

# Computer Lab 1

*Phillip Hölscher*

*29.3.2019*

## Contents

<b>1. Bernoulli ... again.</b>	<b>2</b>
(a) Draw random numbers from the posterior . . . . .	2
(b) Use simulation . . . . .	3
(c) Compute the posterior distribution of the log-odds . . . . .	3
<b>2. Log-normal distribution and the Gini coefficient.</b>	<b>4</b>
(a) Simulate 10,000 draws . . . . .	4
(b) Gini coefficient . . . . .	5
(c) tail credible interval for G . . . . .	6
<b>3. Bayesian inference</b>	<b>9</b>
(a) Plot the posterior distribution . . . . .	9
(b) Find the (approximate) posterior mode . . . . .	10
<b>Appendix</b>	<b>12</b>

## 1. Bernoulli ... again.

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

### (a) Draw random numbers from the posterior

$\theta \mid y \sim \text{Beta}(\alpha_0 + s, \beta_0 + f)$ ,  $y = (y_1, \dots, y_n)$ , 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 calculate the true mean and true standard deviation of the Beta distribution.

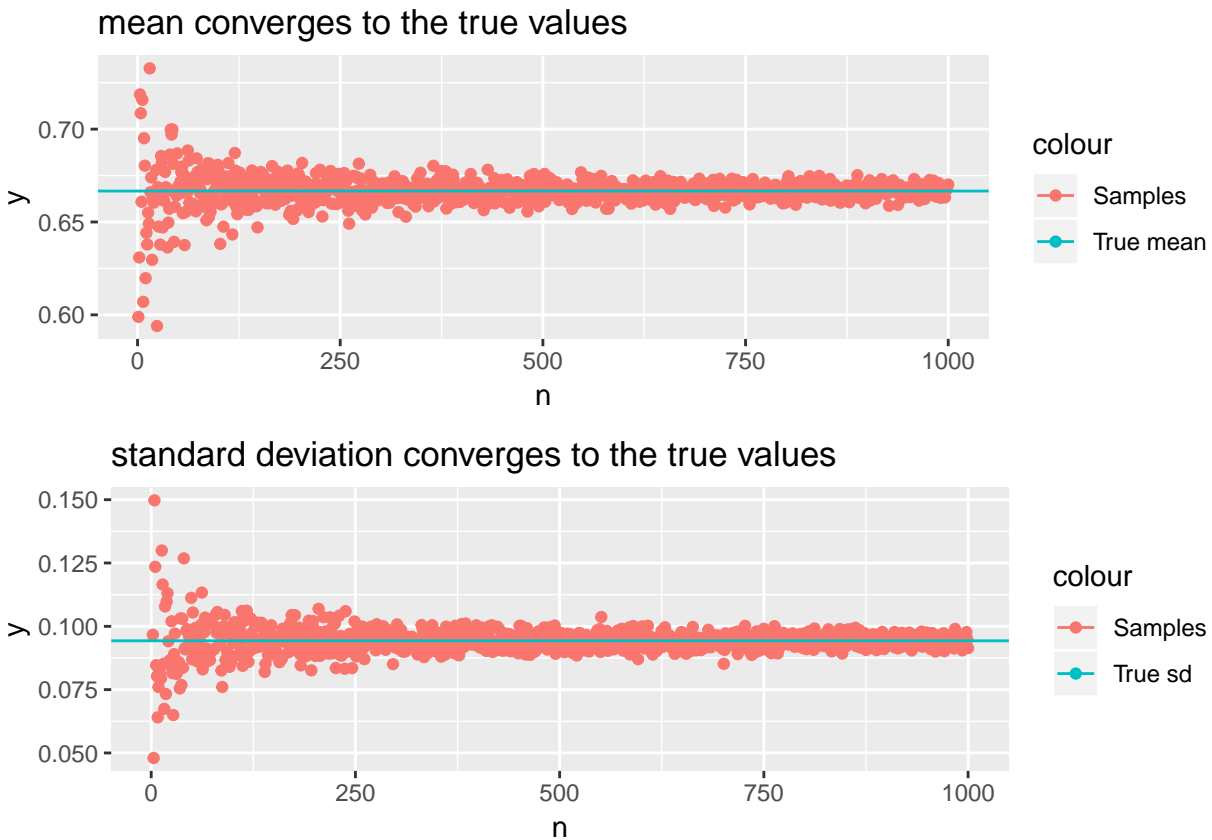
$$E[\theta] = \frac{\alpha}{\alpha + \beta} \text{Var}[\theta] = \frac{\alpha\beta}{(\alpha + \beta)^2(\alpha + \beta + 1)}$$

```
# 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(12345)
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  $\Pr(\theta < 0.4 \mid y)$  and compare with the exact value [Hint: pbeta()].

