Lab1 Report

Ludvig Noring, Michael Sörsäter

April 11, 2017

1. Bernoulli

(a)

Using the sample with 14 successes and 6 failures and a Beta prior with $\alpha = \beta = 2$ we get the posterior $\theta|y \sim Beta(16, 8)$ from which we draw random numbers. As the number of draws increases the sampled mean and standard deviation converges towards the theoretical mean and standard deviation for the posterior. This can be seen in Figure 1 and Figure 2.

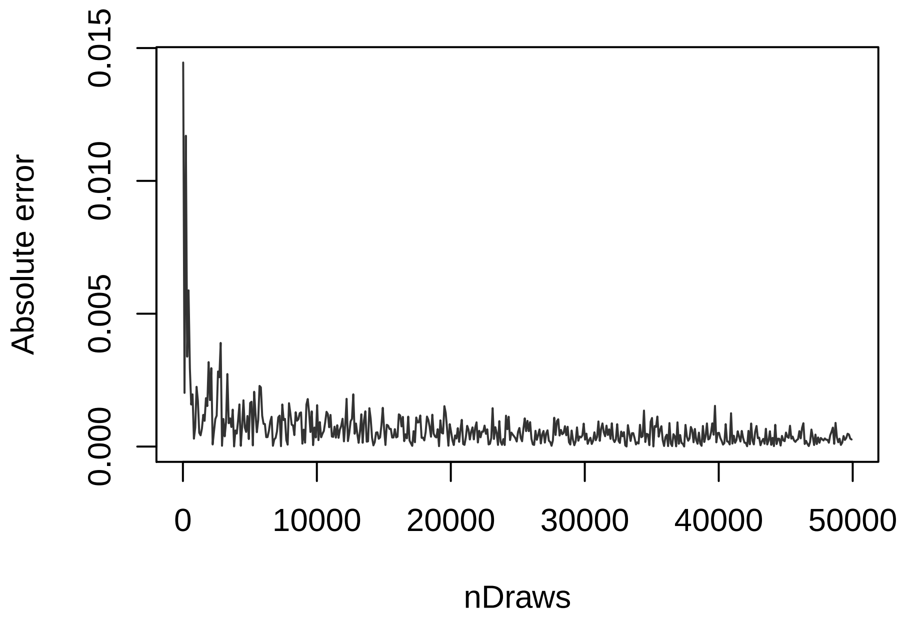

Figure 1: Absolute error of the sampled and theoretical mean

(b)

With 10 000 simulated draws we compute the posterior probability to $Pr(\theta < 0.4|y) = 0.35\%$. This can be compared with the theoretical value 0.397%.

With just so few as 10 000 draws we get a probability close to the exact value.

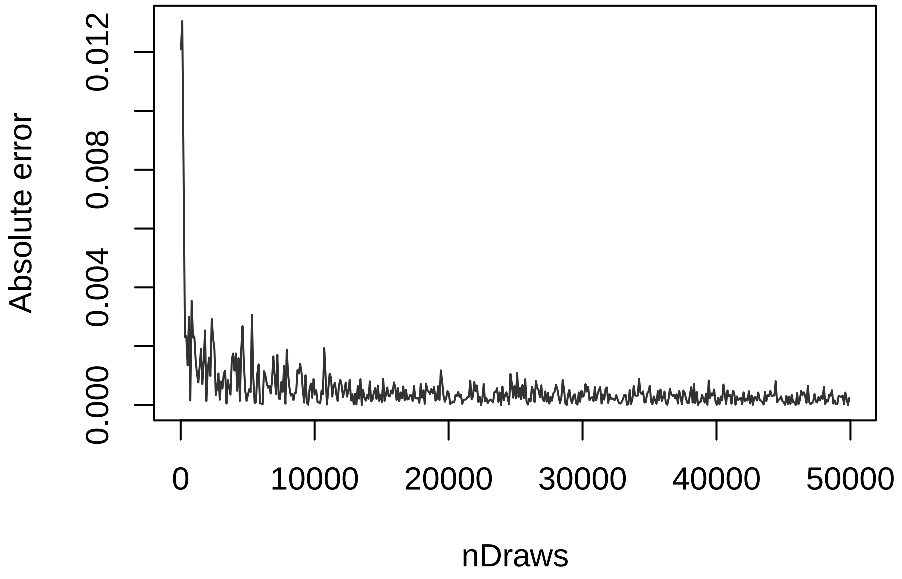

Figure 2: Absolute error of the sampled and theoretical standard deviation

(c)

When using simulation it is easy to compute the log-odds of the posterior distribution. The plot is seen in Figure 3..

