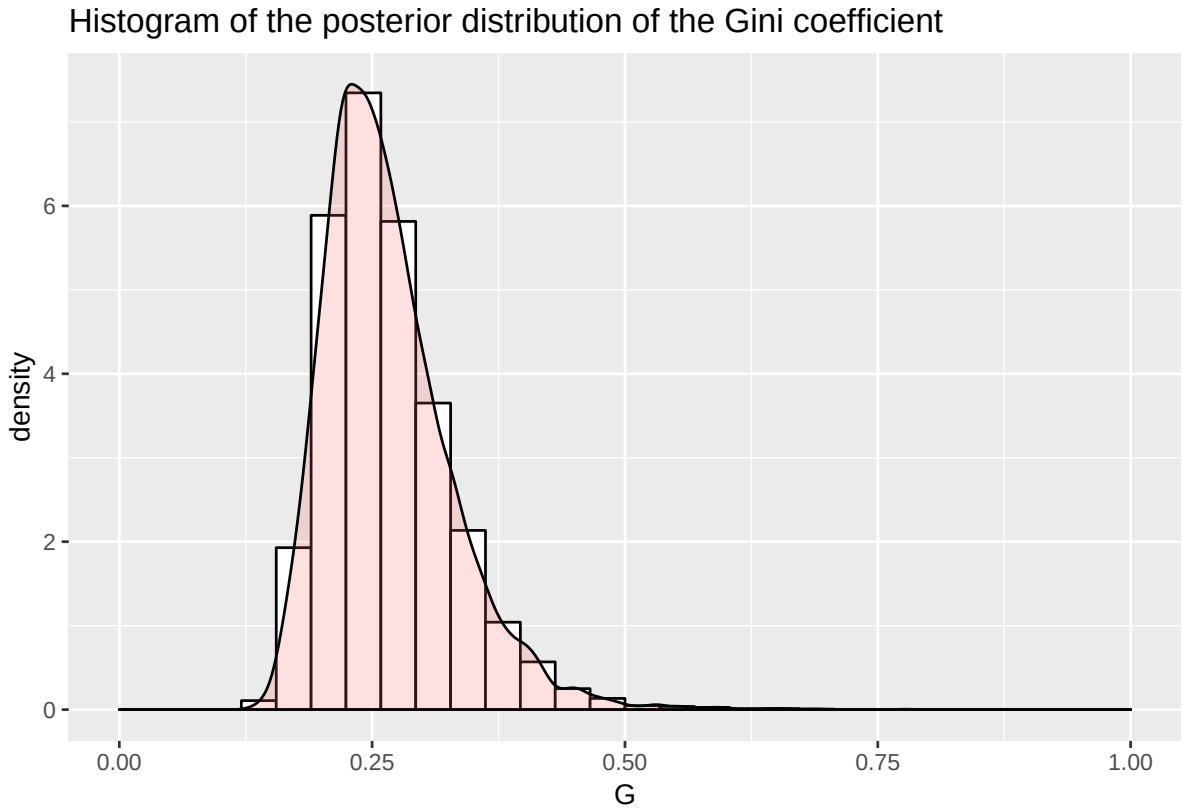
	mean	variance
theoretical.value	0.2473497	0.0203940
simulated.rchisq	0.2467178	0.0207204
simulated.rinvchisq	0.2449486	0.0195968

(b) Gini coefficient

The most common measure of income inequality is the Gini coefficient, G, where $0 \leqslant G \leqslant 1$. 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 $G = 2\Phi(\frac{\sigma}{\sqrt{2}}) - 1$ when incomes follow a log $\mathcal{N}(\mu, \sigma)$ 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.

