Bayesian Learning

2024-05-27

Computer Lab 1

Hugo Morvan (hugmo418).

```
library(ggplot2)
```

1. Daniel Bernoulli

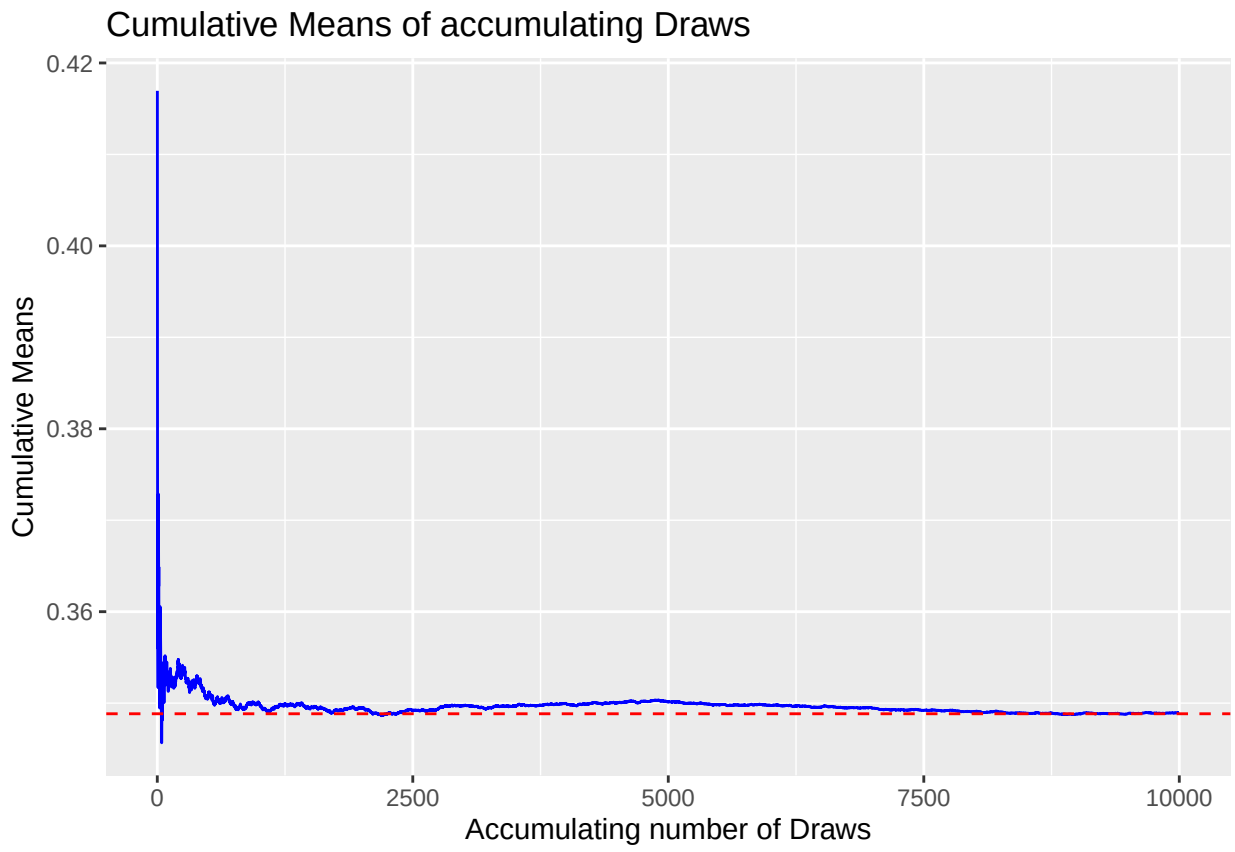
Let $y_1, ..., y_n|\theta \sim Bern(\theta)$, and assume that you have obtained a sample with s = 22 successes in n = 70 trials. Assume a Beta(α_0, β_0) prior for θ and let $\alpha_0 = \beta_0 = 8$.

1.a

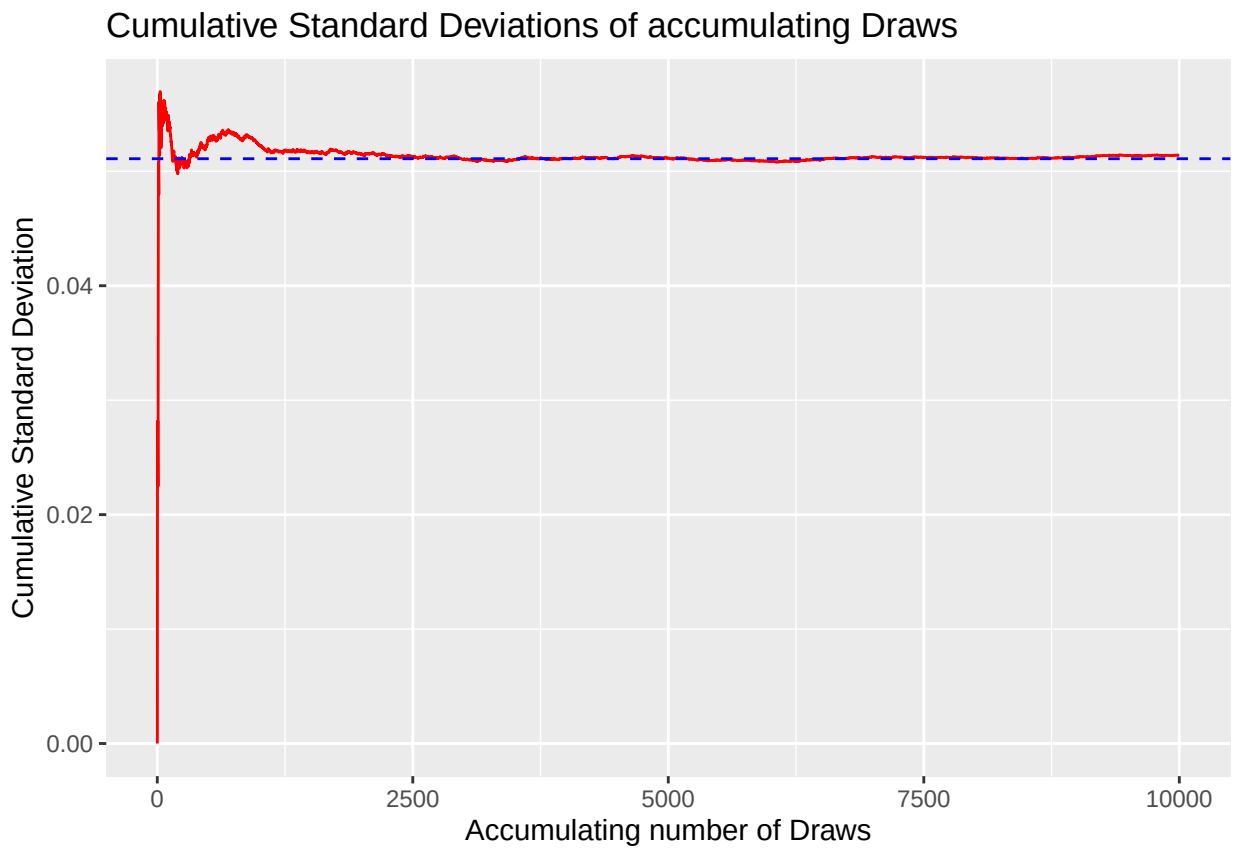
Draw 10000 random values (nDraws = 10000) from the posterior $\theta|y \sim Beta(\alpha_0 + s, \beta_0 + f)$, where $y = (y_1, ..., y_n)$, and verify graphically that the posterior mean $E[\theta|y]$ and standard deviation $SD[\theta|y]$ converges to the true values as the number of random draws grows large. [Hint: use rbeta() to draw random values and make graphs of the sample means and standard deviations of θ as a function of the accumulating number of drawn values].

```
s <- 22
n <- 70
f <- n-s
nDraws <- 10000
alpha_0 <- 8
beta_0 <- 8

post_alpha <- alpha_0 + s
post_beta <- beta_0 + f

# rbeta drawing
delta <- rbeta(nDraws, post_alpha, post_beta)

# true values of mean and sd
mean_true <- post_alpha/(post_alpha + post_beta)
sd_true <- sqrt((post_alpha*post_beta)/((post_alpha+post_beta+1)*(post_alpha+post_beta)^2))

# sample means based on accumulating number of drawn values
cumu_mean <- cumsum(delta)/(1:nDraws)
cumu_sd <- sqrt(cumsum((delta - cumu_mean)^2) / (1:nDraws))

plot_data <- data.frame(
```

```r
  Draw = 1:nDraws,
  Mean = cumu_mean,
  SD = cumu_sd
)

# Mean plot
cumulativeMeans_plot <- ggplot(plot_data, aes(x = Draw)) +
  geom_line(aes(y = Mean), color = "blue") +
  # true mean line
  geom_hline(yintercept = mean_true, color = "red", linetype = "dashed") +
  ggtitle("Cumulative Means of accumulating Draws") +
  xlab("Accumulating number of Draws") +
  ylab("Cumulative Means")

# SD plot
cumulativeSD_plot <- ggplot(plot_data, aes(x = Draw)) +
  geom_line(aes(y = SD), color = "red") +
  # true standard deviation line
  geom_hline(yintercept = sd_true, color = "blue", linetype = "dashed") +
  ggtitle("Cumulative Standard Deviations of accumulating Draws") +
  xlab("Accumulating number of Draws") +
  ylab("Cumulative Standard Deviation")

cumulativeMeans_plot
```



```
cumulativeSD_plot
```



1.b

Draw 10000 random values from the posterior to compute the posterior probability $Pr(\theta > 0.3|y)$ and compare with the exact value from the Beta posterior. [Hint: use pbeta()]

```
# posterior probability:
p_post <- mean(delta > 0.3)

# beta posterior probability:
p_beta_post <- 1-(pbeta(0.3, post_alpha, post_beta))

p_post
```

```
## [1] 0.8287
```

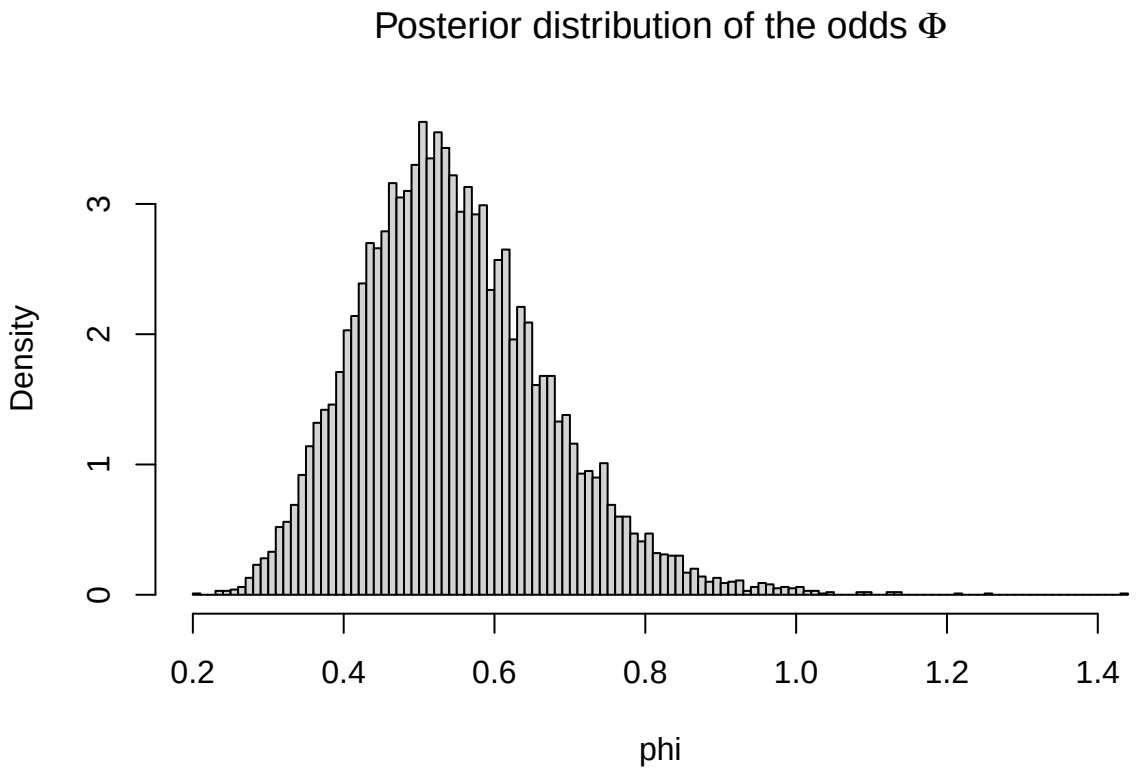
```
p_beta_post
```

```
## [1] 0.8285936
```

1.c

Draw 10000 random values from the posterior of the odds $\phi = \frac{\theta}{1-\theta}$ by using the previous random draws from the Beta posterior for θ and plot the posterior distribution of ϕ. [Hint: hist() and density() can be utilized].

```
phi <- delta/(1-delta)

hist(phi, breaks=100, prob=TRUE, xlab="phi", main=expression(paste("Posterior distribution of the odds
```



2. Log-normal distribution and Gini coefficient

Assume that you have asked 8 randomly selected persons about their monthly income (in thousands Swedish Krona) and obtained the following eight observations: 33, 24, 48, 32, 55, 74, 23, and 17. A common model for non-negative continuous variables is the log-normal distribution. The log-normal distribution $logN(\mu, \sigma^2)$ has density function

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

, where y > 0, $-\infty < \mu < \infty$ and $\sigma^2 > 0$. The log-normal distribution is related to the normal distribution as follows: if $y \sim logN(\mu, \sigma^2)$ then $logy \sim N(\mu, \sigma^2)$. Let $y_1, ..., y_n|\mu$, $\sigma^2 iid \sim logN(\mu, \sigma^2)$, where $\mu = 3.6$ is assumed to be known but σ^2 is 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 $\tau^2 = \frac{\sum_{i=1}^{n}(\log y_i - \mu)^2}{n}$.

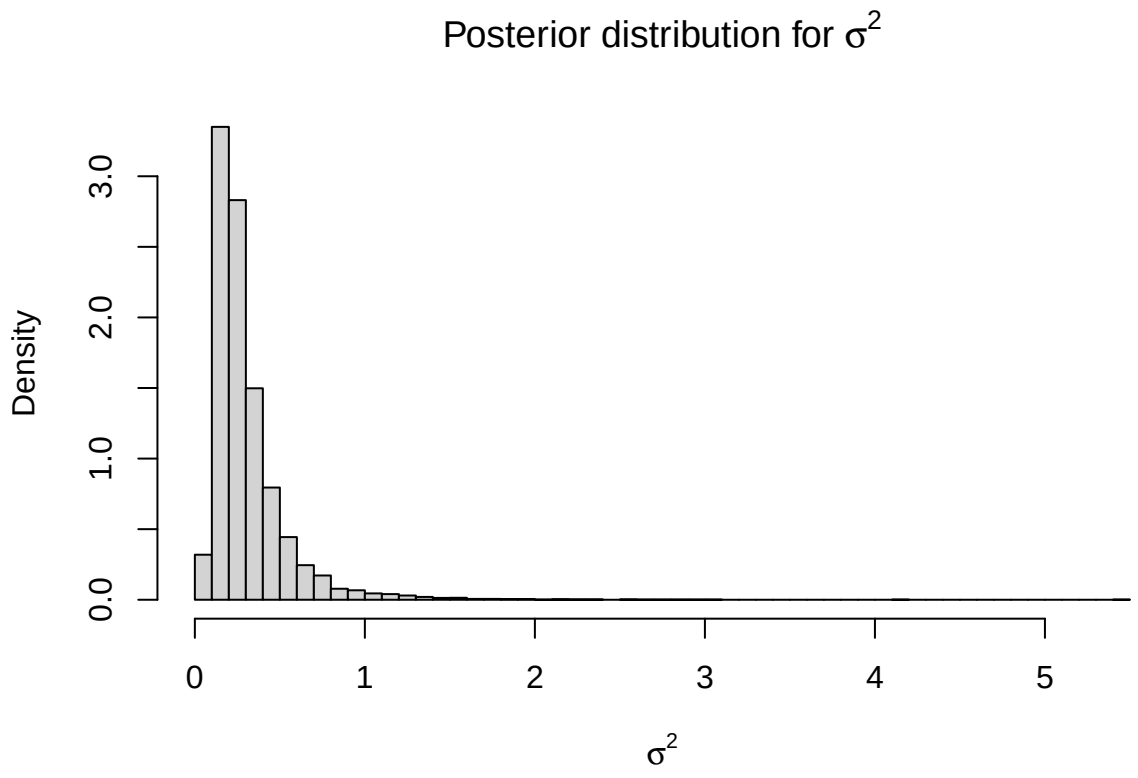
2.a

Draw 10000 random values from the posterior of σ^2 by assuming $\mu = 3.6$ and plot the posterior distribution.