2. Log-normal distribution and the Gini coefficient

(a)

10 000 draws are simulated from the posterior $Inv - \chi^2(n, \tau^2)$. The result is plotted with the theoretical distribution and is seen in Figure 4. They are very similar.

(b)

We calculate the Gini coefficient for different values of σ and plot the result which can be seen in Figure 5.

(c)

The equal tails interval and highest posterior density interval are computed from the draws in 2b. The density have a very long right tail which influence the equal tail interval. Therefore the highest posterior density seems to fit the data better. The plot with the credible intervals can be seen in Figure 6.

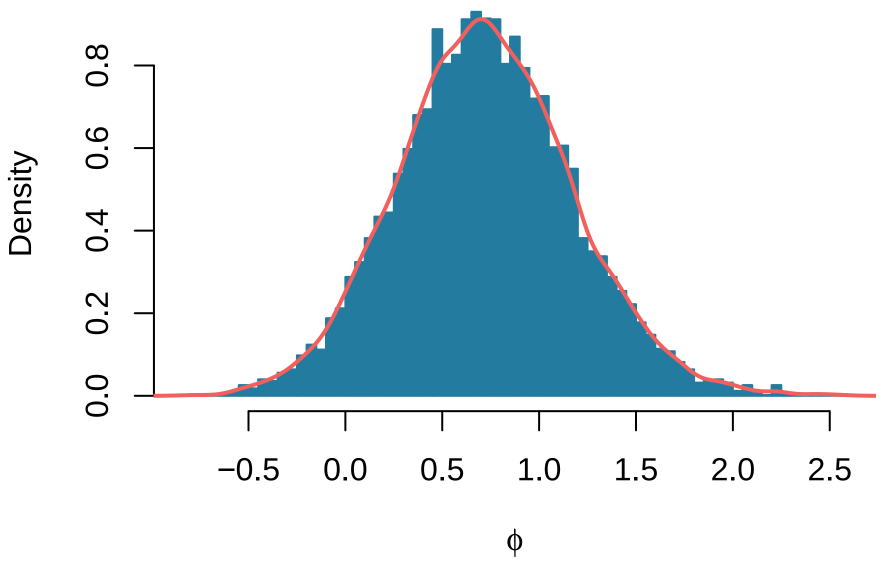

Figure 3: Log-odds

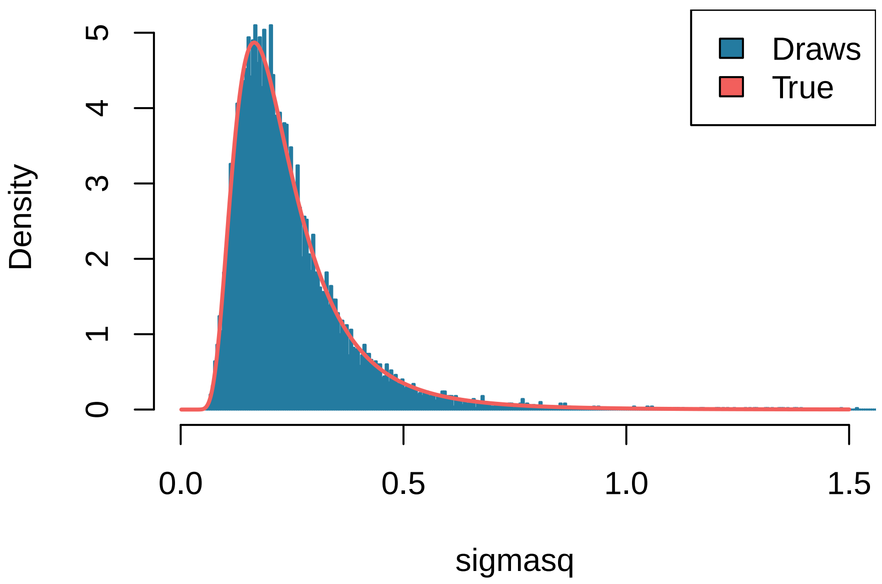

Figure 4: $Inv - \chi^2(n, \tau^2)$ Simulated and theoretical

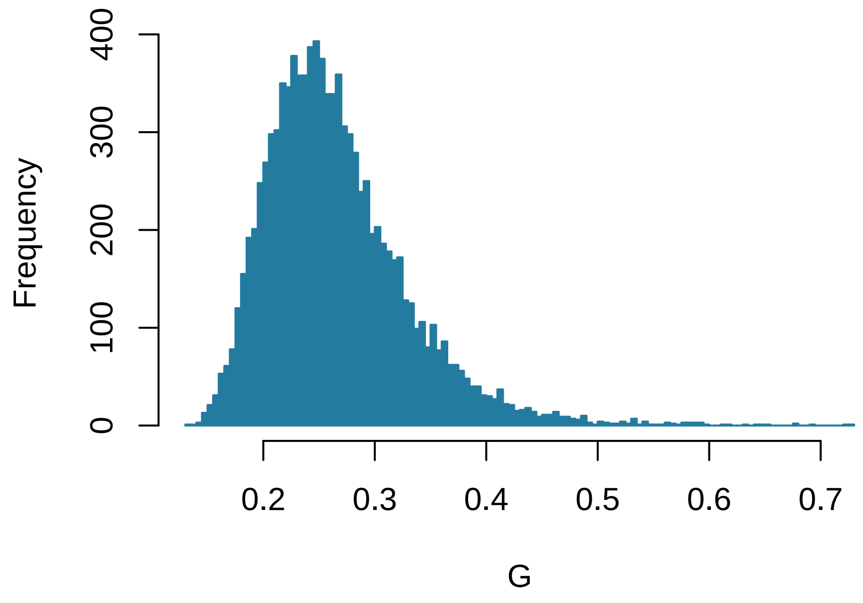

Figure 5: Gini coefficient

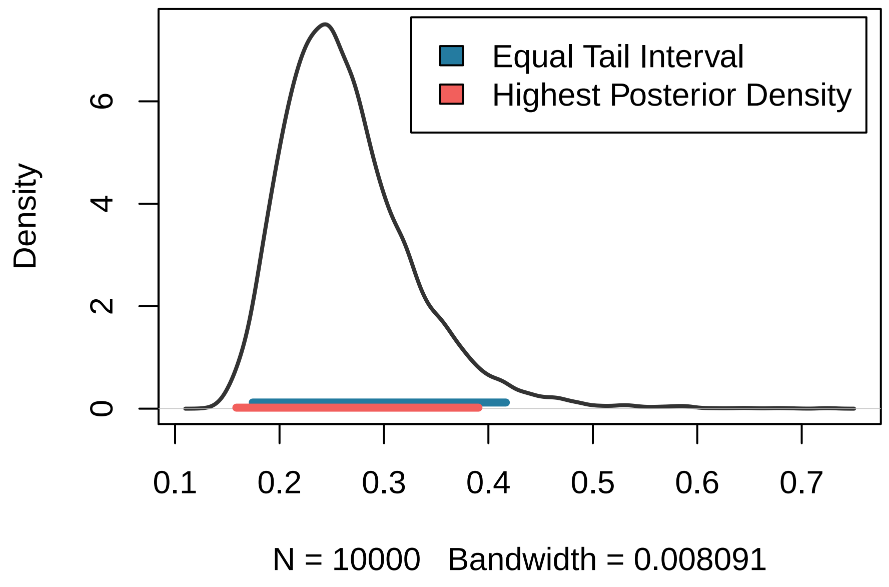

Figure 6: Credible intervals

3. Von Mises distribution

The posterior distribution is calculated by multiplying the individual observations together with the prior for a fine grid of κ values. The mode is calculated for the posterior. The plot can be found in Figure 7.