- *Calculate G:*

```
# Gini coefficient G
# phi(z) - CDF for standard normal distribution - mu = 0, unit variance
# posterior drawn in a - computed_variance
# compute  posterior distribution of the Gini coefficient
G = 2 * pnorm(sqrt(computed_variance)/sqrt(2), mean = 0, sd = 1) -1
```



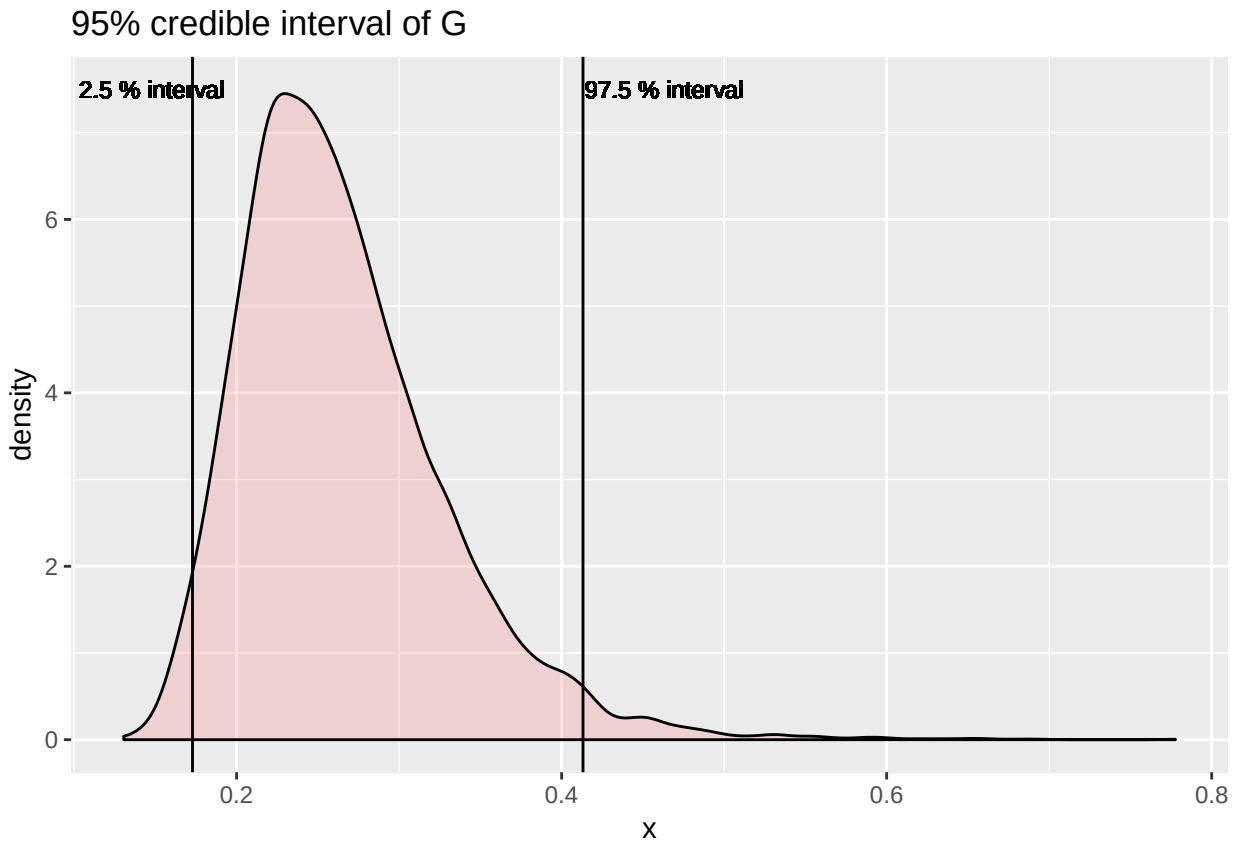
The most part of the distribution of Gini coefficient is located around [0.12, 0.37], which means the income inequality is not serious.

(c) tail credible interval for G

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.