```
set.seed(12345)
# simulation to compute the posterior probability - of beta theata < 0.4
nDraws_b = 10000
sample_b = rbeta(n = nDraws_b, shape1 = 16, shape2 = 8)
sample_b_binary = ifelse(sample_b < 0.4, 1, 0) # if value in sample is smaler than 0, than 1 else 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)
```

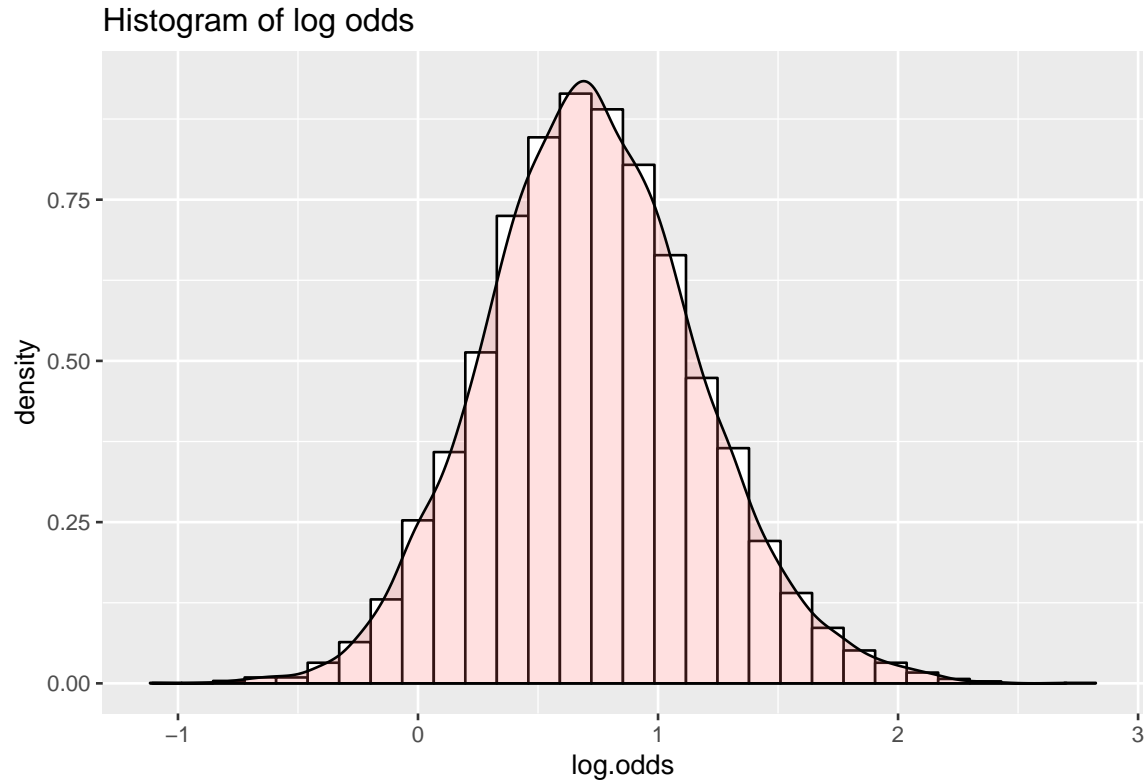
Expected.value	Simulated.vaule
0.004	0.0046

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

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

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

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  $y_1, \dots, y_n \mid \mu, \sigma^2 \stackrel{\text{iid}}{\sim} \log \mathcal{N}(\mu, \sigma^2)$ , where  $\mu = 3.5$  is assumed to be known but  $\sigma^2$  is unknown with non-informative prior  $p(\sigma^2) \propto \frac{1}{\sigma^2}$ . The posterior for  $\sigma^2$  is the  $\text{Inv } \chi^2(n, \tau^2)$  distribution, where

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

### (a) Simulate 10,000 draws

from the posterior of  $\sigma^2$  (assuming  $\mu = 3.5$ ) and compare with the theoretical  $\text{Inv } \chi^2(n, \tau^2)$  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 as follows:

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

- 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