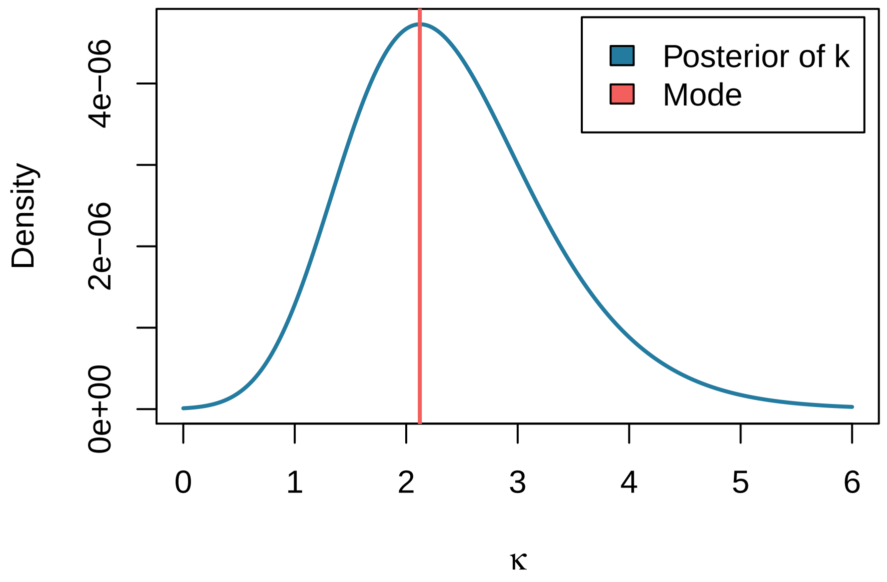

Figure 7: Von Mises distribution

Appendix A - Code for assignment 1

```r
set.seed(12345)
col1 = '#247ba0'
col2 = '#f25f5c'
col3 = '#333333'

imgw = 5
imgh = 4

a = 2
b = 2
s = 14
n = 20

xGrid <- seq(0.001, 0.999, 0.001)

## 1a
draw = function(nDraws){
  sample = rbeta(nDraws, a+s, b+(n-s))
  mean_sample = mean(sample)
  sd_sample = sd(sample)

  return (c(mean_sample, sd_sample))
}

alpha = a + s
beta = b + (n-s)

intervals = seq(20,50000,100)
data = sapply(intervals, draw)

true = dbeta(xGrid, a+s, b+(n-s))
mean_true = alpha / (alpha + beta)
sd_true = sqrt( (alpha*beta) / ((alpha+beta)^2*(alpha+beta+1)) )

# Mean
pdf("plots/1-converge_mean.pdf", width=imgw, height=imgh)
  plot(intervals, abs(data[1,]-mean_true), type='l', col=col3,
    xlab = 'nDraws', ylab='Absolute error')
dev.off()

# SD
pdf("plots/1-converge_sd.pdf", width=imgw, height=imgh)
  plot(intervals, abs(data[2,]-sd_true), type='l', col=col3,
    xlab = 'nDraws', ylab='Absolute error')
dev.off()

## 1b
nDraws = 10000
posterior = rbeta(nDraws, a+s, b+(n-s))
true = pbeta(0.4, a+s, b+(n-s))
```

```r
# Simulated value
message((sum(posterior <= 0.4) / length(posterior)) * 100)
# Theoretical value
message(round(true*100, 3))

## 1c
nDraws = 10000
posterior = rbeta(nDraws, a+s, b+(n-s))
phi = log(posterior / (1 - posterior))

pdf("plots/1-phi.pdf", width=imgw, height=imgh)
  hist(phi, 100, prob=TRUE, col=col1, border=col1, main='', xlab=expression(phi))
  lines(density(phi), lwd=2, col=col2)
dev.off()
```