- 95% Credible Interval

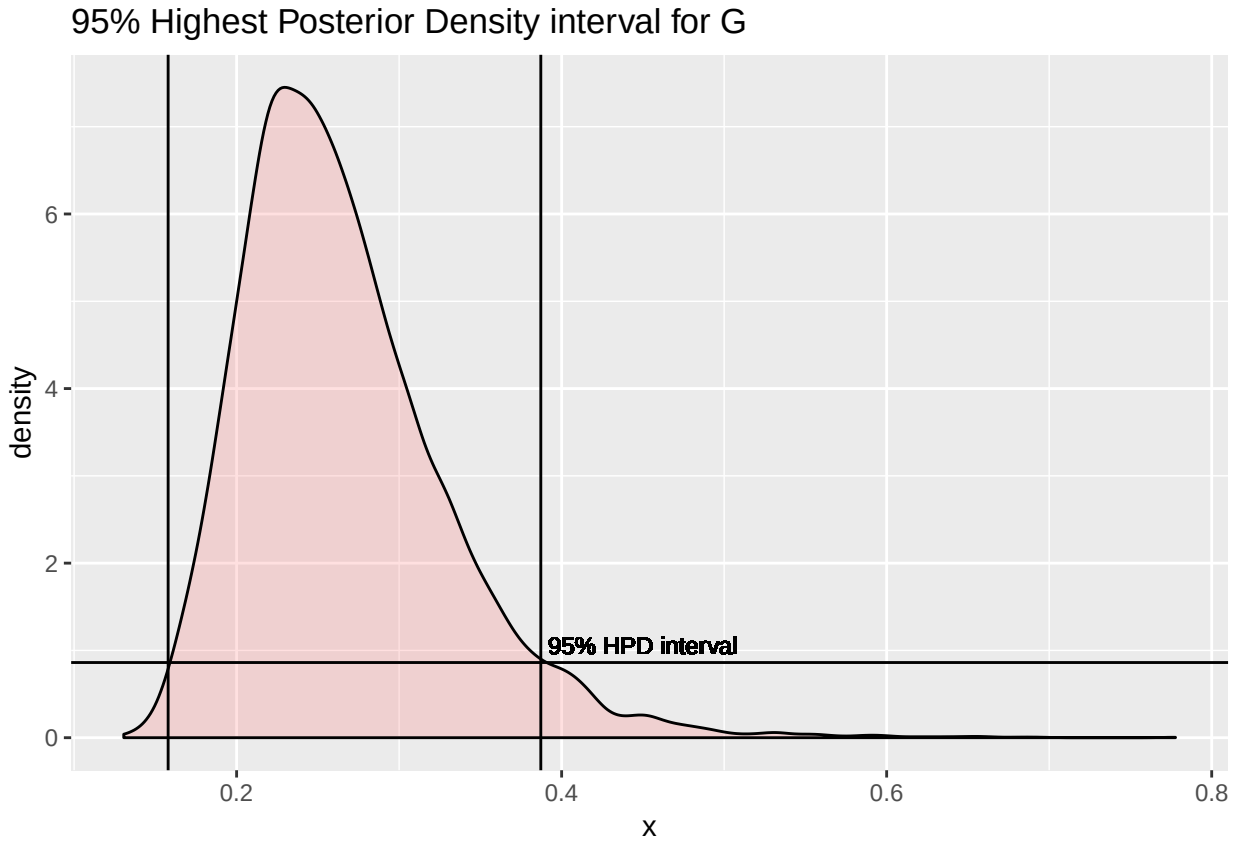
```
# Method 1:
# calculate the 95% credible(confidence) interval
a =0.025
b = 0.975
CI_0025 = quantile(G, probs = c(a,b))[1]
CI_095 = quantile(G, probs = c(a,b))[2]
```



The x values of G are 0.1728546 and 0.413131.

- 95% Highest Posterior Density interval

```
# Method 2:
# idea 2 - h-line
density_G = density(G)
density_y = density_G$y
density_x = density_G$x
density_df = data.frame(nr = 1:length(density_x),
                        density_x = density_x,
                        density_y = density_y)
density_df_ordered = density_df[order(density_y, decreasing = TRUE),]
density_df_ordered$cumsum_y = cumsum(density_df_ordered$density_y)
density_df_ordered$cumsum_y_proportional_percent = density_df_ordered$cumsum_y/sum(density_y)*100


density_df_ordered$in_ci = density_df_ordered$cumsum_y_proportional_percent <= 95

#data frame with just true values
density_df_ordered_trueci = density_df_ordered[(density_df_ordered$in_ci == TRUE),]
HPD = density_df_ordered_trueci[nrow(density_df_ordered_trueci),]


# calculate the x values  wiht hdi
x_vals = hdi(G)
```



The x values of G are 0.1578584 and 0.3871752.