To simulate from the posterior distribution, we use the steps indicated in lecture 3:

1. Draw $X \sim \chi^2(n-1)$
2. Compute $\sigma^2 = \frac{(n-1)s^2}{X}$ (a draw from $Inv - \chi^2(n-1, s^2)$)
3. Draw a θ from $N(\bar{x}, \frac{\sigma^2}{n})$ conditional on the previous σ^2.
4. Repeat step 1-3 many times.

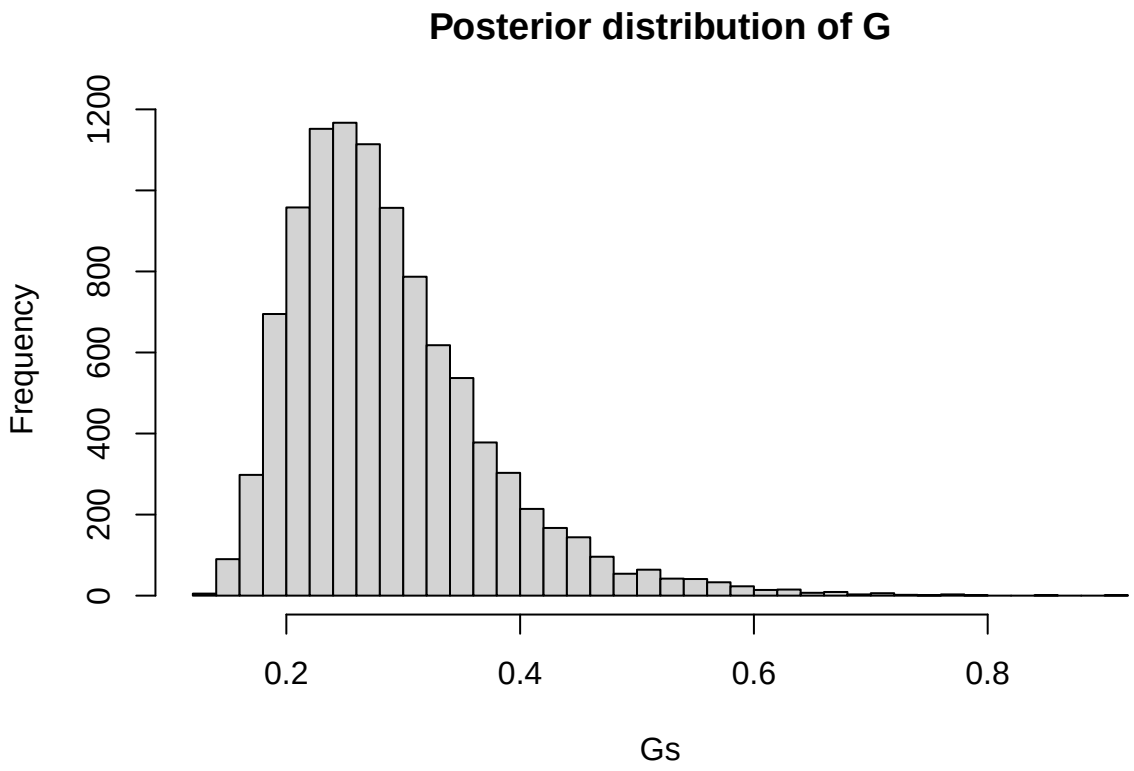
```
Y = c(33, 24, 48, 32, 55, 74, 23, 17)
n = length(Y)
df = n-1

mu = 3.6
tau_sq = (sum((log(Y)-mu)^2))/n #s^2 in the lecture

sigsq_draws = (df*tau_sq) / rchisq(10000, df)

hist(sigsq_draws, prob=TRUE, xlab = expression(sigma^2),
     main=expression(paste("Posterior distribution for ", sigma^2)),
     breaks = 50)
```



2.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 e.g. Wikipedia for more information about the Gini coefficient). It can be shown that $G = 2\Phi(\sigma/\sqrt{2}) - 1$ when incomes follow a $logN(\mu, \sigma^2)$ distribution. $\Phi(z)$ is the cumulative distribution function function (CDF) for the standard normal distribution with mean zero and unit variance. Use the posterior draws in a) to compute the posterior distributiuon of the Gini coefficient G for the current data set.

```
Gs = 2*pnorm((sqrt(sigsq_draws)/sqrt(2))) - 1

hist(Gs, main = "Posterior distribution of G", breaks = 50 )
```

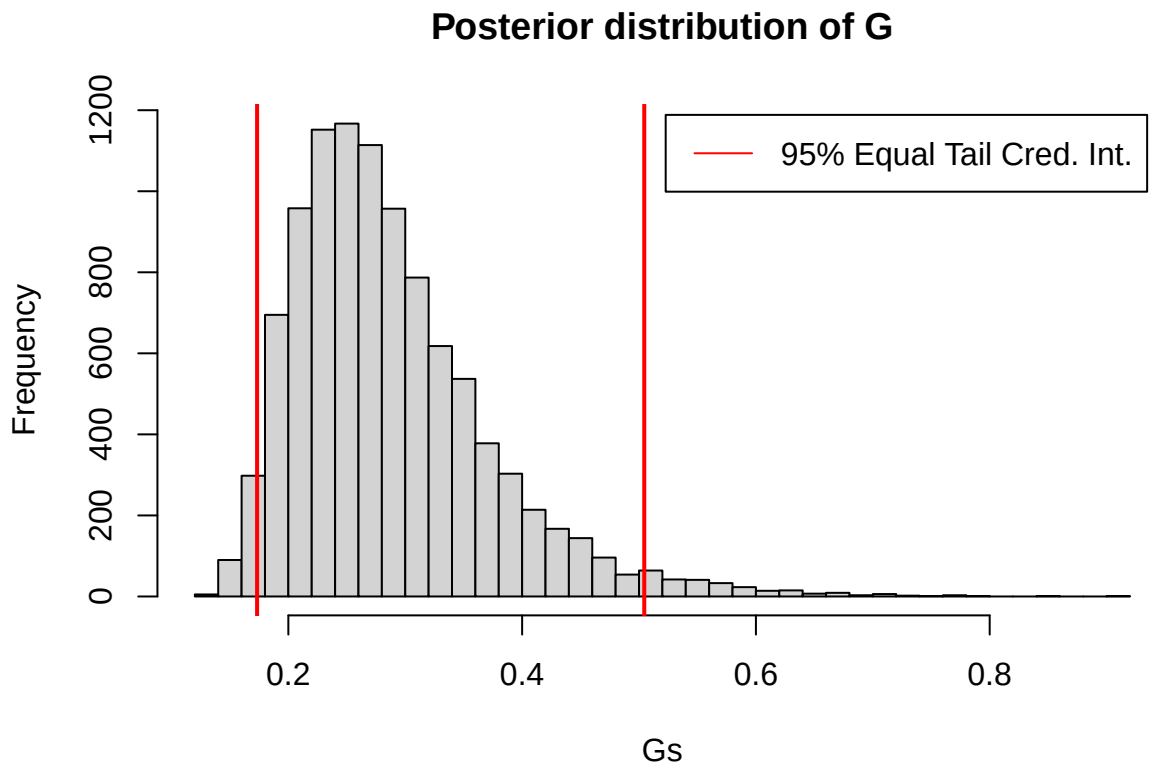


2.c

Use the posterior draws from b) to compute a 95% equal tail credible interval for G. A 95% equal tail credible interval (a, b) cuts off 2.5% percent of the posterior probability mass to the left of a, and 2.5% to the right of b.

```
sortedGs = sort(Gs)
G250 = sortedGs[250]
G9750 = sortedGs[9750]

hist(Gs, main = "Posterior distribution of G", breaks = 50 )
abline(v = G250, col = "red", lwd=2)
```

```r
abline(v = G9750, col= "red", lwd=2)
legend("topright", inset=.02, legend=c("95% Equal Tail Cred. Int."), col=c("red"), lty=1)
```



The 95% equal tail credible interval is [0.1732443, 0.5044891], marked by the red vertical lines on the plot above.

2.d

Use the posterior draws from b) to compute a 95% Highest Posterior Density Interval (HPDI) for G. Compare the two intervals in (c) and (d). [Hint: 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 the HPDI. Note that you need to order/sort the estimated density values to obtain the HPDI.].

```r
kernel_estimate = density(Gs)
x = kernel_estimate$x
y = kernel_estimate$y

y_sorted = sort(y)

#Estimating the HDPI using the Reimann sum method for approximating the area under the curve:
y_val = 7.0 #starting value
#by default, density() returns 512 points, thus Reimann sum with 512 boxes.
#box width is range(x)/ # of boxes
box_width = (max(kernel_estimate$x)-min(kernel_estimate$x))/512
```

```r
delta = 0.001 #step size for y_val

while(sum(kernel_estimate$y[kernel_estimate$y >= y_val]*box_width) < .95){
  #cat("y=", y_val, "\r")
  y_val = y_val - delta
}

print(paste("y_val=",y_val))
```

```
## [1] "y_val= 0.594999999999329"
```

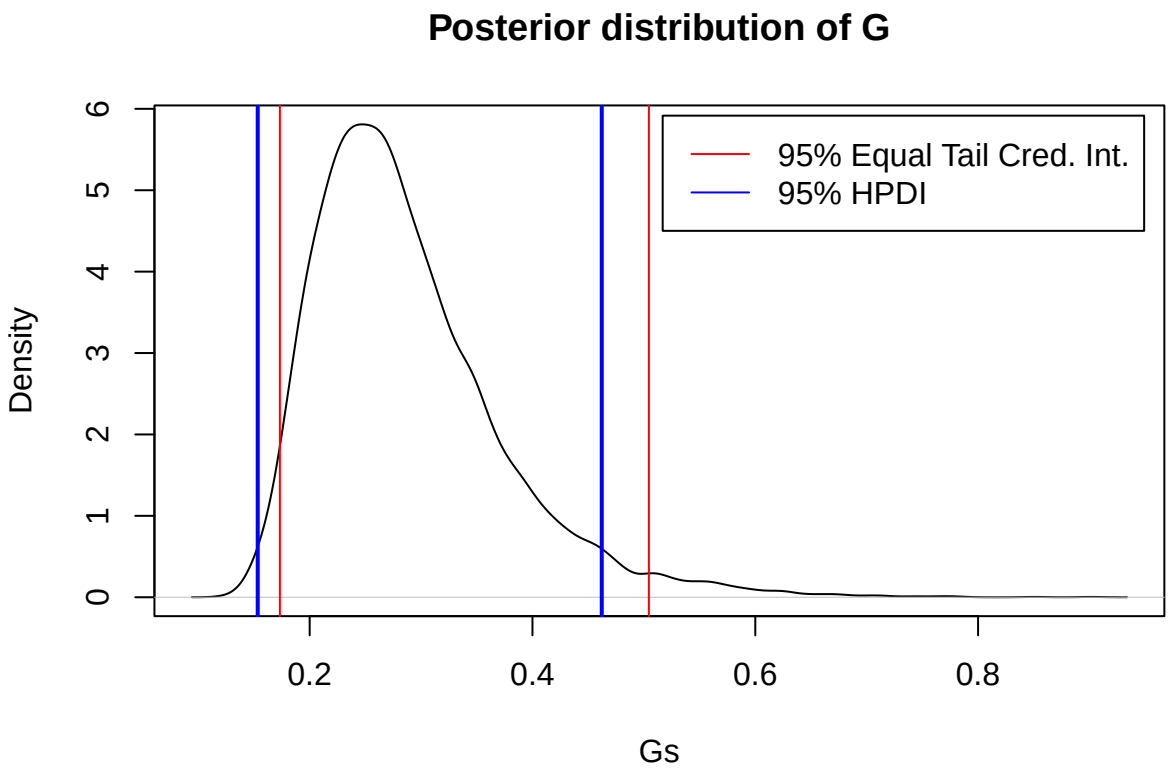
```r
x_left_bound = min(kernel_estimate$x[kernel_estimate$y >= y_val])
print(paste("x_left_bound",x_left_bound))
```

```
## [1] "x_left_bound 0.153386873364884"
```

```r
x_right_bound = max(kernel_estimate$x[kernel_estimate$y >= y_val])
print(paste("x_right_bount",x_right_bound))
```

```
## [1] "x_right_bount 0.462193440177751"
```

```r
plot(kernel_estimate, main = "Posterior distribution of G", xlab = "Gs")
abline(v=x_left_bound, col = "blue", lwd=2)
abline(v=x_right_bound, col = "blue", lwd=2)
abline(v = G250, col = "red", lwd=1)
abline(v = G9750, col= "red", lwd=1)
legend("topright", inset=.02, legend=c("95% Equal Tail Cred. Int.", "95% HPDI"),
       col=c("red", "blue"), lty=c(1,1))
```



The 95% Highest Posterior Density Interval is [0.1533869, 0.4621934], marked by the blue vertical lines on the plot above. It is shifted to the left compared to the 95% equal tail credible interval in red, which makes sense since the posterior distribution is skewed right.

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:

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

, where North is located at zero degrees (see Figure 1 on the next page, where the angles are measured clockwise). 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

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

. Assume that these data points conditional on (μ, κ) are independent observations from the following von Mises distribution:

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

where $I_0(\kappa)$ is the modified Bessel function of the first kind of order zero [see ?besselI in R]. The parameter $\mu(-\pi \leq \mu \leq \pi)$ 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 2.4. Let $\kappa \sim Exponential(\lambda = 0.5)$ a priori, where λ is the rate parameter of the exponential distribution (so that the mean is $1/\lambda$).

3.a

Derive the expression for what the posterior $p(\kappa|y,\mu)$ is proportional to. Hence, derive the function $f(\kappa)$ such that $p(\kappa|y,\mu) \propto f(\kappa)$. Then, plot the posterior distribution of κ for the wind direction data over a fine grid of κ values. [Hint: you need to normalize the posterior distribution of κ so that it integrates to one.]

To find the posterior distribution, we first need to find the Likelihood:

$$L = \prod_{i=1}^{n} \frac{\exp(\kappa * \cos(y_i - \mu))}{2\pi I_0(\kappa)} = \frac{\exp(\sum_{i=1}^{n} \kappa * \cos(y_i - \mu))}{(2\pi I_0(\kappa))^n}$$

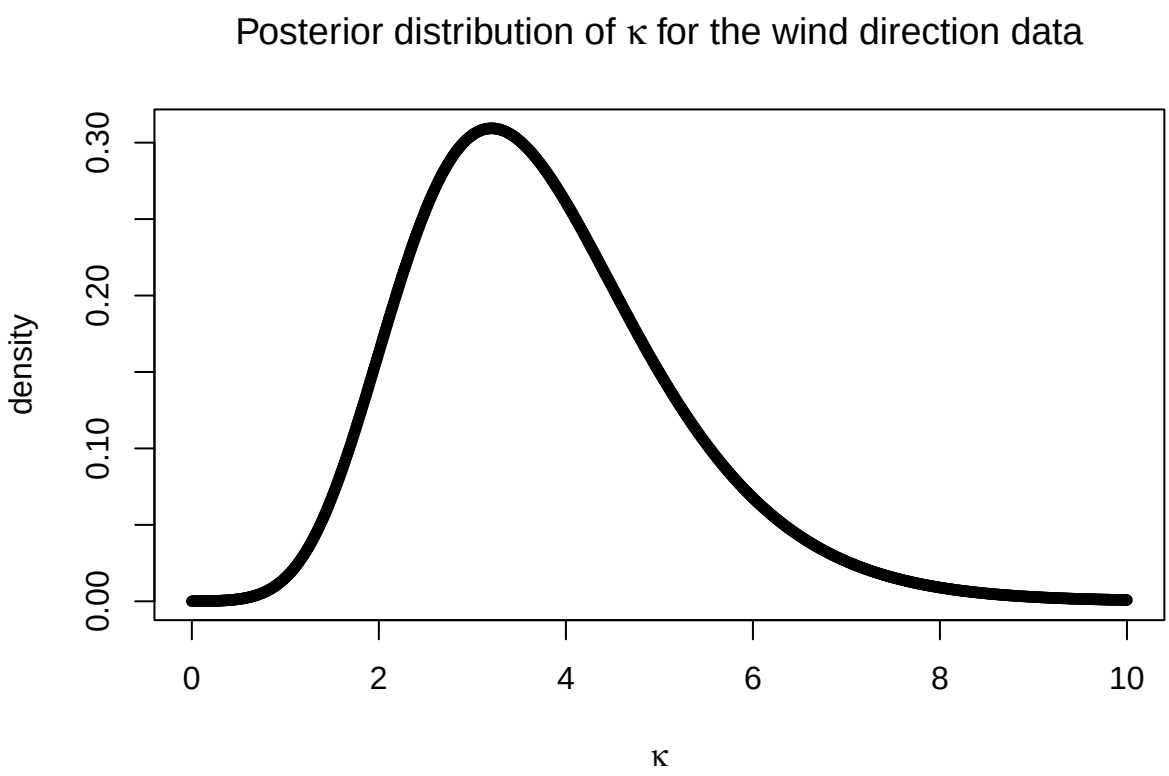
hence:

$$p(\kappa|y,\mu) \propto L * p(\kappa) = \frac{\exp(\sum_{i=1}^{n} \kappa * \cos(y_i - \mu))}{(2\pi I_0(\kappa))^n} * \frac{1}{2}\exp(-\kappa/2)$$

which can be simplified to :

$$p(\kappa|y,\mu) \propto f(\kappa) = \frac{\exp(\kappa * \sum_{i=1}^{n} \cos(y_i - \mu) - \kappa/2)}{(I_0(\kappa))^n}$$