```
# --- simulate values
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
}
simulated_mean = mean(computed_variance)
simulated_var = var(computed_variance)
```

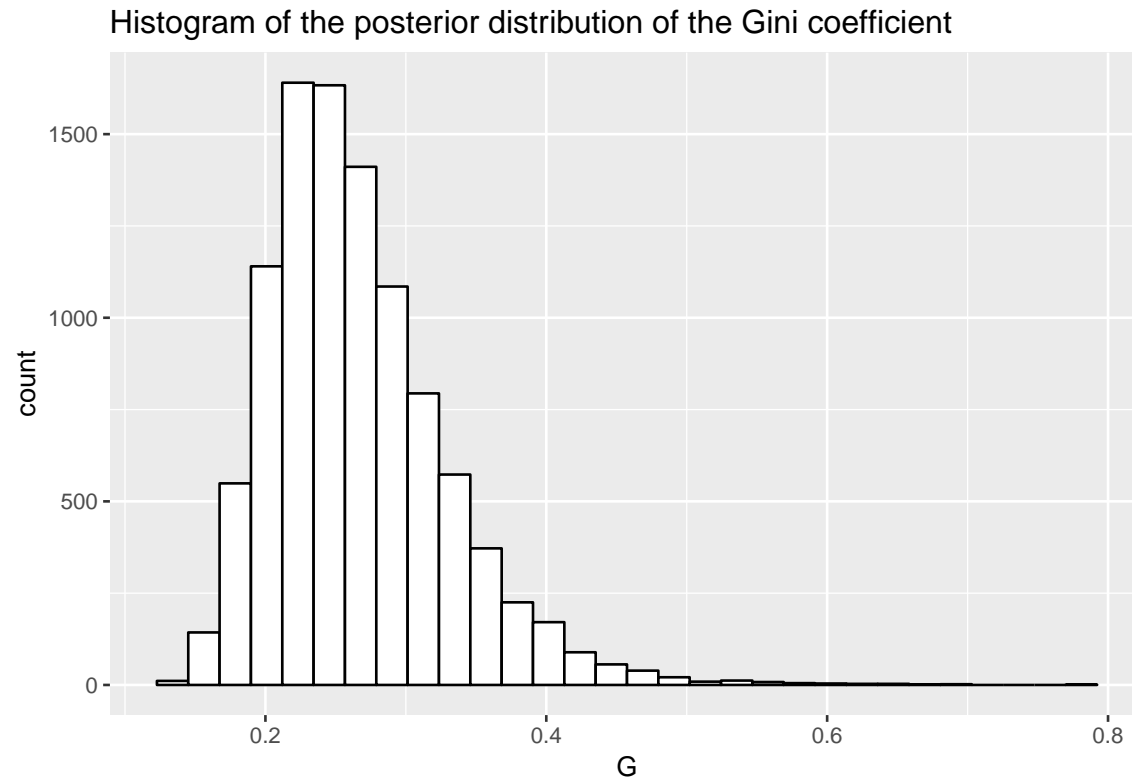
	theoreticle.value	simulated.value
mean	0.2473497	0.2467178
variance	0.0203940	0.0207204

## (b) Gini coefficient

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  $G = 2\phi(\frac{\sigma}{\sqrt{2}}) - 1$  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(mean = 0, sd = 1, sqrt(computed_variance)/sqrt(2)) -1
```