3. Bayesian inference

for the concentration parameter in the von Mises distribution. This exercise is concerned with directional data. The point is to show you that the posterior distribution for somewhat weird models can be obtained by plotting it over a grid of values. The data points are observed wind directions at a given location on ten different days. The data are recorded in degrees:

$$(40, 303, 326, 285, 296, 314, 20, 308, 299, 296),$$

where North is located at zero degrees. To fit with Wikipedias description of probability distributions for circular data we convert the data into radian $-\pi \leqslant y \leqslant \pi$. The 10 observations in radians are

$$(-2.44, 2.14, 2.54, 1.83, 2.02, 2.33, -2.79, 2.23, 2.07, 2.02).$$

Assume that these data points are independent observations following the von Mises distribution

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

where $I_0(\kappa)$ is the modified Bessel function of the first kind of order zero. The parameter $\mu(-\pi \leqslant y \leqslant \pi)$ is the mean direction and $\kappa > 0$ is called the concentration parameter. Large κ gives a small variance around μ, and vice versa. Assume that μ is known to be 2.39. Let $\kappa \sim Exponential(\lambda = 1)$ a priori, where λ is the rate parameter of the exponential distribution (so that the mean is $1/\lambda$).

(a) Plot the posterior distribution

of κ for the wind direction data over a fine grid of κ values.

(b) Find the (approximate) posterior mode

of κ from the information in a).

To plot the posterior distribution do we have to calculate the posterior first:

- Likelihood - Mises distribution

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

Later on we do not care about the constant and remove $1/(2\pi)^n$.

- priror - Exponential distribution

$$\begin{aligned}
&\lambda \cdot \exp[-\lambda x] \\
&= \exp[-\kappa],
\end{aligned}$$

- posteriror - likelihood * prior

$$\begin{aligned}
&\frac{1}{I_o(\kappa)^n} \cdot \exp[\kappa \cdot \sum_{i=1}^{n} \cos(y_i - \mu)] \cdot \exp[-\kappa] \\
&= \frac{1}{I_o(\kappa)^n} \cdot \exp[\kappa \cdot \sum_{i=1}^{n} \cos(y_i - \mu) - \kappa]
\end{aligned}$$

```
# calcualte the posterior
result_3a = exp(k*sum(cos(radians-mu_3a))-k)/besselI(x = k, nu=0)^n_radians
```

The mode has the κ value of 2.12.