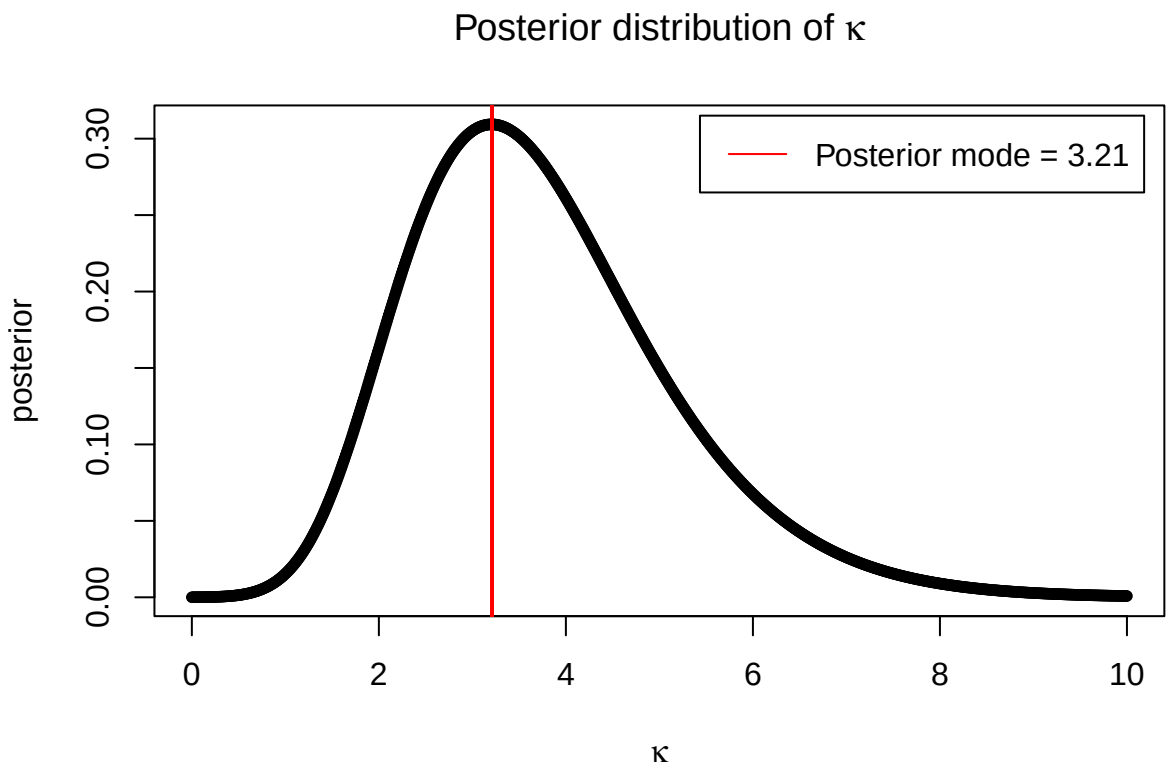
```
obs = c(-2.79, 2.33, 1.83, -2.44, 2.23, 2.33, 2.07, 2.02, 2.14, 2.54)
n = length(obs)
mu = 2.4

minimum = 0
maximum = 10
delta = 0.01
kappaGrid <- seq(minimum, maximum, by = delta)

posterior_func <- function(kappa, mu, y){
  n = length(y)
  (exp(kappa * sum(cos(y - mu))) - kappa/2)/(besselI(kappa, nu = 0))^n
}
posterior = posterior_func(kappaGrid, mu, obs)
#normalizing:
posterior = posterior/(sum(posterior)*delta)

plot(posterior ~ kappaGrid, pch = 20,
     main = expression(paste("Posterior distribution of ", kappa, " for the wind direction data")),
     xlab=expression(kappa), ylab = "density")
```



3.b

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

```
idx_mode = which.max(posterior)
post_mode = kappaGrid[idx_mode]

plot(posterior ~ kappaGrid, pch = 20,
     xlab = expression(kappa), main = expression(paste("Posterior distribution of ", kappa)))
abline(v = post_mode, col = "red", lwd=2)
legend("topright", inset=.02, legend=paste0("Posterior mode = ", post_mode), col=c("red"), lty=1)
```



The posterior mode can be approximated by finding the point kappa with the highest density. The posterior mode is approximated to be 3.21, visualized by the red vertical line on the plot.

Lab 2

Linear and polynomial regression

The dataset Linkoping2022.xlsx contains daily average temperatures (in degree Celcius) in Linköping over the course of the year 2022. Use the function read_xlsx(), which is included in the R package readxl (install.packages("readxl")), to import the dataset in R. The response variable is temp and the covariate time that you need to create yourself is defined by:

$$time = \frac{the\ number\ of\ days\ since\ the\ beginning\ of\ the\ year}{365}$$

A Bayesian analysis of the following quadratic regression model is to be performed:

$$temp = \beta_0 + \beta_1 * time + \beta_2 * time^2 + \epsilon, \epsilon \overset{\text{iid}}{\sim} N(0, \sigma^2).$$

```
library(readxl)
data = as.data.frame(read_xlsx("Linkoping2022.xlsx"))
#365 obs of 3 variables (name, datetime, temp)
time = c(0:364)/365
data$time <- time
```

a)

Use the conjugate prior for the linear regression model. The prior hyperparameters μ_0, Ω_0, ν_0 and σ^2 shall be set to sensible values. Start with $\mu_0 = (0, 100, -100)^T$, $\Omega_0 = 0.01 * I_3$, $\nu_0 = 1$ and $\sigma^2 = 1$. Check if this prior agrees with your prior opinions by simulating draws from the joint prior of all parameters and for every draw compute the regression curve. This gives a collection of regression curves; one for each draw from the prior. Does the collection of curves look reasonable? If not, change the prior hyperparameters until the collection of prior regression curves agrees with your prior beliefs about the regression curve. [Hint: R package mvtnorm can be used and your $Inv - \chi^2$ simulator of random draws from Lab 1.]

Conjugate prior for the linear regression:

- Joint prior for β and σ^2 : $\beta|\sigma^2 \sim N(\mu_0, \sigma^2\Omega_0^{-1})$ where $\sigma^2 \sim Inv - \chi^2(\nu_0, \sigma_0^2)$
- Posterior : $\beta|\sigma^2, y \sim N(\mu_n, \sigma^2\Omega_n^{-1})$ and $\sigma^2|y \sim Inv - \chi^2(\nu_n, \sigma_n^2)$

$\mu_n = (X'X + \Omega_0)^{-1}(X'X\hat{\beta} + \Omega_0\mu_0)$

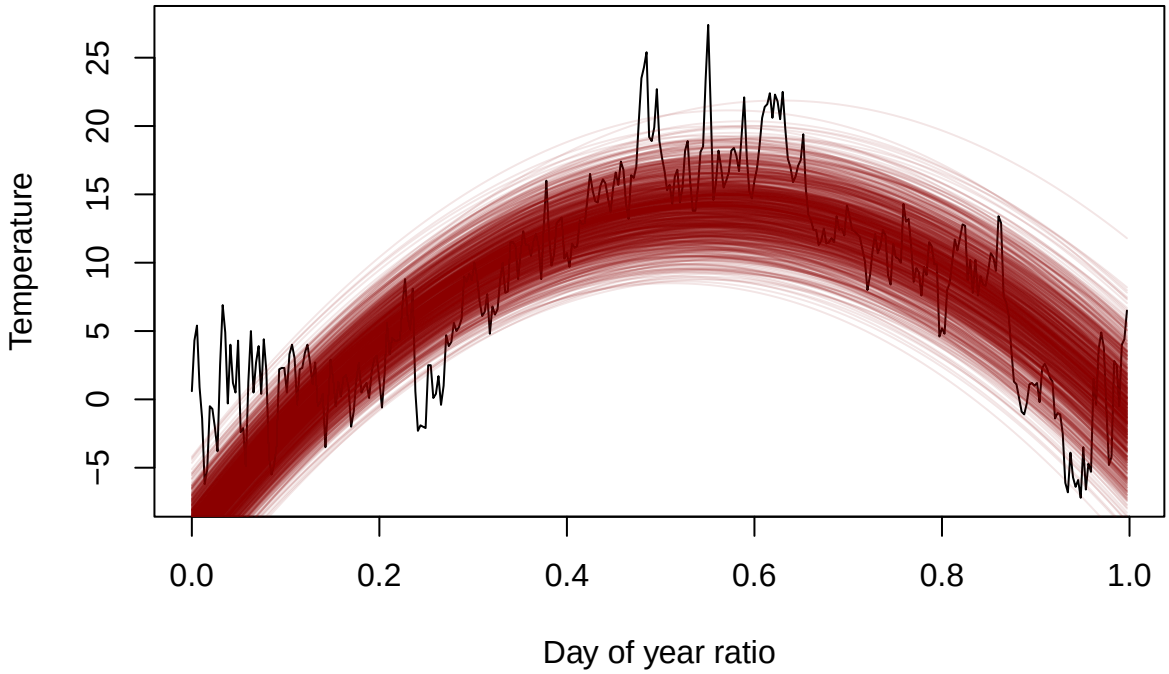
$\Omega_n = X'X + \Omega_0$

$\nu_n = \nu_0 + n$

$\nu_n\sigma_n^2 = \nu_0\sigma_0^2 + (y'y + \mu_0'\Omega_0\mu_0 - \mu_n'\Omega_n\mu_n)$

```
library(scales)
library(mvtnorm)
# Initializing the parameters
mu_0 = c(-10,87,-78) #(y intercept, next two have to be balanced. High value -> high arch, vice versa)
Omega_0 = as.matrix(0.01 * diag(3)) #thickness (higher = thinner)
nu_0 = 3 #Vertical thickness (higher = thicker)
sigma_sq_0 = 4 #vertical thickness (higher = thicker)
n = 365

X = matrix(c(time*0+1, time, time*time), ncol=3)
y = data$temp

#Prior
sigsq_draws_prior = as.matrix((nu_0*sigma_sq_0) / rchisq(1000, n-1))
beta_draws_prior = matrix(NA,nrow=0, ncol=3)
for(i in (1:length(sigsq_draws_prior))){
  sig = sigsq_draws_prior[i]
  beta_draws_prior = rbind(beta_draws_prior,rmvnorm(n = 1, mean = mu_0, sigma = sig*solve(Omega_0)))
}


#Plot
f = function(x,betas,sigsq){
  betas[1] + betas[2]*x + betas[3]*x^2 + rnorm(1, 0, sigsq)
}

x = time
betas = beta_draws_prior[1,]
plot(x, y, type = 'l', xlab="Day of year ratio", ylab = "Temperature")
for(i in 1:1000){
  x = time
  betas = beta_draws_prior[i,]
```

```
  sigsq = sigsq_draws_prior[i,]
  lines(x, f(x, betas, sigsq), type = 'l', col=alpha('darkred', 0.1))
}
```



The collection of curves looks reasonable for most of the data. Only the beginning of the year is not well represented by the prior, but it is hard to find a prior that fits the data perfectly.

b)

Write a function that simulate draws from the joint posterior distribution of β_0, β_1,β_2 and σ^2.

```
#Posterior
beta_draws_post = matrix(0,nrow=0, ncol=3)
sigsq_draw_post = matrix(0,nrow=0, ncol=1)
for(i in 1:1000){
  beta_hat = beta_draws_prior[i,]
  mu_n = solve(t(X) %*% X + Omega_0)%*%(t(X) %*% X %*% beta_hat + Omega_0 %*% mu_0)
  Omega_n = t(X)%*%X + Omega_0
  nu_n = nu_0 + n
  nu_n_sig_n = nu_0%*%sigma_sq_0+(t(y)%*%y + t(mu_0)%*%Omega_0%*%mu_0 - t(mu_n)%*%Omega_n%*%mu_n)

  sig = nu_n_sig_n / rchisq(1, n-1)
  sigsq_draw_post = rbind(sigsq_draw_post, sig)

  beta_draws_post = rbind(beta_draws_prior,
```

```
                              rmvnorm(n=1,mean=mu_n, sigma = sigsq_draw_post[1,]*solve(Omega_n)))
}
```

```
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
```

```
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
```

```
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
```

```
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
```

```
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
```

```
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
```

```
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
```

```
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
```

```
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
```

```
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
```

```
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
```

```
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
```

```
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
```

```
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
```

```
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
```

```
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
```

```
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
```

```
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
```

```
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
```

```
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
```

```
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
```

```
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
```

```
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
```

```
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
```

```
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
```

```
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
```

```
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
```

```
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
```

```
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
```

```
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
```

```
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
```

```
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
```

```
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
```

```
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
```

```
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
```

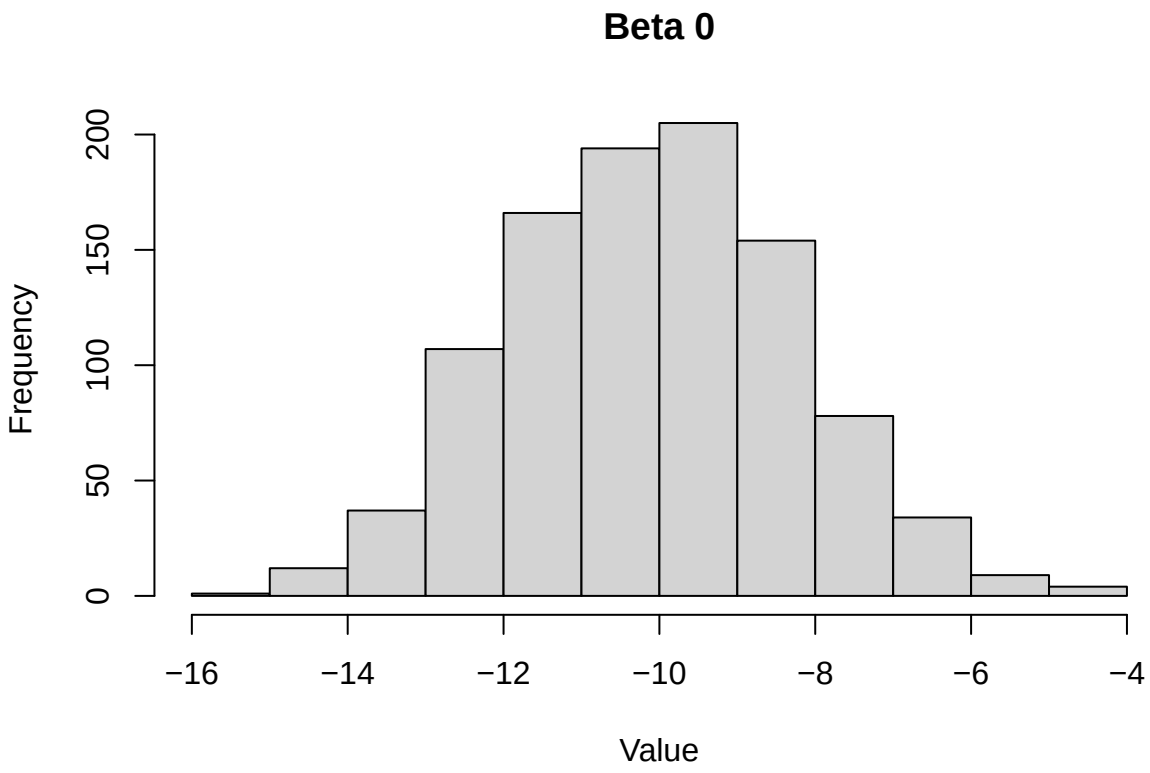
```
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
```

```
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
```

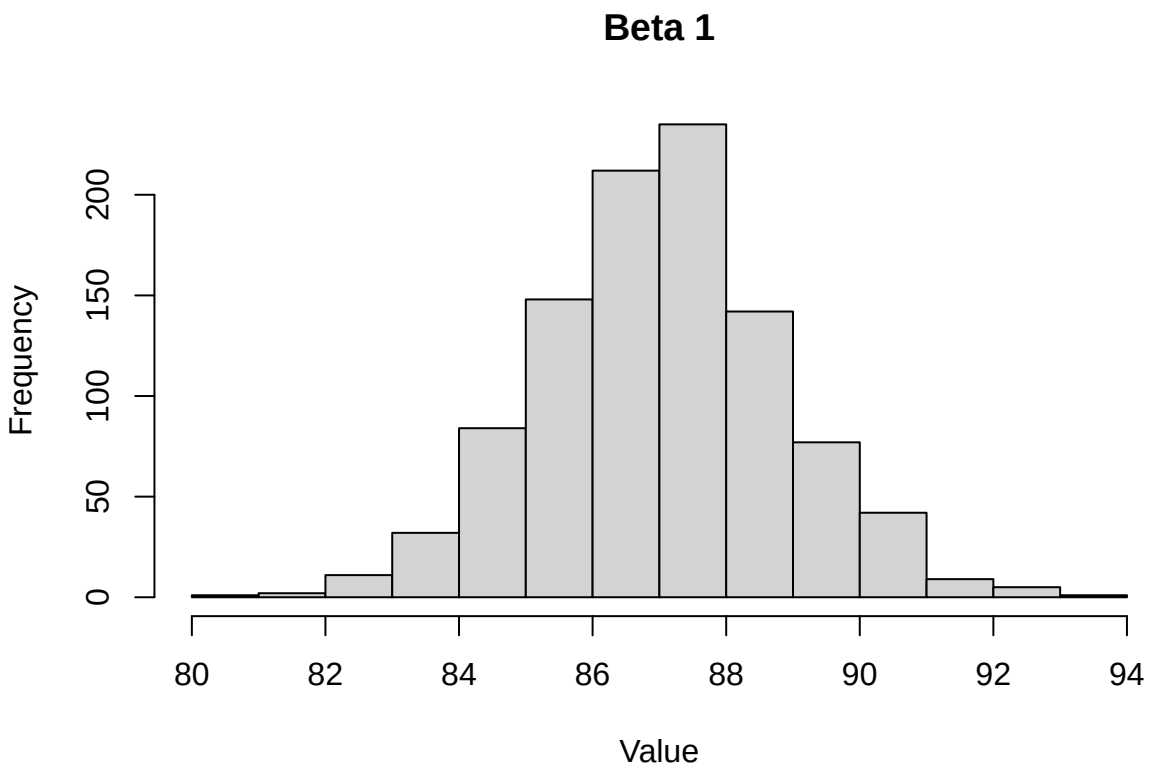
```
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
## Warning in rmvnorm(n = 1, mean = mu_n, sigma = sigsq_draw_post[1, ] *
## solve(Omega_n)): sigma is numerically not positive semidefinite
```

i) Plot a histogram for each marginal posterior of the parameters.