Appendix B - Code for assignment 2

```
col1 = '#247ba0'
col2 = '#f25f5c'
col3 = '#333333'
imgw = 5
imgh = 4

mu = 3.5
obs = c(14, 25, 45, 25, 30, 33, 19, 50, 34, 67)
n = length(obs)

tausq = sum((log(obs) - mu)^2) / (n)

chiDraw = function(nDraws){
  #draws = rchisq(nDraws, n)
  #sigmasq = n * tausq / draws
  sigmasq = rinvchisq(nDraws, n, tausq)
  return (sigmasq)
}

nDraws = 10000
sigmasq = chiDraw(nDraws)

s = seq(0, 1.5, length.out = 1000)
#exp_part = exp( ((-n*tausq) / (2*s)) ) / (s^(1+n/2))
#constant = ((tausq * n/2)^(n/2)) / (factorial(n/2-1))
#                                     gamma(n/2)
#PDF = constant * exp_part
PDF = dinvchisq(s, n, tausq)

pdf('plots/2-chi-squared.pdf', width=imgw, height=imgh)
  hist(sigmasq, 500, col=col1, border=col1, main='', freq=FALSE, xlim=c(0,1.5))
  lines(s, CDF, col=col2, lwd=2)
  legend('topright', c('Draws', 'True'), fill=c(col1, col2))
dev.off()

## 2.b Gini
vals = sqrt(sigmasq/2)
G = 2 * pnorm(vals) - 1

pdf('plots/2-g.pdf', width=imgw, height=imgh)
  hist(G, 100, col=col1, border=col1, main='')
dev.off()


## 2.c Credible Intervals

# Equal tail interval
x_eti = quantile(G, probs=c(0.025, 0.975))

# Highest Posterior Density
dg = density(G)
```

```r
# Sort data on y decending values
ordered_x = dg$x[order(-dg$y)]
ordered_y = dg$y[order(-dg$y)]

cur = 0
for(i in 1:length(dg$y)){
  cur = cur + ordered_y[i]
  if(cur / sum(dg$y) >= 0.95){
    break
  }
}
x_hpd = c(min(ordered_x[1:i]), max(ordered_x[1:i]))

pdf('plots/2-credible-intervals.pdf', width=imgw, height=imgh)
  plot(dg, col=col3, lwd=2, main="")
  lines(x_eti, rep(0.12, 2), col=col1, lwd=4)
  lines(x_hpd, rep(0.02, 2), col=col2, lwd=4)
  #points(ordered_x[1:i], rep(0, i), cex=0.1)
  legend("topright",
         legend = c("Equal Tail Interval","Highest Posterior Density"),
         fill = c(col1, col2),
         inset = 0.02)
dev.off()
```

Appendix C - Code for assignment 3

```r
col1 = '#247ba0'
col2 = '#f25f5c'
imgw = 5
imgh = 4

y = c(-2.44, 2.14, 2.54, 1.83, 2.02, 2.33, -2.79, 2.23, 2.07, 2.02)
mu = 2.39

von_misen = function(y, kappa, mu){
  I_zero = besselI(x=kappa, nu=0)
  numerator = exp(kappa * cos(y - mu))
  return (numerator / (2 * pi * I_zero))
}

exponential = function(kappa, lambda=1){
  return (lambda * exp(-lambda*kappa))
}

calculate_posterior = function(y, kappa, mu) {
  prior = exponential(kappa)
  likelihood = prod(von_misen(y, kappa, mu))
  return (prior * likelihood)
}

# To calculate the mode
get_mode = function(vec, kappas) {
  vec = round(vec, 10)
  uniqv = unique(vec)
  mode = uniqv[which.max(tabulate(match(vec, uniqv)))]

  mode_idx = which(vec == mode)[1]
  mode_x = kappas[mode_idx]

  return (mode_x)
}

kappas = seq(0, 6, 0.001)

posterior = sapply(kappas, calculate_posterior, y=y, mu=mu)
mode = get_mode(posterior, kappas)

pdf("plots/3-posterior-distribution.pdf", width=imgw, height=imgh)
  plot(kappas, posterior, type='l', lwd=2, col=col1,
       xlab=expression(kappa), ylab="Density")
  abline(v=mode, col=col2, lwd=2)
  legend('topright', c('Posterior of k', 'Mode'), fill=c(col1, col2), inset=0.02)
dev.off()
```