Appendix

```
knitr::opts_chunk$set(echo = TRUE)
rm(list=ls())
# libraries for this lab
library(ggplot2)
library(gridExtra)
#install.packages('latex2exp')
library(latex2exp)
#install.packages("HDInterval")
library(HDInterval)
library(coda)
#install.packages("LaplacesDemon")
library(LaplacesDemon)
# clean environment
rm(list=ls())
# calcualte true beta mean and variacne
# initialize given values
alpha0 = 2
beta0 = 2
n = 20
s = 14
f = n - s
alpha_new = alpha0 + s
beta_new = beta0 + f
# calculate the true mean and standard deviation
true_mean = alpha_new/(alpha_new + beta_new)
true_var = (alpha_new*beta_new) / ((alpha_new + beta_new)^2 * (alpha_new + beta_new +1))
true_sd = sqrt(true_var)
# calculate how the mean changes with increasing number of n - until 10000 values
set.seed(123456)
iterations = 1:1000
mean_of_n = c()
for (i in 1:length(iterations)) {
  mean_of_n[i] = mean(rbeta(n = i, shape1 = alpha_new, shape2 = beta_new))
}

# calculate how the sd changes with increasing number of n
sd_of_n = c()
for (i in 1:length(iterations)) {
  sd_of_n[i] = sd(rbeta(n = i, shape1 = alpha_new, shape2 = beta_new))
}
# create plot data
plot_data1.a = data.frame(x = iterations,
                          true_mean = true_mean,
                          true_sd = true_sd)

# create converges plot for mean and sd
mean_converges = ggplot(data = plot_data1.a, aes(x = plot_data1.a$x)) +
  geom_point(aes(x = x, y = mean_of_n, colour = "Samples")) +
  geom_hline(aes(yintercept = true_mean, colour = "True mean")) +
  ggtitle("mean converges to the true values") + xlab("n") + ylab("y")
```

```r
sd_converges = ggplot(data = plot_data1.a, aes(x = plot_data1.a$x)) +
  geom_point(aes(x = x, y = sd_of_n, colour = "Samples"))+
  geom_hline(aes(yintercept = true_sd,colour = "True sd")) +
  ggtitle("standard deviation converges to the true values") + xlab("n") + ylab("y")

grid.arrange(mean_converges, sd_converges, nrow = 2)

set.seed(123456)
# simulation to compute the posterior probability - of beta theata < 0.4
nDraws_b = 10000
sample_b = rbeta(n = nDraws_b, shape1 = 16, shape2 = 8)
# if value in sample is smaler than 0, than 1 else 0
sample_b_binary = ifelse(sample_b < 0.4, 1, 0)
prob_sample_b = sum(sample_b_binary)/nDraws_b

# exact value theta < 0.4
exact_value_1b  = pbeta(q = 0.4, shape1 = alpha_new, shape2 = beta_new)
exact_value_1b = round(exact_value_1b, 4)

# result table
result_1b_data = data.frame("Expected value" = exact_value_1b,
                            "Simulated vaule" = prob_sample_b)
knitr::kable(result_1b_data)
set.seed(12345)
nDraws_1c = 10000
sample_1c = rbeta(n = nDraws_b, shape1 = 16, shape2 = 8)
log_odds = log(sample_1c/(1-sample_1c))

# Visualization
# Histogram + Density

# basic plot example
# hist(log_odds, probability = TRUE)
# lines(density(log_odds)) # run all lines at the same time to create the plot
# create ggplot data
plot_data_1c = data.frame("Draw" = 1:nDraws_1c,
                          "log-odds" = log_odds)

ggplot(data = plot_data_1c, aes(x=log.odds)) +
 geom_histogram(aes(y=..density..),
                colour="black",
                fill="white",
                bins=30)+
  geom_density(alpha=.2, fill="#FF6666") +
  ggtitle("Histogram of log odds")
# given values
observations = c(14, 25, 45, 25, 30, 33, 19, 50, 34, 67)
n_observations = length(observations)
nDraws_2a = 10000
mu = 3.5
# --- theoreticle value
# calculate tau with given formula
tau = function(y,mu = 3.5){
```

```
  n = length(y)
  result = sum((log(y)-mu)^2)/n
  return(result)
}

tau_2 = tau(observations)

# theoreticle mean
theoreticle_mean = n_observations * tau_2/(n_observations-2)

# theoreticle var
theoreticle_var = 2 * n_observations^2 * tau_2^2 /
  ((n_observations -2)^2 * (n_observations - 4))

#--------------- Method 1: Theory from the lecture