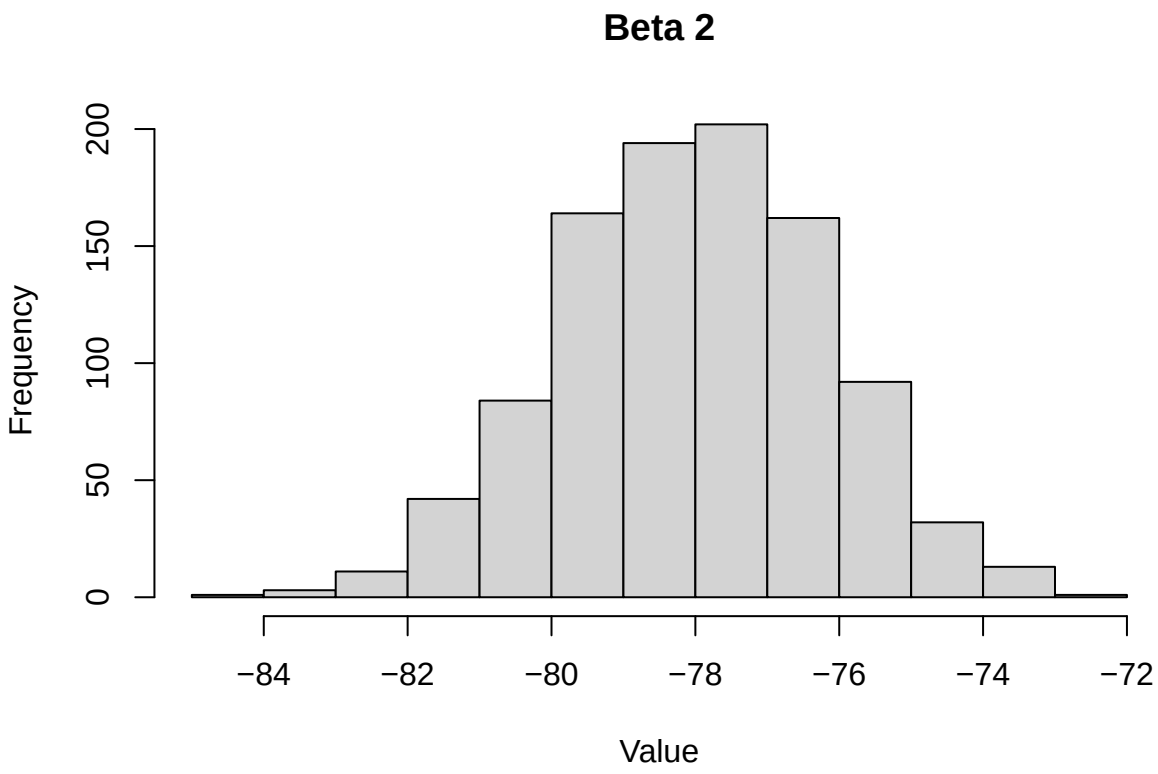
```
#histogram for each marginal posterior of the parameters
hist(beta_draws_post[,1], main="Beta 0", xlab="Value")
```



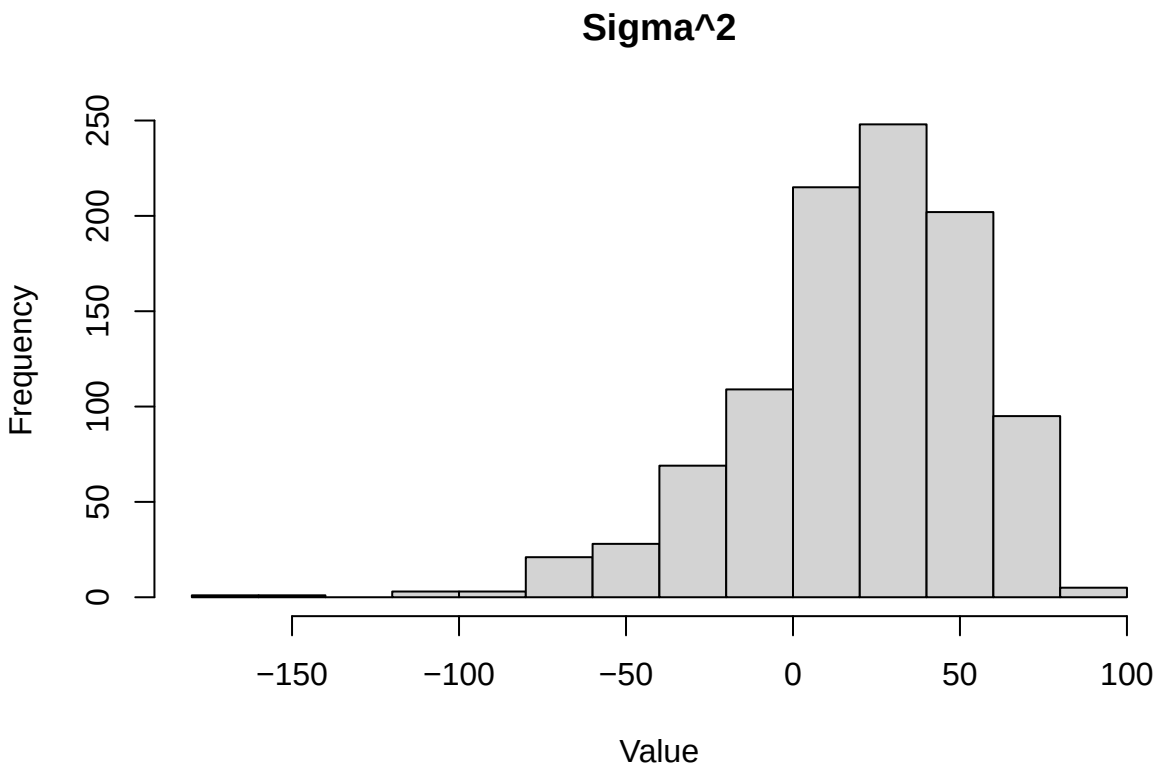

```
hist(beta_draws_post[,2], main="Beta 1", xlab="Value")
```




```
hist(beta_draws_post[,3], main="Beta 2", xlab="Value")
```

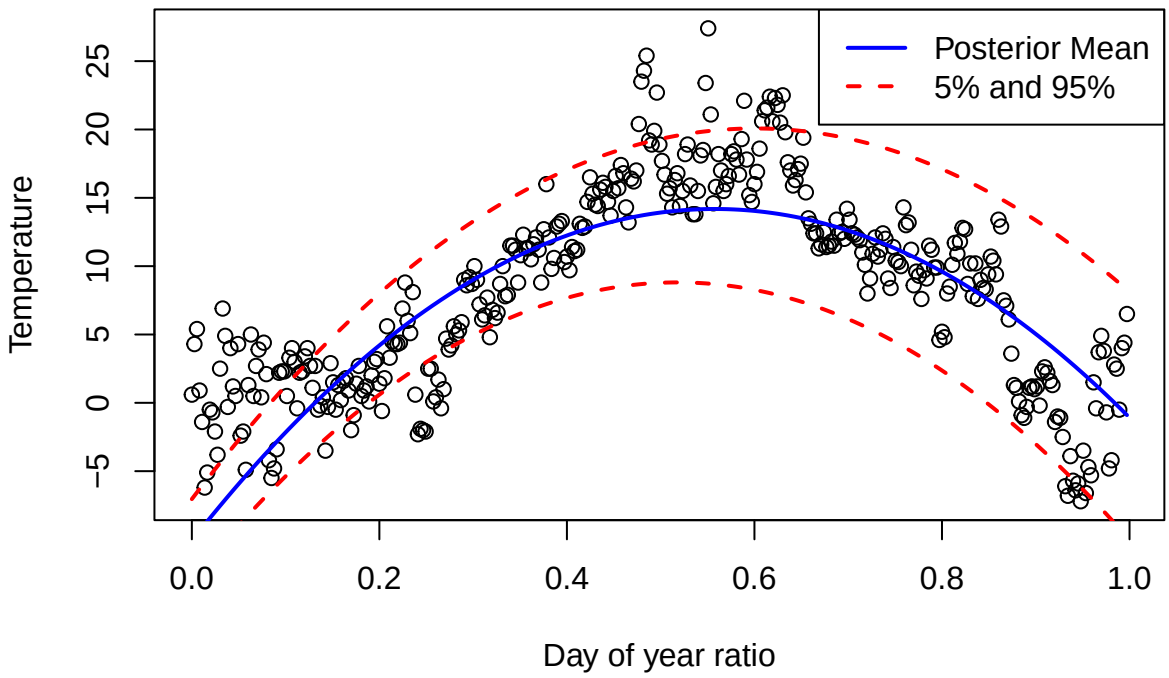



```
hist(sigsq_draw_post, main="Sigma^2", xlab="Value")
```



ii) Make a scatter plot of the temperature data and overlay a curve for the posterior median of the regression function $f(time) = E[temp|time] = \beta_0 + \beta_1 * time + \beta_2 * time^2$, i.e. the median of f (time) is computed for every value of time. In addition, overlay curves for the 90% equal tail posterior probability intervals of f (time), i.e. the 5 and 95 posterior percentiles of f (time) is computed for every value of time. Does the posterior probability intervals contain most of the data points? Should they?

```
#Scatter plot of the temperature data + curve for the posterior median of the regression function
plot(time, data$temp, type = 'p', xlab="Day of year ratio", ylab = "Temperature")
#Mean
betas_mean = colMeans(beta_draws_post)
lines(time, betas_mean[1] + betas_mean[2]*time + betas_mean[3]*time^2, type = 'l', col='blue', lwd=2)

#5 and 95 percentiles for each beta for each time
betas_5 = c(quantile(beta_draws_post[,1], 0.05),
            quantile(beta_draws_post[,2], 0.05),
            quantile(beta_draws_post[,3], 0.05))

betas_95 = c(quantile(beta_draws_post[,1], 0.95),
             quantile(beta_draws_post[,2], 0.95),
             quantile(beta_draws_post[,3], 0.95))

lines(time, betas_5[1] + betas_5[2]*time + betas_5[3]*time^2, type = 'l', col='red', lty=2, lwd=2)
lines(time, betas_95[1] + betas_95[2]*time + betas_95[3]*time^2, type = 'l', col='red', lty=2, lwd=2)
legend("topright", legend=c("Posterior Mean", "5% and 95%"), col=c("blue", "red"), lty=c(1,2),
       lwd=c(2,2))
```



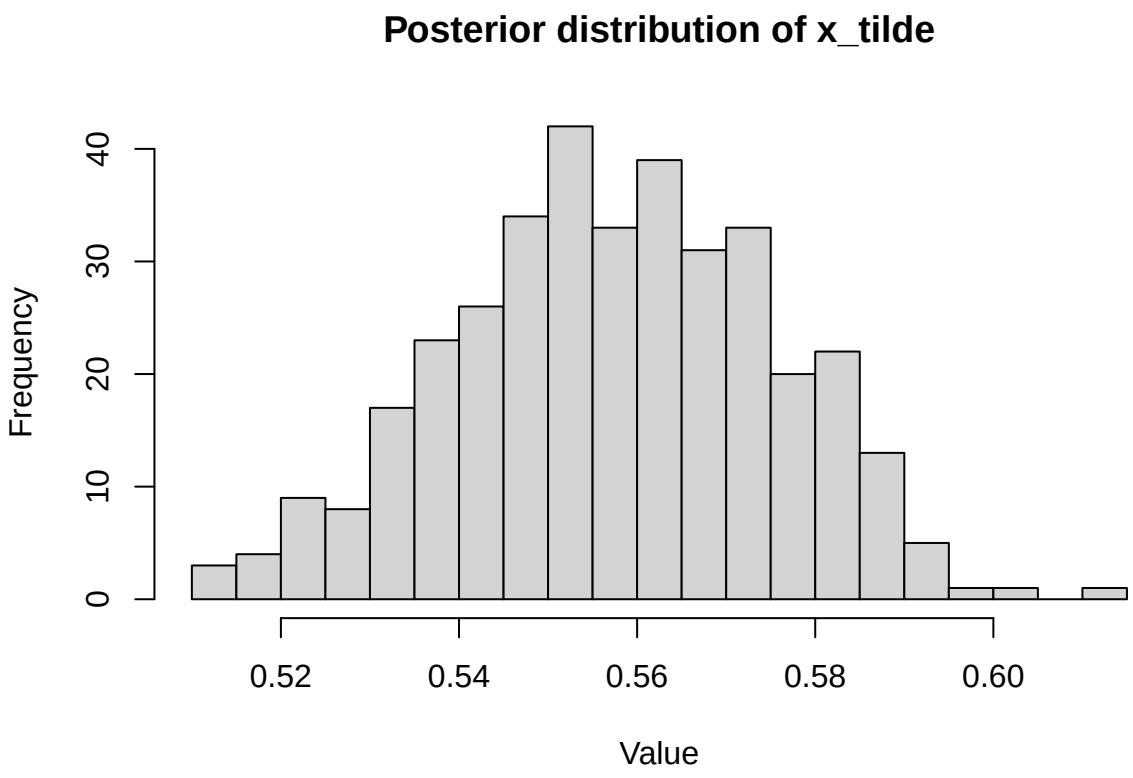
The posterior probability intervals contain most of the data points, but they are not perfect. They should contain most of the data points if the temperature in a year followed our model, which is a quadratic polynomial. However, the reality is different so it is not surprising that the intervals do not contain all the data points, especially where the data is noisy.

c)

It is of interest to locate the time with the highest expected temperature (i.e. the time where f (time) is maximal). Let's call this value $\tilde{x}$. Use the simulated draws in (b) to simulate from the posterior distribution of $\tilde{x}$. [Hint: the regression curve is a quadratic polynomial. Given each posterior draw of β_0, β_1 and β_2 , you can find a simple formula for $\tilde{x}$.]

To find the maximum of a quadratic polynomial, we can use the formula $x = -\frac{\beta_1}{2\beta_2}$ which is the maximum of the quadratic polynomial $f(x) = \beta_0 + \beta_1 x + \beta_2 x^2$ (derivative of f(x) equal to 0). We can use this formula to find the maximum temperature for each draw of β_0, β_1 and β_2 and simulate the posterior distribution of $\tilde{x}$, and also calculate the mean of these maximum temperatures.