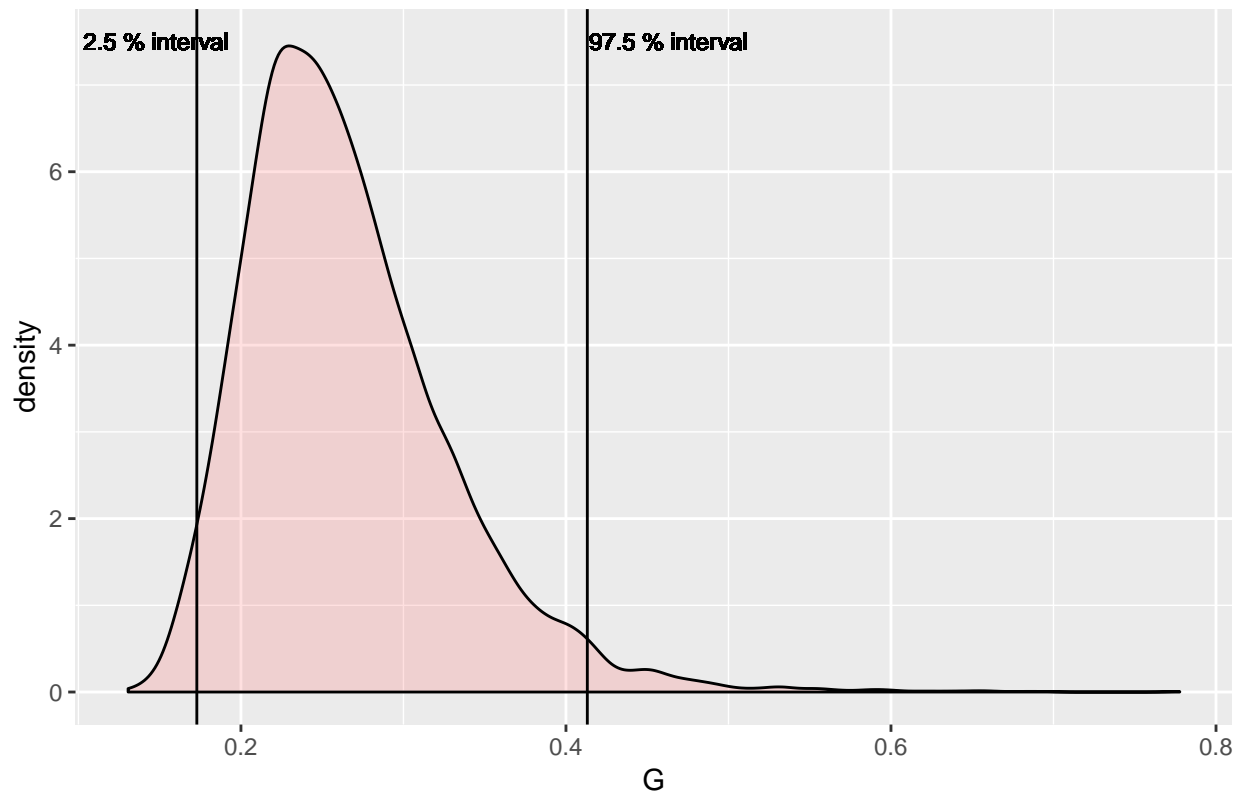


### (c) tail credible interval for G

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.

```
# 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]
# 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)
```