# --- simulate values from rchisq with formula in lecture 3
set.seed(12345)
nDraws_2b = 10000
computed_variance = c()
for (i in 1:nDraws_2b) {
  X = rchisq(1, n_observations)
  computed_variance[i] =  (n_observations) * tau_2 / X
}

# --- simulate values by LaplacesDemon
val <- LaplacesDemon::rinvchisq(nDraws_2b,df=10,tau_2)
data_2a = data.frame(computed_variance, val)

p2a1=ggplot(data_2a,aes(x=computed_variance))+
 geom_histogram(aes(y=..density..),
                colour="black",
                fill="white",
               bins = 30)+
  geom_density(alpha=.2, fill="#FF6666") +
  ggtitle(TeX("Histogram of $\\sigma^2$ from rchisq"))+
  scale_x_continuous(limits = c(0,1))

p2a2=ggplot(data_2a,aes(x=val))+
 geom_histogram(aes(y=..density..),
                colour="black",
                fill="white",
               bins = 30)+
  geom_density(alpha=.2, fill="#FF6666") +
  ggtitle(TeX("Histogram of $\\sigma^2$ by LaplacesDemon"))+
  scale_x_continuous(limits = c(0,1))

grid.arrange(p2a1,p2a2, nrow = 2)


result_2a_data = data.frame("theoretical value" = c(theoreticle_mean,theoreticle_var),
                            "simulated rchisq" = c(mean(computed_variance), var(computed_variance)),
"simulated rinvchisq" = c(mean(val),var(val)))
```

```r
rownames(result_2a_data) = c("mean", "variance")
knitr::kable(t(result_2a_data))
# Gini coefficient G
# phi(z) - CDF for standard normal distribution - mu = 0, unit variance
# posterior drawn in a - computed_variance
# compute  posterior distribution of the Gini coefficient
G = 2 * pnorm(sqrt(computed_variance)/sqrt(2), mean = 0, sd = 1) -1
# create histogramm
#hist(G)

# plot data
plot_data2.b = data.frame("G" = G,
                          "nr" = 1:length(G))
# Histogram
ggplot(data = plot_data2.b,aes(x=G)) +
  ggtitle("Histogram of the posterior distribution of the Gini coefficient") +
   geom_histogram(aes(y=..density..),
                  colour="black",
                  fill="white",
                  bins=30)+
  geom_density(alpha=.2, fill="#FF6666")+
  xlim(c(0,1))
# Method 1:
# calculate the 95% credible(confidence) interval
a =0.025
b = 0.975
CI_0025 = quantile(G, probs = c(a,b))[1]
CI_095 = quantile(G, probs = c(a,b))[2]

# create plot data

########### test
#plot_data2.c = plot_data2.b
#plot_data2.c$ci095 = ifelse(G <= CI_0025 | G >= CI_095, TRUE, FALSE) # binary if G - value is in confi
########### test

# plot density
ggplot(data = plot_data2.b, aes(x = G)) +
  ggtitle("95% credible interval of G") +
  geom_density(alpha=.2, fill="#FF6666") +
  geom_vline(aes(xintercept = CI_095)) +
  geom_vline(aes(xintercept = CI_0025))+
  geom_text(aes(label ="97.5 % interval", x =  CI_095 + 0.05, y = 7.5),size = 3 )+
  geom_text(aes(label ="2.5 % interval", x =  CI_0025 -0.025, y = 7.5),size = 3 ) +
  xlab("x")
# Highest Posterior Density (HPD)

#-----------------------------------------------
#This thinking is not correct - take a value from the left & right with max value of function as highes
#-----------------------------------------------

# idea 1 - thinking problem it has to be h-line
density_G = density(G)
```

```
density_y = density_G$y
density_x = density_G$x
density_df = data.frame(density_x = density_x,
                        density_y = density_y)
density_y_sum = sum(density_y)
density_y_max =max(density_y)
density_y_max_index = which.max(density_y)
density_y[density_y_max_index]/density_y_sum*100

ci_95 = 91
ci_volum_start = density_y[density_y_max_index]/density_y_sum*100 # volumen at the higest point
ci_volum = ci_volum_start
x_left = (density_y_max_index-1):1
x_right = (density_y_max_index+1):length(density_y)
x_left_point = (density_y_max_index-1)
x_right_point = (density_y_max_index+1)
x_left_volum = 0
x_right_volum = 0
while(ci_volum <= ci_95) {
  # calculate the volumen to the left and the right
  x_left_volum = density_y[x_left_point]/density_y_sum*100