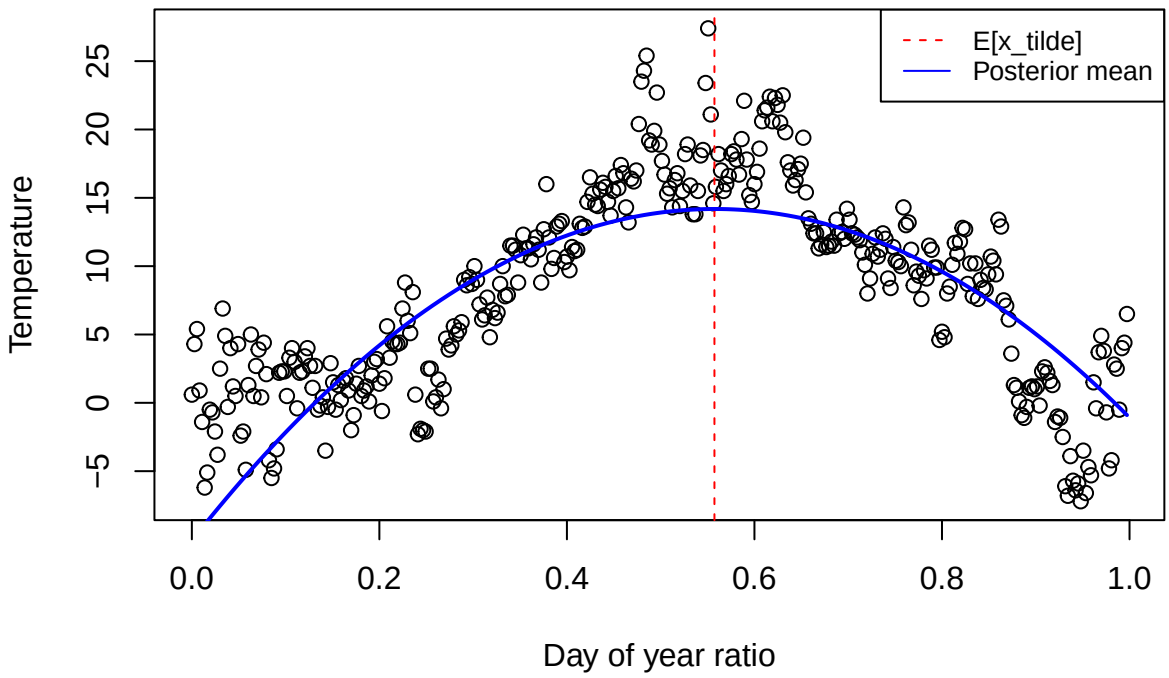
```
max_temp <- function(betas){
  (-betas[2])/(2*betas[3])
}
x_maxs = matrix(0,nrow=0, ncol=1)
for(i in 1:365){
  betas = beta_draws_post[i,]
  x_maxs = rbind(x_maxs, max_temp(betas))
}
hist(x_maxs, main="Posterior distribution of x_tilde", xlab="Value", ylab="Frequency", breaks=30)
```




```
x_tilde = mean(x_maxs)
print(paste("E[x_tilde] is",x_tilde))
```

```
## [1] "E[x_tilde] is 0.557318518295331"
```

```
plot(time, data$temp, type = 'p', xlab="Day of year ratio", ylab = "Temperature")
abline(v=x_tilde, col='red', lty=2)
legend("topright", legend=c("E[x_tilde]", "Posterior mean"), col=c("red","blue"),
       lty=c(2,1), cex=c(0.8,0.8))
lines(time, betas_mean[1] + betas_mean[2]*time + betas_mean[3]*time^2, type = 'l', col='blue', lwd=2)
```



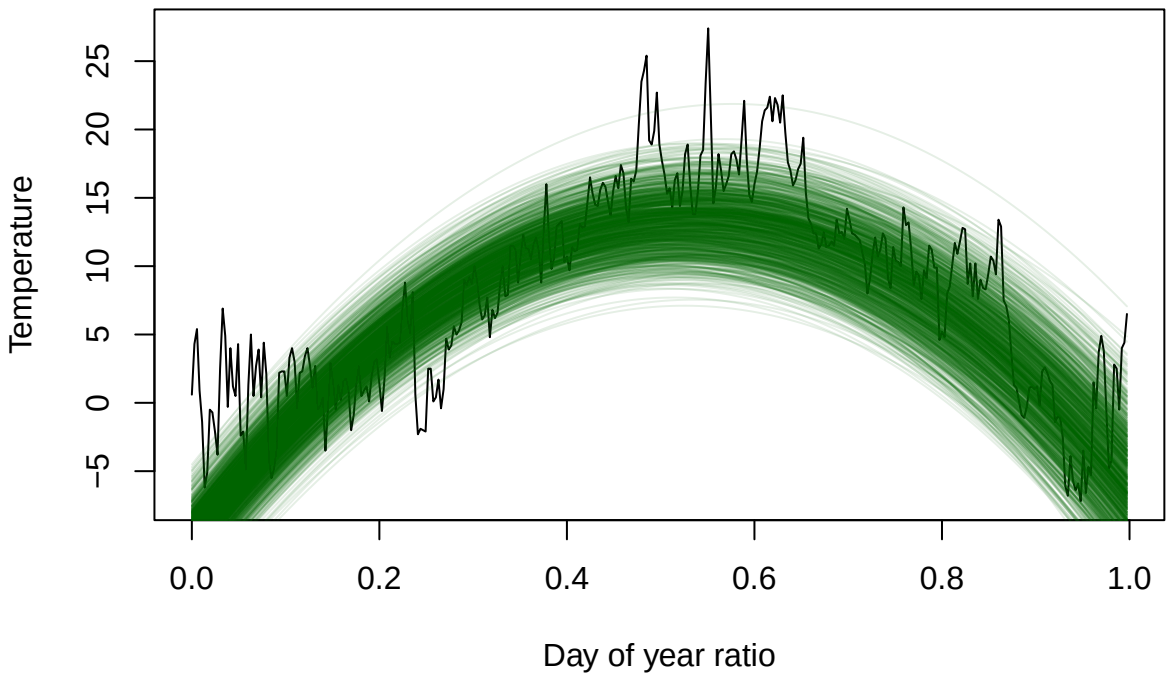
d)

Say now that you want to estimate a polynomial regression of order 10, but you suspect that higher order terms may not be needed, and you worry about overfitting the data. Suggest a suitable prior that mitigates this potential problem. You do not need to compute the posterior. Just write down your prior. [Hint: the task is to specify μ_0 and Ω_0 in a suitable way.]

Because we suspect that higher order terms may not be needed (the data seems to follow a quadratic path), and it might cause overfitting, a prior that has null/close to null mean for higher order terms, and a small variance (high omegas) is suggested. This way, the higher order terms will not have a big impact on the model. We keep the mean of the first three terms the same as before, as the previous questions showed that the quadratic polynomial is a good fit for most of the data. The posterior is recalculated for sanity check.

```
# Initializing the parameters

##### MODIFIED #######
mu_0 = c(-10,87,-78, -5, 0, 0, 0, 0, 0, 0, 0) #(y, beta_0, ..., beta_9)
Omega_0 = diag(c(rep(0.01,4), rep(100,7))) # Small variance (high omega) for higher order terms
#######################

nu_0 = 3 #Vertical thickness (higher = thicker)
sigma_sq_0 = 4 #vertical thickness (higher = thicker)
n = 365

X = matrix(c(time*0+1, time, time*time), ncol=3)
y = data$temp
```

```r
#Prior
sigsq_draws_prior = as.matrix((nu_0*sigma_sq_0) / rchisq(1000, n-1))

########## MODIFIED ##########
beta_draws_prior = matrix(NA,nrow=0, ncol=11)
##############################

for(i in (1:length(sigsq_draws_prior))){
  sig = sigsq_draws_prior[i]
  beta_draws_prior = rbind(beta_draws_prior,rmvnorm(n = 1, mean = mu_0, sigma = sig*solve(Omega_0)))
}

#Plot
########## MODIFIED ##########
f = function(x,betas,sigsq){
  betas[1] + betas[2]*x + betas[3]*x^2 + betas[4]*x^3 + betas[5]*x^4 + betas[6]*x^5 + betas[7]*x^6 + be
}
##############################

x = time
betas = beta_draws_prior[1,]
plot(x, y, type = 'l', xlab="Day of year ratio", ylab = "Temperature")
for(i in 1:1000){
  x = time
  betas = beta_draws_prior[i,]
  sigsq = sigsq_draws_prior[i,]
  lines(x, f(x, betas, sigsq), type = 'l', col=alpha('darkgreen', 0.1))
}
```



2. Posterior approximation for classification with logistic regression

The dataset WomenAtWork.dat contains n = 132 observations on the following eight variables related to women (see table in lab compendium).

a)

Consider the logistic regression model:

$$Pr(y = 1|x, \beta) = \frac{exp(x^T\beta)}{1 + \exp(x^T\beta)},$$

where y equals 1 if the woman works and 0 if she does not. x is a 7-dimensional vector containing the seven features (including a 1 to model the intercept). The goal is to approximate the posterior distribution of the parameter vector β with a multivariate normal distribution

$$\beta|y, x \sim N(\tilde{\beta}, J_y^{-1}(\tilde{\beta})),$$

where $\tilde{\beta}$ is the posterior mode and $J_y^{-1} = -\frac{complete}{later}$ is the negative of the observed Hessian evaluated at the posterior mode. Note that $\frac{complete}{later}$ is a 7*7 matrix with second derivatives on the diagonal and cross derivatives *inserteq* on the off-diagonal. You can compute this derivative by hand, but we will let the computer do it numerically for you. Calculate both $\tilde{\beta}$ and $J(\tilde{\beta})$ by using the optim function in R. [Hint: You may use code snippets from my demo of logistic regression in Lecture 6.] Use the prior $beta \sim N(0, \tau^2 I)$, where $\tau = 2$.

Present the numerical values of $\tilde{\beta}$ and $J_y^{-1}(\tilde{\beta})$ for the WomenAtWork data. Compute an approximate 95% equal tail posterior probability interval for the regression coefficient to the variable NSmallChild. Would you say that this feature is of importance for the probability that a woman works? [Hint: You can verify that your estimation results are reasonable by comparing the posterior means to the maximum likelihood estimates, given by: r glmModel<- glm(Work ~ 0 + ., data = WomenAtWork, family = binomial).]

```
# Template code from Lecture material, modified for the WomenAtWork data set:

Covs <- c(2:8) # Selects which covariates/features to include
standardize <- FALSE # If TRUE, covariates/features are standardized to mean 0 and variance 1
lambda <- 1 # scaling factor for the prior of beta
tau <- 2

WomenData <- read.table("WomenAtWork.dat", header = T) # read data from .dat file
Nobs <- dim(WomenData)[1] # number of observations
y <- WomenData$Work

X <- as.matrix(WomenData[,Covs]);
Xnames <- colnames(X)
if (standardize){
  Index <- 2:(length(Covs)-1)
  X[,Index] <- scale(X[,Index])
}
Npar <- dim(X)[2]

# Setting up the prior
mu <- as.matrix(rep(0,Npar)) # Prior mean vector
Sigma <- (1/lambda)*(tau^2)*diag(Npar) # Prior covariance matrix

# Functions that returns the log posterior for the logistic and probit regression.
# First input argument of this function must be the parameters we optimize on,
# i.e. the regression coefficients beta.

# Log posterior for logistic regression
LogPostLogistic <- function(betas,y,X,mu,Sigma){
  linPred <- X%*%betas;
  logLik <- sum( linPred*y - log(1 + exp(linPred)) );
  #if (abs(logLik) == Inf) logLik = -20000; # Likelihood is not finite,
  # stear the optimizer away from here!
  logPrior <- dmvnorm(betas, mu, Sigma, log=TRUE);
  #posterior = likelihood*prior, log posterior = log likelihood + log prior
  return(logLik + logPrior)
}

# Select the initial values for beta
initVal <- matrix(0,Npar,1)

# The argument control is a list of options to the optimizer optim, where fnscale=-1
# means that we minimize the negative log posterior. Hence, we maximize the log posterior.
OptimRes <- optim(initVal,LogPostLogistic,gr=NULL,y,X,mu,Sigma,method=c("BFGS"),
                  control=list(fnscale=-1),hessian=TRUE)

# Printing the results to the screen
names(OptimRes$par) <- Xnames # Naming the coefficient by covariates
```

```r
approxPostStd <- sqrt(diag(solve(-OptimRes$hessian))) # Computing approximate standard deviations.
names(approxPostStd) <- Xnames # Naming the coefficient by covariates
print('---- The posterior mode is: ----')
```

```
## [1] "---- The posterior mode is: ----"
```

```r
print(OptimRes$par)
```

```
##             [,1]
## [1,] -0.04036943
## [2,] -0.03730689
## [3,]  0.17868950
## [4,]  0.12073637
## [5,] -0.04618995
## [6,] -1.47248930
## [7,] -0.02014458
## attr(,"names")
## [1] "Constant"    "HusbandInc"  "EducYears"   "ExpYears"    "Age"
## [6] "NSmallChild" "NBigChild"
```

```r
print('---- The approximate posterior standard deviation is: ----')
```

```
## [1] "---- The approximate posterior standard deviation is: ----"
```

```r
print(approxPostStd)
```

```
##    Constant  HusbandInc   EducYears    ExpYears         Age NSmallChild
##  1.38198486  0.02198474  0.08920960  0.03335982  0.02747315  0.47746764
##   NBigChild
##  0.16401959
```

```r
glmModel <- glm(Work ~ 0 + ., data = WomenData, family = binomial)
print("---- comparing with GLM ----")
```

```
## [1] "---- comparing with GLM ----"
```

```r
print(glmModel)
```

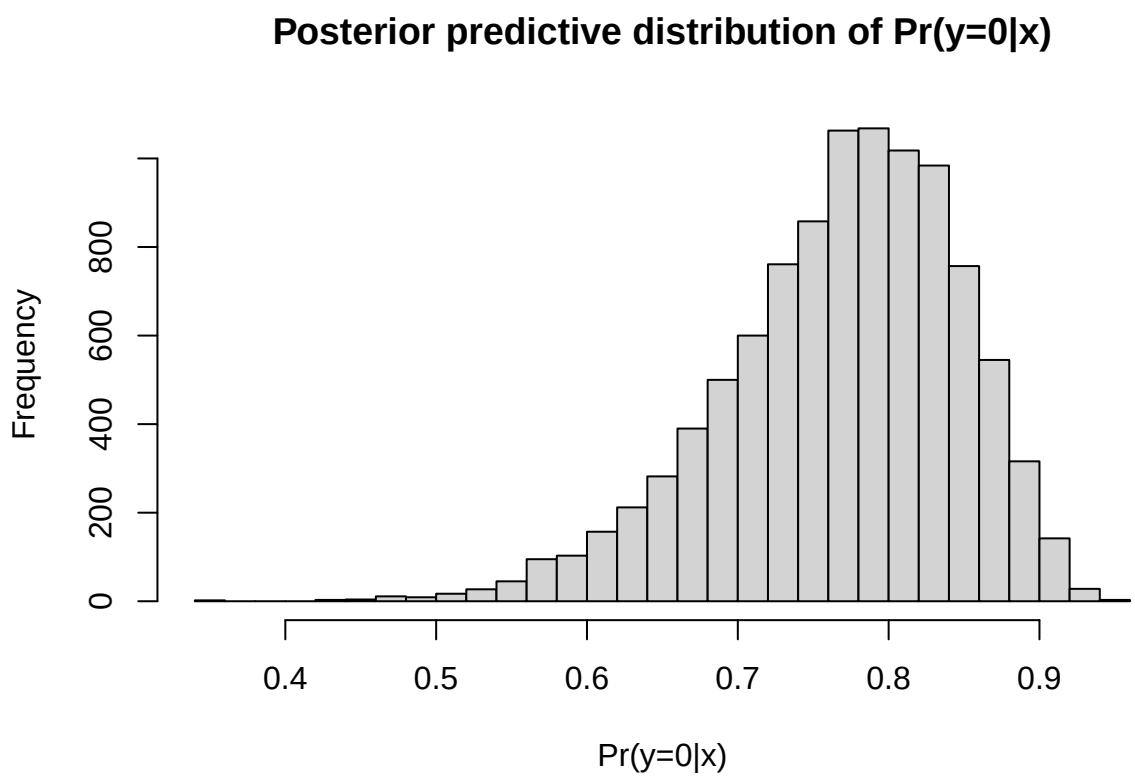
```
##
## Call:  glm(formula = Work ~ 0 + ., family = binomial, data = WomenData)
##
## Coefficients:
##    Constant   HusbandInc    EducYears     ExpYears          Age  NSmallChild
##     0.02263     -0.03796      0.18447      0.12132     -0.04858     -1.56485
##   NBigChild
##    -0.02526
##
## Degrees of Freedom: 132 Total (i.e. Null);  125 Residual
## Null Deviance:       183
## Residual Deviance: 146.7     AIC: 160.7
```

NSmallChild has the biggest absolute value out of all the posterior means, hence it is the feature with the most importance for the probability that a woman works.

b)

Use your normal approximation to the posterior from (a). Write a function that simulate draws from the posterior predictive distribution of $Pr(y = 0|x)$, where the values of x corresponds to a 40-year-old woman, with two children (4 and 7 years old), 11 years of education, 7 years of experience, and a husband with an income of 18. Plot the posterior predictive distribution of $Pr(y = 0|x)$ for this woman. [Hints: The R package mvtnorm will be useful. Remember that $Pr(y = 0|x)$ can be calculated for each posterior draw of β .]