## 95% credible interval of G



```
# Method 2:
# Highest Posterior Density (HPD)
# 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

## [1] 1.014393

ci_95 = 91
ci_volum_start = density_y[density_y_max_index]/density_y_sum*100 # volumen at the highest 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
}

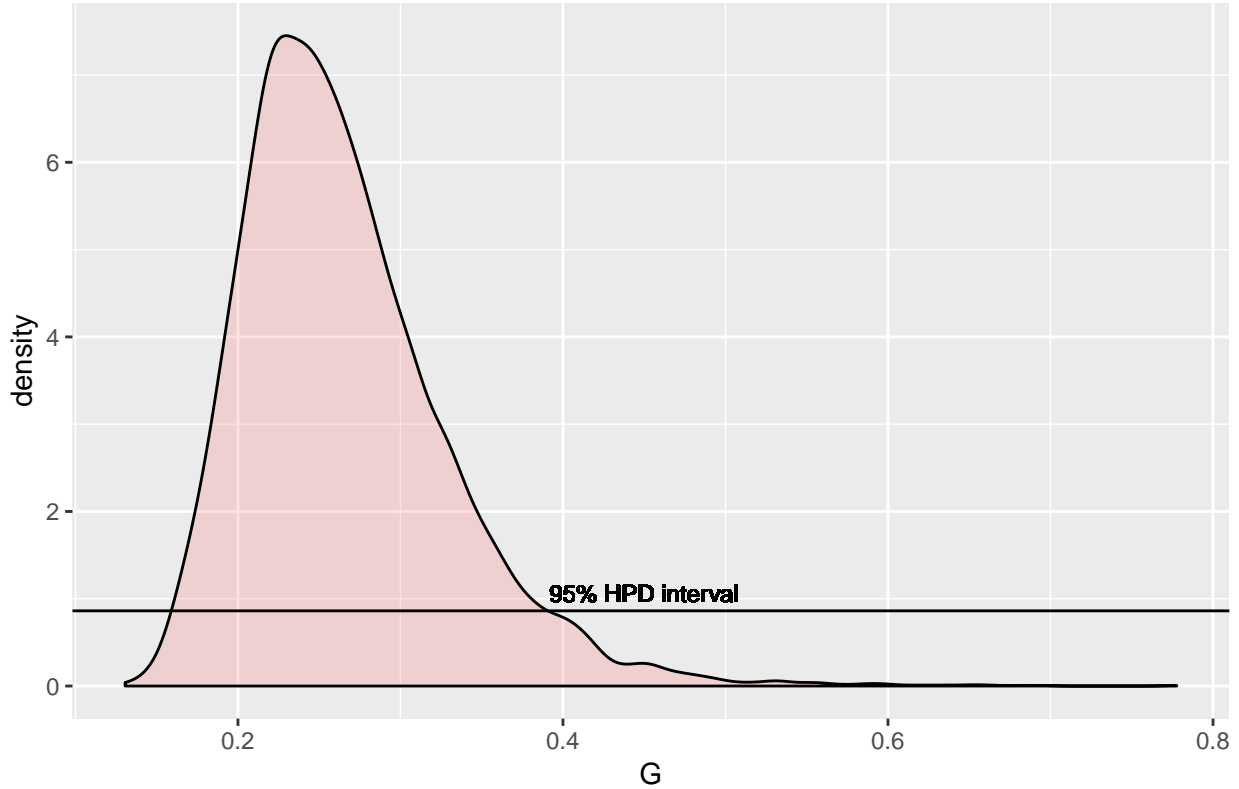
# idea 2 - h-line
density_G = density(G)
density_y = density_G$y
density_x = density_G$x
density_df = data.frame(nr = 1:length(density_x),
                        density_x = density_x,
                        density_y = density_y)
density_df_ordered = density_df[order(density_y, decreasing = TRUE),]
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),]

ggplot(data = plot_data2.b, aes(x = G)) +
  ggtitle("95% Highest Posterior Density interval for G") +
  geom_density(alpha=.2, fill="#FF6666") +
  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 )

```



### 95% Highest Posterior Density interval for G



## 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 Wikipedia's description of probability distributions for circular data we convert the data into radians  $-\pi \leq y \leq \pi$ . 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

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

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

### (a) Plot the posterior distribution

of  $\kappa$  for the wind direction data over a fine grid of  $\kappa$  values.

## (b) Find the (approximate) posterior mode

of  $\kappa$  from the information in a).

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

$$posterior \propto likelihood \times prior$$

- \*\* Likelihood - Mises distribution\*\*

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

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

- \*\* prior - Exponential distribution\*\*

$$\lambda \cdot \exp[-\lambda x], \text{ where } \lambda = 1 \exp[-\kappa]$$

$$posterior \propto \frac{1}{I_o(\kappa)^2} \cdot \exp[-\kappa] \propto \frac{1}{I_o(\kappa)^2} \exp[\kappa \cdot \cos(y_i - \mu) - \kappa] \propto \frac{1}{I_o(\kappa)^2} \exp[\kappa(\cos(y_i - \mu) - 1)]$$

```
# calculate the posterior
# version 1
result_3a = exp(k*sum(cos(radians-mu_3a))-k)/besselI(x = k, nu=0)^n_radians
# version 2
#result_3a = exp(k*(sum(cos(radians-mu_3a))-1))/besselI(x = k, nu=0)^n_radians
```