  x_right_volum = density_y[x_right_point]/density_y_sum*100
  # move the point to the left and the right
  x_left_point = x_left_point -1 # move to the left
  x_right_point = x_right_point +1 # move to the right
  # update the confidence interval
  ci_volum = ci_volum + x_left_volum + x_right_volum
}
# Method 2:
# idea 2 - h-line
density_G = density(G)
density_y = density_G$y
density_x = density_G$x
density_df = data.frame(nr = 1:length(density_x),
                        density_x = density_x,
                        density_y = density_y)
density_df_ordered = density_df[order(density_y, decreasing = TRUE),]
density_df_ordered$cumsum_y = cumsum(density_df_ordered$density_y)
density_df_ordered$cumsum_y_proportional_percent = density_df_ordered$cumsum_y/sum(density_y)*100


density_df_ordered$in_ci = density_df_ordered$cumsum_y_proportional_percent <= 95

#data frame with just true values
density_df_ordered_trueci = density_df_ordered[(density_df_ordered$in_ci == TRUE),]
HPD = density_df_ordered_trueci[nrow(density_df_ordered_trueci),]


# calculate the x values  wiht hdi
x_vals = hdi(G)
# library(dplyr)
# library(ggplot2)
#
```

```r
# dens <- density(G)
#
# data <- tibble(x = dens$x, y = dens$y) %>%
#   mutate(variable = case_when(
#     (x >= interval[1] & x <= interval[2]) ~ "On",
#     ( x<=interval[1]) ~ "Off",(x >=interval[2])~"onOff",
#     TRUE ~ NA_character_))
# #> Warning: package 'bindrcpp' was built under R version 3.4.4
#
# ggplot(data, aes(x, y)) + geom_line(size=2) +
#   geom_area(data = filter(data, variable == 'On'), fill = 'lightslateblue') +
#   geom_area(data = filter(data, variable == 'Off'), fill = 'orange')+
#   geom_area(data = filter(data, variable == 'onOff'), fill = 'orange')
ggplot(data = plot_data2.b, aes(x = G)) +
  ggtitle("95% Highest Posterior Density interval for G") +
  geom_density(alpha=.2, fill="#FF6666") +
  geom_vline(aes(xintercept = x_vals[1])) +
  geom_vline(aes(xintercept = x_vals[2])) +
  geom_hline(aes(yintercept = HPD$density_y)) +
  geom_text(aes(label ="95% HPD interval", x =  0.45, y = HPD$density_y+0.2),size = 3 ) +
  xlab("x")
# given values
degrees = c(40, 303, 326, 285, 296, 314, 20, 308, 299, 296)
n_degrees = length(degrees)
radians = c(-2.44, 2.14, 2.54, 1.83, 2.02, 2.33, -2.79, 2.23, 2.07, 2.02)
n_radians = length(radians)
mu_3a = 2.39
# grid of k values
k = seq(from = 0.1,to = 8, by = 0.01) # k>0
# calcualte the posterior
result_3a = exp(k*sum(cos(radians-mu_3a))-k)/besselI(x = k, nu=0)^n_radians

# create plot data
plot_data3.a = data.frame("k" = k,
                          "posterior" = result_3a)
# histogram
# ggplot(data = plot_data3.a, aes(x=result_3a_v2)) +
#   geom_histogram()

plot_3a = ggplot(data = plot_data3.a, aes(x=k, y=result_3a)) +
  geom_line() +
  ylab("Posterior value") + ggtitle(TeX("Posterior distribution of $\\kappa$")) +
  xlab(TeX("Grid of $\\kappa$ values"))

# find the mode
#k_max = max(result_3a)

k_max_index = which.max(result_3a)

plot_3a + geom_vline(aes(xintercept = k[k_max_index]))

#cat("posterior mode :",k[k_max_index])
```

```
NormalNonInfoPrior<-function(NDraws,Data){

#############################################################################################
# PURPOSE:  Generates samples from the joint posterior distribution of the parameters in the
#
#       MODEL: x1,....xn iid Normal(mu,sigma^2) model.
#       PRIOR: p(mu,sigma^2) propto 1/sigma^2
#
#
# INPUT:    NDraw:      (Scalar)    The number of posterior draws
#       Data:       (n-by-1)    Data vector of n observations
#
# OUTPUT:   PostDraws:  (NDraws-by-2)   Matrix of posterior draws.
#                       First column holds draws of mu
#                       Second column holds draws of sigma^2
#
# AUTHOR:   Mattias Villani, Sveriges Riksbank and Stockholm University.
#       E-mail: mattias.villani@riksbank.se.
#
# DATE:     2005-04-13
#
#############################################################################################