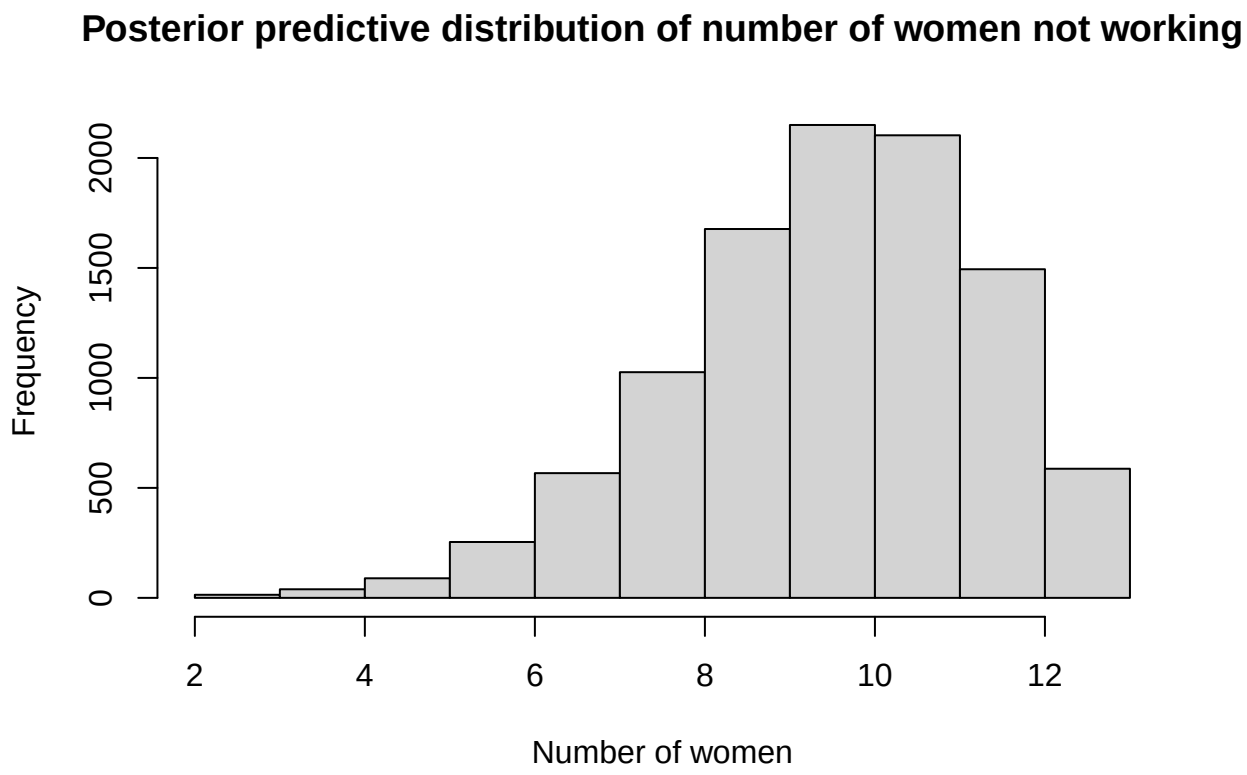
```
SimulatePostPred <- function(Nsim,OptimRes,X){
  Npar <- dim(X)[2]
  postMean <- OptimRes$par #Posterior mean
  postCov <- solve(-OptimRes$hessian) #Posterior covariance matrix
  postDraws <- rmvnorm(Nsim, postMean, postCov) #Draws from the posterior
  postPred <- rep(0,Nsim)
  for (i in 1:Nsim){
    postPred[i] <- 1/(1+exp(X%*%postDraws[i,])) # Pr(y=0|x)
  }
  return(postPred)
}

Nsim <- 10000
x <- c(1, 18, 11, 7, 40, 1, 1) #cst, HusbandIncome, Education, Experience, Age, NSmallChild, NBigChild
postPred <- SimulatePostPred(Nsim,OptimRes,x)
hist(postPred,main="Posterior predictive distribution of Pr(y=0|x)",xlab="Pr(y=0|x)",ylab="Frequency",
     breaks=30)
```



c)

Now, consider 13 women which all have the same features as the woman in (b). Rewrite your function and plot the posterior predictive distribution for the number of women, out of these 13, that are not working. [Hint: Simulate from the binomial distribution, which is the distribution for a sum of Bernoulli random variables.]

```
SimulatePostPredBin <- function(Nsim,OptimRes,X,Nwomen){
  Npar <- dim(X)[2]
  postMean <- OptimRes$par #Posterior mean
  postCov <- solve(-OptimRes$hessian) #Posterior covariance matrix
  postDraws <- rmvnorm(Nsim, postMean, postCov) #Draws from the posterior
  postPred <- rep(0,Nsim)
  for (i in 1:Nsim){
    postPred[i] <- rbinom(1, Nwomen, 1 -1/(1+exp(-X%*%postDraws[i,]))) # Pr(y=0|x) = 1 - Pr(y=1|x)
  }
  return(postPred)
}

Nsim <- 10000
Nwomen <- 13
postPred <- SimulatePostPredBin(Nsim,OptimRes,x,Nwomen)
hist(postPred,main="Posterior predictive distribution of number of women not working",
     xlab="Number of women",ylab="Frequency")
```



Lab 3

1. Gibbs sampling for logistic regression

Consider again the logistic regression model in problem 2 from the previous computer lab 2. Use the prior $\beta \sim N(0, \tau^2 I)$, where $\tau = 3$.

a)

Implement (code!) a Gibbs sampler that simulates from the joint posterior $p(\omega, \beta|x)$ by augmenting the data with Polya-gamma latent variables ω_i , i = 1,. . . , n. The full conditional posteriors are given on the slides from Lecture 7. Evaluate the convergence of the Gibbs sampler by calculating the Inefficiency Factors (IFs) and by plotting the trajectories of the sampled Markov chains.

From the lecture: Gibbs sampling for the Logistic regression:

Given $\omega = (\omega_1, \omega_2, ..., \omega_n)$, the conditional posterior of β with prior $\beta \sim N(b, B)$ follows a multivariate normal distribution. ω is not observed –> Gibbs sampling to the rescue. Simulate from the joint posterior $p(\omega, \beta|y)$ by iterating between :

- $\omega_i|\beta \sim PG(1, x_i^T\beta)$, i = 1. . . n
- $$\beta|y, \omega \sim N(m_\omega, V_\omega)$$

  ,

  $$V_\omega = (X^T\Omega X + B^{-1})^{-1}$$

  ,

  $$m_\omega = V_\omega(X^T\kappa + B^{-1}b)$$

  ,

where $\kappa = (y_i - 0.5, ..., y_n - 0.5)$, Ω is the diagonal matrix of $\omega_i's$ and $PG(1, x_i^T\beta)$ is the poly-gamma sampler from the BayesLogit R package.

From Lecture 7 slide 23:

Logistic regression:

$$Pr(y_i = 1|x_i, \beta) = \frac{exp(x_i^T\beta)}{1 + exp(x_i^T\beta)}$$

The posterior distribution is not known. Augment the data with Polya-gamma latent variables ω_i , i=1,. . . ,n.

$$\omega_i = \frac{1}{2 * \pi^2} * \sum_{k=1}^{\infty} \frac{g_k}{(k - 0.5)^2 + \frac{(x_i^T\beta)^2}{4\pi^2}}$$

, where g_k are independent draws from the exponential distribution with mean 1.

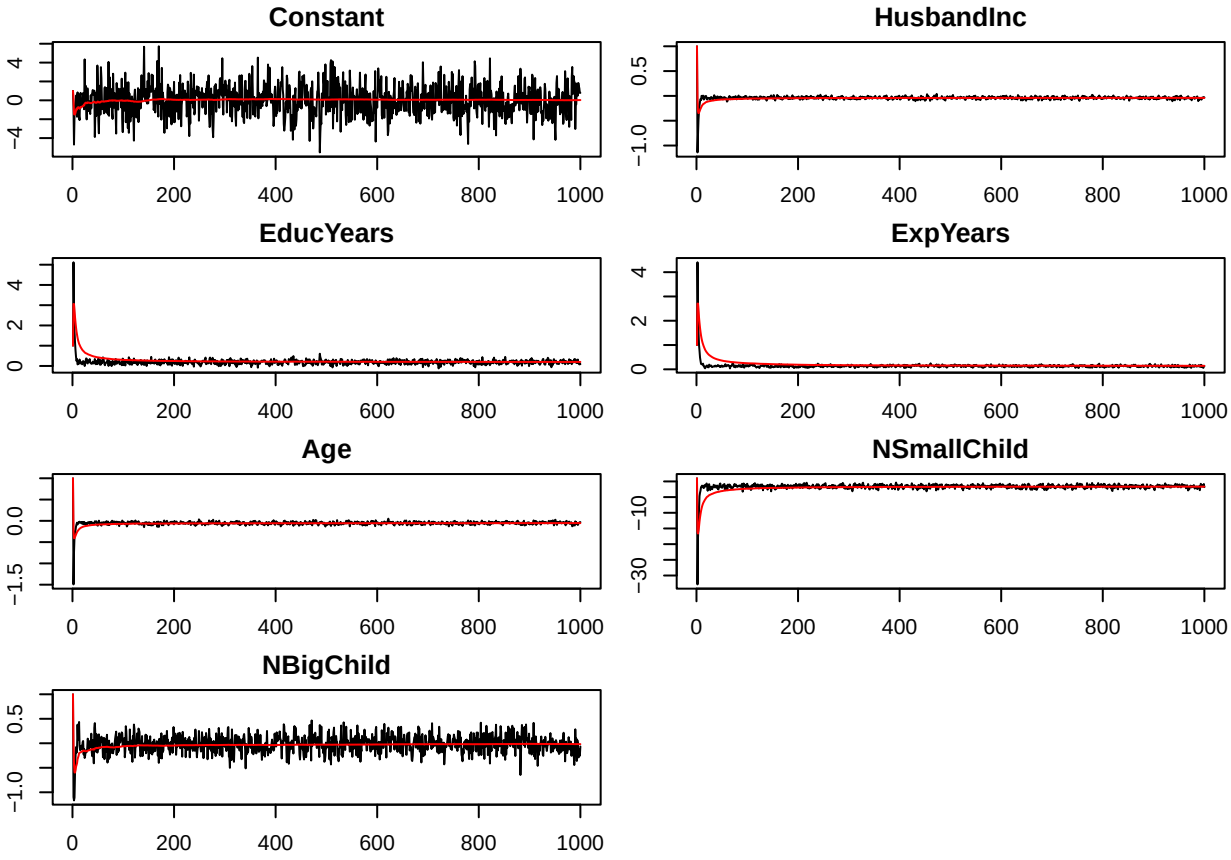
```
library(BayesLogit)
library(mvtnorm)

WomenData <- read.table("WomenAtWork.dat", header = T) # read data from .dat file
Nobs <- dim(WomenData)[1] # number of observations
y <- WomenData$Work
```

```
Covs <- c(2:8) # Selects which covariates/features to include
X <- as.matrix(WomenData[,Covs]);
Xnames <- colnames(X)
nPar <- dim(X)[2] #7

# Setting up the prior

nDraws = 1000

tau <- 3
b = as.matrix(rep(0,nPar)) # Prior mean vector, needed to calculate m_omega
B = tau^2*diag(nPar) #Needed to calculate V_omega and m_omega, prior covariance matrix
kappa <- as.matrix(y-0.5) #Needed to calculate m_omega

omega_i_given_beta = matrix(0, nrow = nrow(X), ncol=nDraws)

betas = matrix(nrow=nDraws, ncol=nPar)
betas[1,] = rep(1,nPar) # Initial betas values, should it be from N(b,B) ?

for(draw in 2:nDraws){ #Start at 2 because we have set 1 above

  #First, augment the data with Poly-gamma latent variables
  for(row in 1:nrow(X)){
    omega_i_given_beta[row, draw-1] = rpg(1, h=1, X[row,]%*%betas[draw-1,])
  }
  #Omega is the diagonal matrix of the omega_i's
  Omega = diag(omega_i_given_beta[,draw-1])
  # Calculate V_omega and m_omega,
  V_omega = solve(t(X) %*% Omega %*% X + solve(B))
  m_omega = V_omega%*%(t(X) %*% kappa + solve(B) %*% b)

  #Finally, calculate beta given y and omega
  betas[draw,] = rmvnorm(1, m_omega, V_omega)
}

#Inefficency Factor
IF_Gibbs = c(rep(0,7))
for(beta in 1:7){
  IF_Gibbs[beta] = 1+2*colSums(colSums(acf(betas[-1,beta], plot=FALSE)$acf))
}
for(i in 1:7){
  print(paste("IF for ", Xnames[i], " : ", IF_Gibbs[i]))
}
```

```
## [1] "IF for  Constant  :  3.35946787822807"
## [1] "IF for  HusbandInc  :  5.30362848589069"
## [1] "IF for  EducYears  :  4.86713479467551"
## [1] "IF for  ExpYears  :  5.8596576894384"
## [1] "IF for  Age  :  4.64379595846253"
## [1] "IF for  NSmallChild  :  5.56998714098952"
## [1] "IF for  NBigChild  :  3.81917143529924"
```

Plotting the trajectories of the sampled Markov chains in black with the cumulative mean in red for each of the 7 parameters:

```r
#Plotting the trajectories + cummulative sums of the sampled Markov chains
cumsumData = matrix(nrow=nDraws, ncol=nPar)
for(i in 1:nPar){
  cumsumData[,i] = cumsum(betas[,i])/seq(1,nDraws)
}
par(mar=c(2,2,2,1))
par(mfrow=c(4,2))
for (i in 1:7){
  plot(1:nDraws, betas[,i], type = "l", main=Xnames[i])
  lines(1:nDraws, cumsumData[,i],type="l", col="red")
}
```

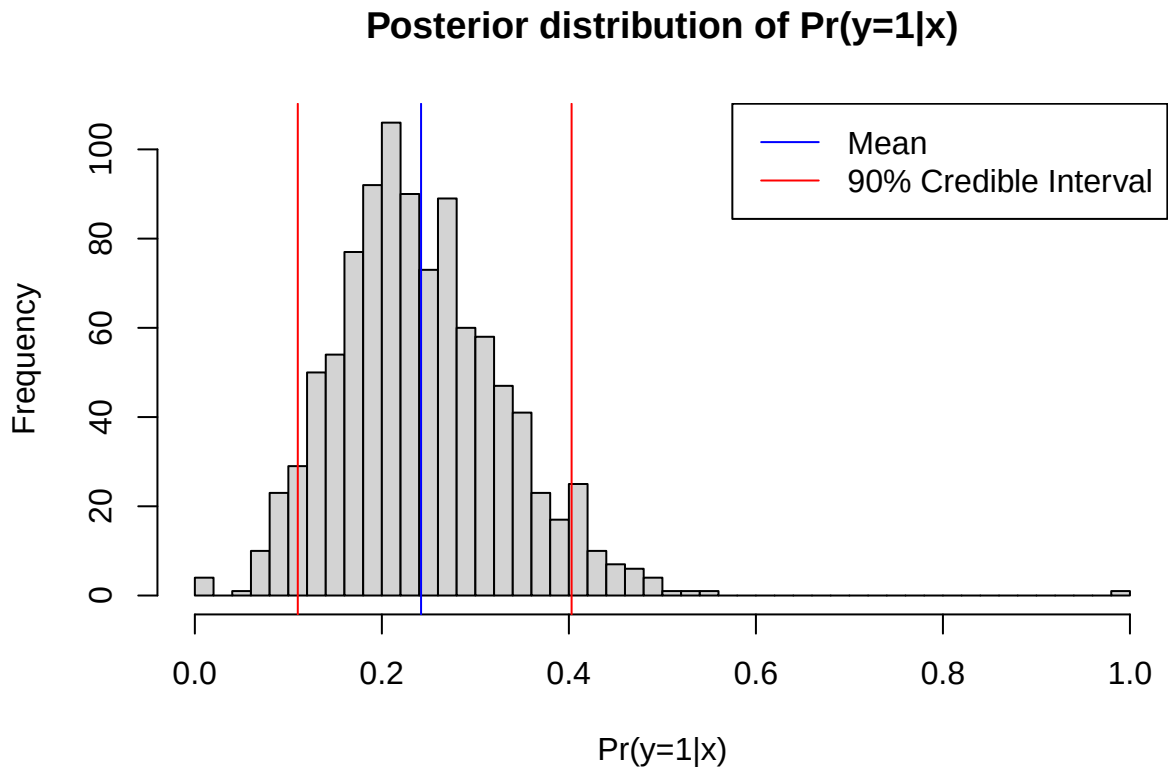


b)