## Appendix

```
knitr::opts_chunk$set(echo = TRUE)
# libraries for this lab
library(ggplot2)
library(gridExtra)
#install.packages('latex2exp')
library(latex2exp)
#install.packages("HDInterval")
library(HDInterval)
library(coda)
# clean environment
rm(list=ls())
# calculate true beta mean and variance
# initialize given values
alpha0 = 2
beta0 = 2
n = 20
s = 14
f = n - s
alpha_new = alpha0 + s
beta_new = beta0 + f
# calculate the true mean and standard deviation
true_mean = alpha_new/(alpha_new + beta_new)
true_var = (alpha_new*beta_new) / ((alpha_new + beta_new)^2 * (alpha_new + beta_new + 1))
true_sd = sqrt(true_var)
# calculate how the mean changes with increasing number of n - until 10000 values
set.seed(12345)
iterations = 1:1000
mean_of_n = c()
for (i in 1:length(iterations)) {
  mean_of_n[i] = mean(rbeta(n = i, shape1 = alpha_new, shape2 = beta_new))
}
# calculate how the sd changes with increasing number of n
sd_of_n = c()
for (i in 1:length(iterations)) {
  sd_of_n[i] = sd(rbeta(n = i, shape1 = alpha_new, shape2 = beta_new))
}
# create plot data
plot_data1.a = data.frame(x = iterations,
                          true_mean = true_mean,
                          true_sd = true_sd)
# create converges plot for mean and sd
mean_converges = ggplot(data = plot_data1.a, aes(x = plot_data1.a$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")
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)
# Example code for the function pnorm
```

```

# What is P(Z > 1.83)?
pnorm(1.83, mean=0, sd = 1, lower=F)
# What is P(Z < -0.42)?
pnorm(-0.42, mean=0, sd = 1, lower=T)
set.seed(12345)
# simulation to compute the posterior probability - of beta theata < 0.4
nDraws_b = 10000
sample_b = rbeta(n = nDraws_b, shape1 = 16, shape2 = 8)
sample_b_binary = ifelse(sample_b < 0.4, 1, 0) # if value in sample is smaler than 0, than 1 else 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
# --- theorecticle 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)
# theorecticle mean
theoreticle_mean = n_observations * tau_2/(n_observations-2)
# theorecticle var
theoreticle_var = 2 * n_observations^2 * tau_2^2 / ((n_observations -2)^2 * (n_observations - 4))

```

```

# --- simulate values
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
}
simulated_mean = mean(computed_variance)
simulated_var = var(computed_variance)
result_2a_data = data.frame("theoreticle value" = c(theoreticle_mean,theoreticle_var),
                             "simulated value" = c(simulated_mean, simulated_var))
rownames(result_2a_data) = c("mean", "variance")
knitr::kable(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(mean = 0, sd = 1, sqrt(computed_variance)/sqrt(2)) -1
# create histogramm
#hist(G)
# plot data
plot_data2.b = data.frame("G" = G,
                           "nr" = 1:length(G))
# Histogram
ggplot(data = plot_data2.b) +
  ggtitle("Histogram of the posterior distribution of the Gini coefficient") +
  geom_histogram(aes(x=G),
                 colour="black",
                 fill="white",
                 bins=30)
# 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]
# 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 )
# Method 2:
# Highest Posterior Density (HPD)
# 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 highest 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
}
# 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$in_ci == TRUE),]
HPD = density_df_ordered_trueci[nrow(density_df_ordered_trueci),]
ggplot(data = plot_data2.b, aes(x = G)) +
  ggtitle("95% Highest Posterior Density interval for G") +
  geom_density(alpha=.2, fill="#FF6666") +
  geom_hline(aes(yintercept = HPD$density_y)) +
  geom_text(aes(label = "95% HPD interval", x = 0.45, y = HPD$density_y+0.2),size = 3 )
# 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
# version 1
result_3a = exp(k*sum(cos(radians-mu_3a))-k)/besselI(x = k, nu=0)^n_radians

```

```

# version 2
#result_3a = exp(k*(sum(cos(radians-mu_3a))-1))/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)
k_max = result_3a[k_max_index]
plot_3a + geom_vline(aes(xintercept = (k_max_index*0.01 + 0.01)))

```