Datamean<-mean(Data)
s2<-var(Data)
n<-length(Data)
PostDraws=matrix(0,NDraws,2)
PostDraws[,2]<-((n-1)*s2)/rchisq(NDraws,n-1)
PostDraws[,1]<-Datamean+rnorm(NDraws,0,1)*sqrt(PostDraws[,2]/n)

return(PostDraws)
}

Data<-rnorm(100,5,10)           # Sampling 100 observations from the N(5,10) density##
PostDraws<-NormalNonInfoPrior(1000,Data) # Generating 1000 draws from the joint posterior density of mu
hist(PostDraws[,1])             # Plotting the histogram of mu-draws

#########################################################################################################
#################  Example of conjugate prior inference of the multinomial model ################
#########################################################################################################

####### Defining a function that simulates from a Dirichlet distribution
SimDirichlet <- function(nIter,param){
    nCat <- length(param)
    thetaDraws <- matrix(0,nIter,nCat) # Matrix where the posterior draws are stored
    for (j in 1:nCat){
        thetaDraws[,j] <- rgamma(nIter,param[j],1)
    }
    for (i in 1:nIter){
        thetaDraws[i,] = thetaDraws[i,]/sum(thetaDraws[i,]) # Diving every column of ThetaDraws by the
    }
    return(thetaDraws)
```

```
}

###########   Setting up data and prior  #################
y <- c(36,87,77) # Data
alpha <- c(1,1,1) # Dirichlet prior hyperparameters
nIter <- 10000 # Number of posterior draws

###########   Posterior sampling from Dirichlet  #################
thetaDraws <- SimDirichlet(nIter,y + alpha)

################ Computing Summary statistics from the posterior sample ##################
mean(thetaDraws[,1])
mean(thetaDraws[,2])
mean(thetaDraws[,3])

sqrt(var(thetaDraws[,1]))
sqrt(var(thetaDraws[,2]))
sqrt(var(thetaDraws[,3]))

sum(thetaDraws[,2]>thetaDraws[,3])/nIter # p(theta2>theta3|Data)

# Plots histograms of the posterior draws
plot.new() # Opens a new graphical window
par(mfrow = c(2,2)) # Splits the graphical window in four parts (2-by-2 structure)
hist(thetaDraws[,1],25) # Plots the histogram of theta[,1] in the upper left subgraph
hist(thetaDraws[,2],25)
hist(thetaDraws[,3],25)


PostAndPredIIDNormalNonInfoPrior<-function(Sample,PopStd,Niter){

#############################################################################################
#
# PURPOSE:  Sample from posterior and predictive distribution in the normal iid model
#       with a non-informative (flat) prior. Population standard deviation is known.
#
# INPUT:    Sample  (n-by-1)    Vector with sample observations
#       PopStd  (scalar)    Standard deviation in the population
#       Niter   (scalar)    Number of draws from posterior and predictive distributions.
#
# OUTPUT:   (Niter-by-2)        Matrix of draws. From posterior in first column.
#                   From predictive distribution in second column.
#
# AUTHOR:   Mattias Villani, Sveriges Riksbank and Stockholm University.
#           E-mail: mattias.villani@riksbank.se
#
# FIRST VER.:   2007-02-04
# THIS VER.:    2007-02-04
#
#############################################################################################

# Initialization
PostSampleTheta=matrix(0,Niter,1)
```

```
PredSampleYtilde=matrix(0,Niter,1)

# Initial computations
n<-length(Sample)
SampleMean<-mean(Sample)                                    # ybar
SampleStd<-PopStd/sqrt(n)                                   # Sigma/sqrt(n)

for (i in 1:Niter){
  PostSampleTheta[i]=SampleMean+rnorm(1,0,1)*SampleStd      # Theta~N[ybar,Sigma/sqrt(n)]
  PredSampleYtilde[i]=PostSampleTheta[i]+rnorm(1,0,1)*PopStd # yhat~N(Theta,Sigma)
}

return(cbind(PostSampleTheta,PredSampleYtilde))
}
```