Use the posterior draws from a) to compute a 90% equal tail credible interval for Pr(y = 1|x), where the values of x corresponds to a 38-year-old woman, with one child (3 years old), 12 years of education, 7 years of experience, and a husband with an income of 22. A 90% equal tail credible interval (a, b) cuts off 5% percent of the posterior probability mass to the left of a, and 5% to the right of b.

```r
#New data
#Constant,  HusbandIncome, EducYears, ExpYears, Age, NSmallCHild, NBigChild
x = c(1, 22, 12, 7, 38, 1, 0)
Pr_y1 = matrix(nrow=nDraws, ncol=1)
for(draw in 1:nDraws){
```

```
  Pr_y1[draw] = exp(t(x)%*%betas[draw,])/(1+exp(t(x)%*%betas[draw,]))
}

hist(Pr_y1, breaks=50, main="Posterior distribution of Pr(y=1|x)", xlab="Pr(y=1|x)")
abline(v=quantile(Pr_y1, c(0.05, 0.95)), col="red")
abline(v=mean(Pr_y1), col="blue")
legend("topright", legend=c("Mean", "90% Credible Interval"), col=c("blue", "red"), lty=1:1)
```




```
print(quantile(Pr_y1, c(0.05, 0.95)))
```

```
##        5%       95%
## 0.1100806 0.4029005
```

```
print(mean(Pr_y1))
```

```
## [1] 0.2419068
```

2. Metropolis Random Walk for Poisson regression

Consider the following Poisson regression model

$$y_i|\beta \sim^{iid} Poisson[exp(x_i^T\beta)], i = 1, ..., n,$$

where y_i is the count for the ith observation in the sample and x_i is the p-dimensional vector with covariate observations for the ith observation. Use the data set eBayNumberOfBidderData_2024.dat. This dataset contains observations from 800 eBay auctions of coins. The response variable is **nBids** and records the number of bids in each auction. The remaining variables are features/covariates (**x**):

- **Const** (for the intercept)
- **PowerSeller** (equal to 1 if the seller is selling large volumes on eBay)
- **VerifyID** (equal to 1 if the seller is a veriffed seller by eBay)
- **Sealed** (equal to 1 if the coin was sold in an unopened envelope)
- **MinBlem** (equal to 1 if the coin has a minor defect)
- **MajBlem** (equal to 1 if the coin has a major defect)
- **LargNeg** (equal to 1 if the seller received a lot of negative feedback from customers)
- **LogBook** (logarithm of the book value of the auctioned coin according to expert sellers. Standardized)
- **MinBidShare** (ratio of the minimum selling price (starting price) to the book value. Standardized).

a)

Obtain the maximum likelihood estimator of β in the Poisson regression model for the eBay data [Hint: glm.R, don't forget that glm() adds its own intercept so don't input the covariate Const]. Which covariates are signifficant?

```
ebayData <- read.table("eBayNumberOfBidderData_2024.dat", header = T)
y <- ebayData$nBids
X <- as.matrix(ebayData[,-1])
Xnames <- colnames(X)

glmFit <- glm(y ~ X - 1, family = poisson)
summary(glmFit)
```

```
##
## Call:
## glm(formula = y ~ X - 1, family = poisson)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## XConst        1.07981    0.03393  31.828  < 2e-16 ***
## XPowerSeller -0.03566    0.04167  -0.856 0.392109
## XVerifyID    -0.45564    0.12748  -3.574 0.000351 ***
## XSealed       0.45515    0.06226   7.311 2.65e-13 ***
## XMinblem     -0.06837    0.07198  -0.950 0.342228
## XMajBlem     -0.22554    0.09525  -2.368 0.017894 *
## XLargNeg      0.05382    0.06406   0.840 0.400787
## XLogBook     -0.08499    0.03234  -2.628 0.008599 **
## XMinBidShare -1.82490    0.07843 -23.269  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 4833.6  on 800  degrees of freedom
## Residual deviance:  691.8  on 791  degrees of freedom
## AIC: 2879.1
```

```
##
## Number of Fisher Scoring iterations: 5
```

The covariates XConst, XVerifyID, XSealed, XMajBlem, XLogBook and XMinBidShare are significant at the 95% level.

b)

Let's do a Bayesian analysis of the Poisson regression. Let the prior be $\beta \sim N[0, 100 * (X^T X)^{-1}$, where X is the n * p covariate matrix. This is a commonly used prior, which is called Zellner's g-prior. Assume first that the posterior density is approximately multivariate normal:

$$\beta|y \sim N[\tilde{\beta}, J_y^{-1}(\tilde{\beta})]$$

, where $\tilde{\beta}$ is the posterior mode and $J_y(\tilde{\beta})$ is the negative Hessian at the posterior mode. $\tilde{\beta}$ and $J_y(\tilde{\beta})$ can be obtained by numerical optimization (optim.R) exactly like you already did for the logistic regression in Lab 2 (but with the log posterior function replaced by the corresponding one for the Poisson model, which you have to code up.).

```
# Prior
n = nrow(X)
p = ncol(X)

mu <- as.matrix(rep(0,p))
Sigma <- as.matrix(100 * solve(t(X) %*% X) )
betas = as.matrix(rep(0,p)) #Initialized to zero

posterior_function <- function(betas, y, X,  mu, Sigma){
  #Note to self: parameter to be optim is the first argument
  linearPredictor <- X %*% betas;
  logLik <- sum( linearPredictor*y - exp(linearPredictor) )
  logPrior <- dmvnorm(betas, mu, Sigma, log=TRUE);
  return(logLik + logPrior)
}

OptimRes <- optim(par = betas, fn = posterior_function, gr=NULL, y, X, mu, Sigma,
                  method=c("BFGS"),control=list(fnscale=-1),hessian=TRUE)

# Printing the results to the screen
names(OptimRes$par) <- Xnames # Naming the coefficient by covariates
approxPostStd <- sqrt(diag(solve(-OptimRes$hessian))) # Computing approximate standard deviations.
names(approxPostStd) <- Xnames # Naming the coefficient by covariates
cat('---- The posterior mode is: ----\n')
```

```
## ---- The posterior mode is: ----
```

```
print(OptimRes$par[1:9])
```

```
##       Const PowerSeller    VerifyID      Sealed     Minblem     MajBlem
##  1.07721720 -0.03567963 -0.45353183  0.45484863 -0.06863401 -0.22583912
##     LargNeg     LogBook MinBidShare
##  0.05387677 -0.08454639 -1.82275698
```

```r
cat('\n---- The approximate posterior standard deviation is: ----\n')
```

```
##
## ---- The approximate posterior standard deviation is: ----
```

```r
print(approxPostStd)
```

```
##       Const PowerSeller    VerifyID      Sealed     Minblem     MajBlem
##  0.03389556  0.04167562  0.12715595  0.06227165  0.07198300  0.09527403
##     LargNeg     LogBook MinBidShare
##  0.06408047  0.03233568  0.07826924
```

```r
cat("\n---- comparing with GLM: ----\n")
```

```
##
## ---- comparing with GLM: ----
```

```r
coef(glmFit)
```

```
##       XConst XPowerSeller    XVerifyID      XSealed     XMinblem     XMajBlem
##   1.07980512  -0.03566493  -0.45563760   0.45515199  -0.06836819  -0.22554138
##     XLargNeg     XLogBook XMinBidShare
##   0.05382386  -0.08498844  -1.82490142
```

c)

Let's simulate from the actual posterior of β using the Metropolis algorithm and compare the results with the approximate results in b). Program a general function that uses the Metropolis algorithm to generate random draws from an arbitrary posterior density. In order to show that it is a general function for any model, we denote the vector of model parameters by θ. Let the proposal density be the multivariate normal density mentioned in Lecture 8 (random walk Metropolis):

$$\theta_p|\theta^{(i-1)} \sim N[\theta^{(i-1)}, c * \Sigma]$$

, where $\Sigma = J_y^{-1}(\tilde{\beta})$ was obtained in b). The value c is a tuning parameter and should be an input to your Metropolis function. The user of your Metropolis function should be able to supply her own posterior density function, not necessarily for the Poisson regression, and still be able to use your Metropolis function. This is not so straightforward, unless you have come across function objects in R. The note HowToCodeRWM.pdf in Lisam describes how you can do this in R.

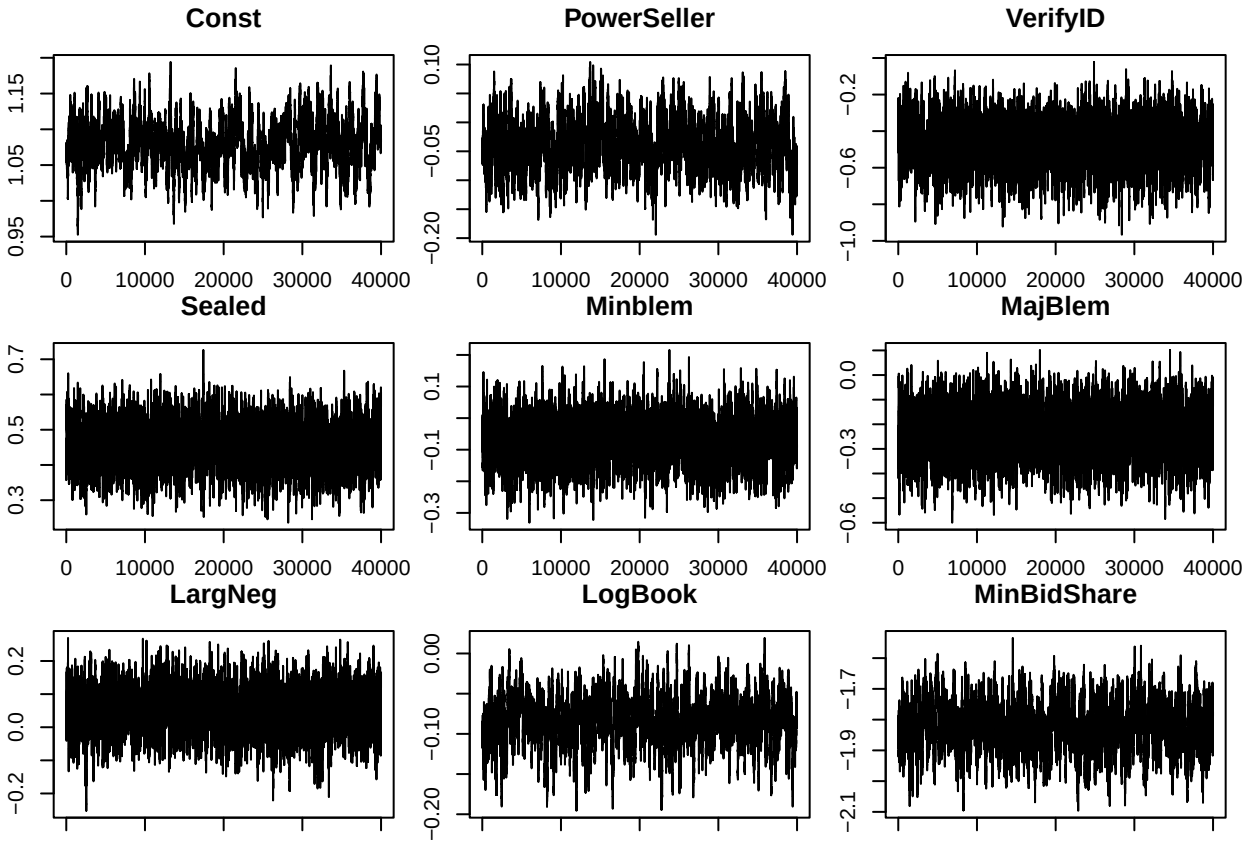
Now, use your new Metropolis function to sample from the posterior of β in the Poisson regression for the eBay dataset. Assess MCMC convergence by graphical methods.

```r
# Metropolis function
Metropolis <- function(n, theta0, posterior_function, c, y, X, mu, Sigma){
  # n: number of iterations
  # theta0: initial value of theta
  # posterior_function: function that returns the log posterior
  # c: tuning parameter
  # y: response variable
```

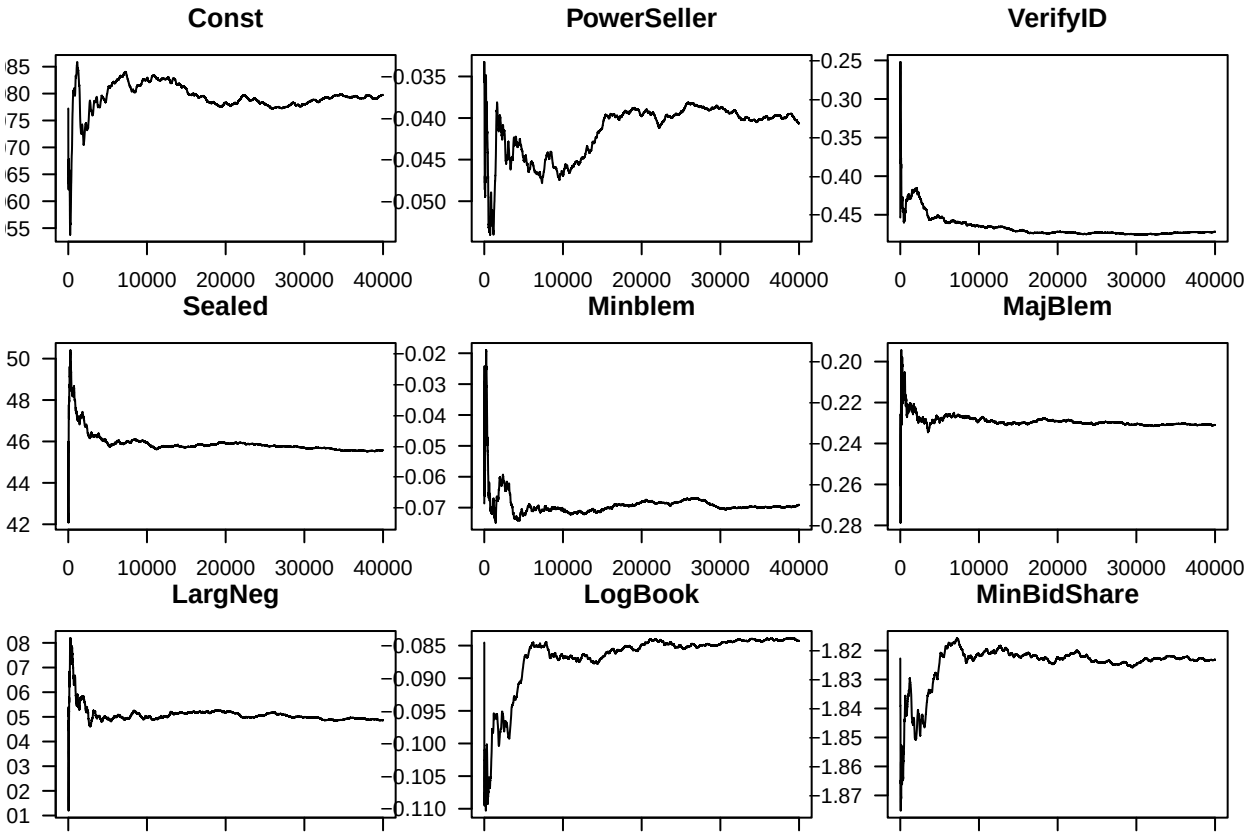
```r
  # X: covariate matrix
  # mu: prior mean
  # Sigma: prior covariance matrix

  p = length(theta0)
  theta = matrix(NA, nrow = n, ncol = p)
  theta[1,] = theta0
  accepted = 0
  for(i in 2:n){
    theta_prop = rnorm(p, theta[i-1,], diag(c*Sigma))
    logR = posterior_function(theta_prop, y, X, mu, Sigma) - posterior_function(theta[i-1,], y, X, mu, S
    if(log(runif(1)) < logR){
      theta[i,] = theta_prop
      accepted = accepted + 1
    } else {
      theta[i,] = theta[i-1,]
    }
  }
  print(paste("Acceptance rate: ", accepted/n))
  return(theta)
}

#Simulating from the posterior
n = 40000
c = 0.03
theta0 = OptimRes$par
posterior_function <- function(betas, y, X,  mu, Sigma){
  #Note to self: parameter to be optim is the first argument
  linearPredictor <- X %*% betas;
  logLik <- sum( linearPredictor*y - exp(linearPredictor) )
  logPrior <- dmvnorm(betas, mu, Sigma, log=TRUE);
  return(logLik + logPrior)
}

thetas = Metropolis(n, theta0, posterior_function, c, y, X, mu, Sigma)
```

```
## [1] "Acceptance rate:  0.29455"
```

```r
#Assessing convergence
par(mar=c(1,2,3,1))
par(mfrow=c(3,3))
for(i in 1:p){
  plot(thetas[,i], type = "l", main = Xnames[i])
}
```




```
cummean <- function(x){
  cumsum(x) / seq_along(x)
}

#Plot the cumulative mean
for (i in 1:p){
  plot(cummean(thetas[,i]), type="l", main = Xnames[i], las=1)
}
```



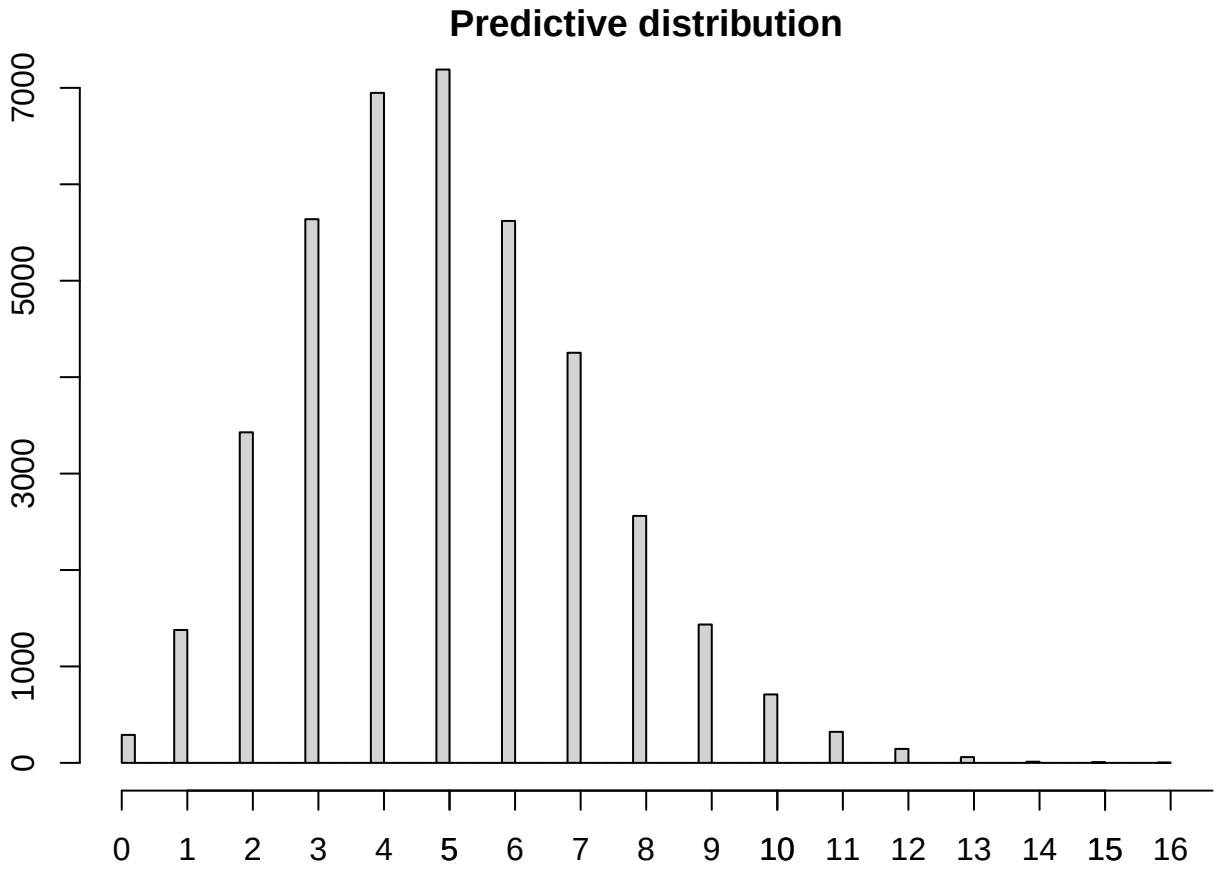
d)

Use the MCMC draws from c) to simulate from the predictive distribution of the number of bidders in a new auction with the characteristics below. Plot the predictive distribution. What is the probability of no bidders in this new auction?

- **PowerSeller** = 1
- **VerifyID** = 0
- **Sealed** = 1
- **MinBlem** = 0
- **MajBlem** = 1
- **LargNeg** = 0
- **LogBook** = 1.2
- **MinBidShare** = 0.8

```
# New auction
Xnew = as.matrix(c(1, 0, 1, 0, 1, 0, 1.2, 0.8))

pred = c()
# Predictive distribution
for (i in 1:n){
  lambda = c(pred, exp(Xnew %*% thetas[i,]))
  pred = append(pred, rpois(1, lambda))
}
```

```r
par(mfrow=c(1,1))
par(mar=c(2,2,1,1))
hist(pred, main = "Predictive distribution", xlab = "Number of bidders", ylab = "Frequency", breaks = "S
axis(1, at = c(1:length(table(pred))), labels = c(1:length(table(pred))))
```




```r
#find the unique values in pred
pred_dist = table(pred)
print(pred_dist)
```

```
## pred
##    0    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15
##  290 1378 3428 5638 6948 7190 5620 4253 2561 1435  710  322  145   60   12    7
##   16
##    3
```

```r
# Probability of no bidders
prob_no_bidders = pred_dist[1]/sum(pred_dist)
cat("The probability of no bidders in the new auction is: ", prob_no_bidders)
```

```
## The probability of no bidders in the new auction is:  0.00725
```

3. Time series models in Stan

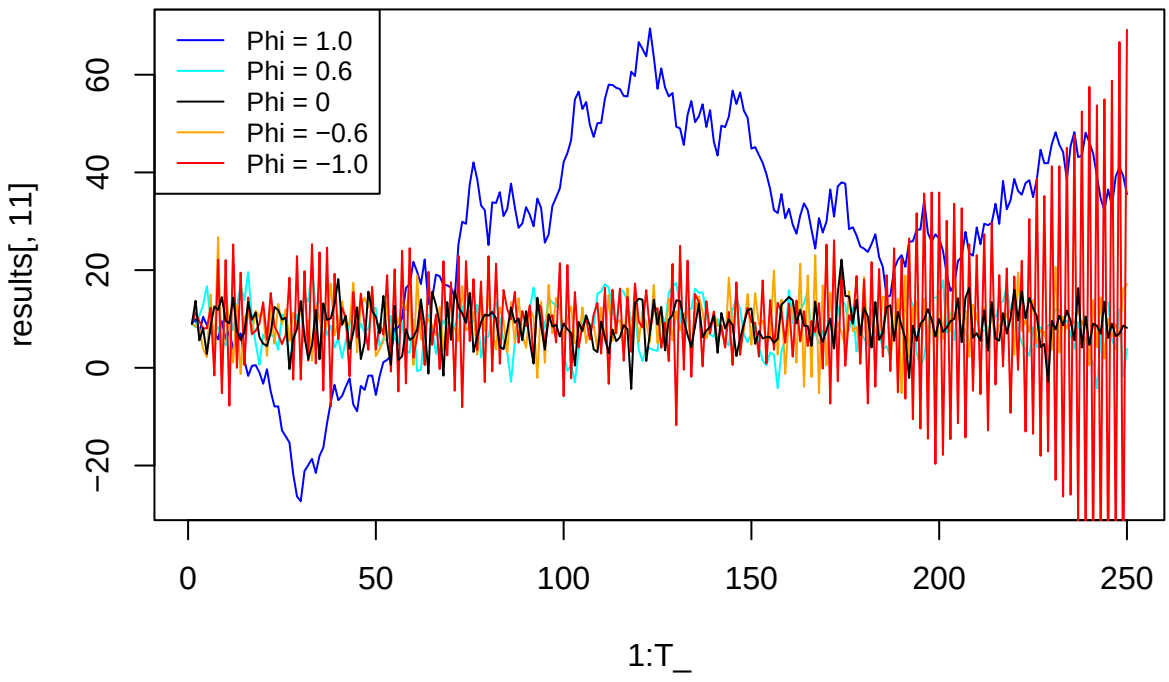
a)

Write a function in R that simulates data from the AR(1)-process

$$x_t = \mu + \phi(x_{t-1} - \mu) + \epsilon_t, \epsilon_t \sim N[0, \sigma^2]$$

, for given values of μ, ϕ and σ^2 . Start the process at $x1 = \mu$ and then simulate values for x_t for t = 2, 3. . . , T and return the vector $x_{1:T}$ containing all time points. Use $\mu = 9, \sigma^2 = 4$ and T = 250 and look at some different realizations (simulations) of $x_{1:T}$ for values of ϕ between -1 and 1 (this is the interval of ϕ where the AR(1)-process is stationary). Include a plot of at least one realization in the report. What effect does the value of ϕ have on $x_{1:T}$?

```
set.seed(12345)
mu = 9
sig_sq = 4
T_ = 250
phis = seq(-1,1,0.2) # -1.0, -0.8, ... 0 ... 0.8, 1.0
results = matrix(nrow = T_, ncol = length(phis))
results[1,] = mu #initialise the matrix to mu for t=1

for (phi in 1:length(phis)){
  for (t in 2:T_){
    #Equation : x_t = mu + phi*      (x_t-1 - mu)              + N(0,sig_sq)
    results[t, phi] = mu + phis[phi]*(results[t-1, phi] - mu) + rnorm(1, 0, sig_sq)
  }
}
#dim(results)

#print(phis[3])
plot(x = 1:T_, y = results[,11],type='l', col = "blue") #phi = 1.0
lines(x = 1:T_, y = results[,9], col = 'cyan') #phi = 0.6

lines(x = 1:T_, y = results[,3], col='orange') #phi = -0.6
lines(x = 1:T_, y = results[,1], col='red') #phi = -1.0

lines(x = 1:T_, y = results[,6], col = 'black') #phi = 0
legend("topleft", legend=c("Phi = 1.0", "Phi = 0.6","Phi = 0","Phi = -0.6","Phi = -1.0"),
       col=c("blue","cyan","black", "orange", "red"),lty=c(1,1,1,1,1), cex=c(0.8,0.8,0.8,0.8,0.8))
```



Phi values closer to 0 seems to concentrate the evolution of $x_{1:T}$ around the starting value of mu = 9. Values of +/- 1.0 does not leads to a convergence.

b)

Use your function from a) to simulate two AR(1)-processes, $x_{1:T}$ with $\phi = 0.3$ and $y_{1:T}$ with $\phi = 0.97$. Now, treat your simulated vectors as synthetic data, and treat the values of μ, ϕ and σ^2 as unknown parameters. Implement Stan-code that samples from the posterior of the three parameters, using suitable non-informative priors of your choice. [Hint: Look at the time-series models examples in the Stan user's guide/reference manual, and note the different parameterization used here.]

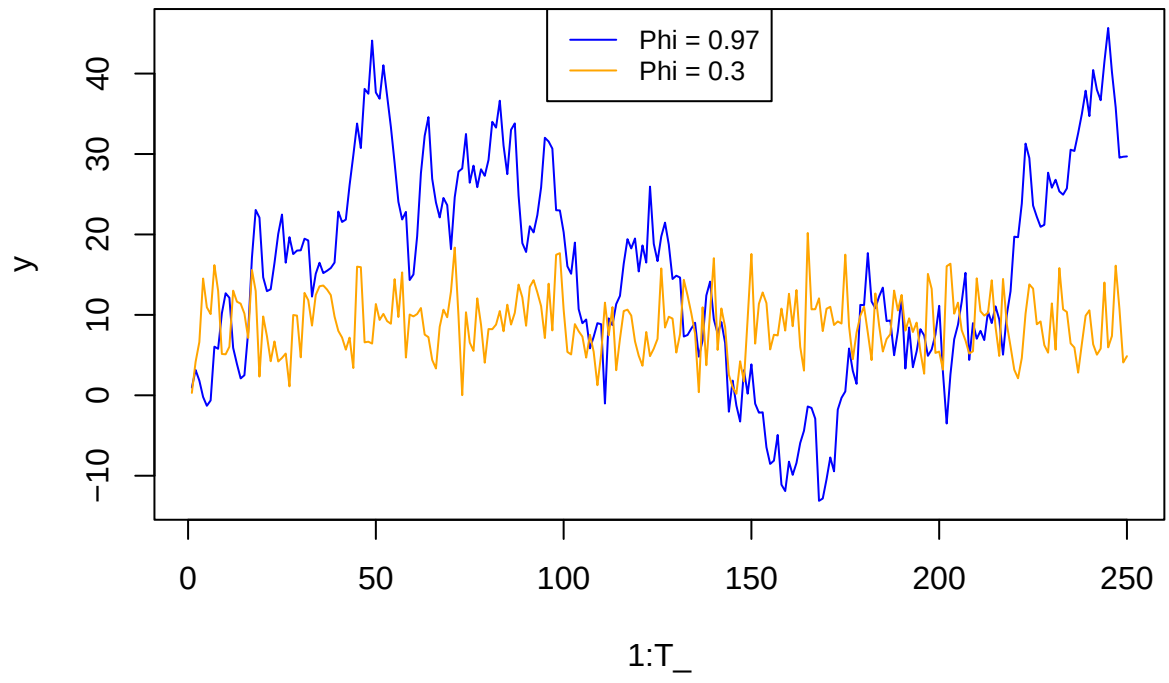
```
library(rstan)
```

```
## Loading required package: StanHeaders
```

```
##
## rstan version 2.32.5 (Stan version 2.32.2)
```

```
## For execution on a local, multicore CPU with excess RAM we recommend calling
## options(mc.cores = parallel::detectCores()).
## To avoid recompilation of unchanged Stan programs, we recommend calling
## rstan_options(auto_write = TRUE)
## For within-chain threading using ‘reduce_sum()‘ or ‘map_rect()‘ Stan functions,
## change ‘threads_per_chain‘ option:
## rstan_options(threads_per_chain = 1)
```

```r
set.seed(123)
x = rep(0, 250)
mu = 9
phi = 0.3
x[1] = phi
for (t in 2:T_){
  #Equation : x_t = mu + phi*(x_t-1 - mu)+ N(0,sig_sq)
  x[t] = mu + phi*(x[t-1] - mu) + rnorm(1, 0, sig_sq)
}

y = rep(0, 250)
phi = 0.97
y[1] = phi
for (t in 2:T_){
  #Equation : x_t = mu + phi*(x_t-1 - mu)+ N(0,sig_sq)
  y[t] = mu + phi*(y[t-1] - mu) + rnorm(1, 0, sig_sq)
}

plot(x = 1:T_, y = y ,type='l', col = "blue") #phi = 0.97
lines(x = 1:T_, y = x , col = 'orange') #phi = 0.3
legend("top", legend=c("Phi = 0.97", "Phi = 0.3"), col=c("blue","orange"), lty=c(1,1), cex=c(0.8,0.8))
```




```r
StanModel = '
data {
  int<lower=0> T;
```

```
  real x[T];
}
parameters {
  real mu;
  real phi;
  real<lower=0> sigma;
}
model {
  mu ~ normal(9, 30); //very flat bell curve, with mean 9, and large variance
  phi ~ uniform(-1, 1);
  sigma ~ scaled_inv_chi_square(1,2); // Scaled-inv-chi2 with nu 1,sigma 2
  for (t in 2:T){
    x[t] ~ normal(mu + phi*(x[t-1] - mu), sigma);
  }
}'

#phi = 0.3
data = list(T = T_, x = x)
fit = stan(model_code = StanModel, data = data, warmup = 1000, iter = 3000, chains = 4)
```

```
## Trying to compile a simple C file

## Running /usr/lib/R/bin/R CMD SHLIB foo.c
## using C compiler: 'gcc (Ubuntu 11.4.0-1ubuntu1~22.04) 11.4.0'
## gcc -I"/usr/share/R/include" -DNDEBUG   -I"/usr/lib/R/site-library/Rcpp/include/"  -I"/home/h/R/x86_
## In file included from /home/h/R/x86_64-pc-linux-gnu-library/4.4/RcppEigen/include/Eigen/Core:19,
##                  from /home/h/R/x86_64-pc-linux-gnu-library/4.4/RcppEigen/include/Eigen/Dense:1,
##                  from /usr/lib/R/site-library/StanHeaders/include/stan/math/prim/fun/Eigen.hpp:22,
##                  from <command-line>:
## /home/h/R/x86_64-pc-linux-gnu-library/4.4/RcppEigen/include/Eigen/src/Core/util/Macros.h:679:10: fata
##   679 | #include <cmath>
##       |          ^~~~~~
## compilation terminated.
## make: *** [/usr/lib/R/etc/Makeconf:195: foo.o] Error 1
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.000151 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 1.51 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 3000 [  0%]  (Warmup)
## Chain 1: Iteration:  300 / 3000 [ 10%]  (Warmup)
## Chain 1: Iteration:  600 / 3000 [ 20%]  (Warmup)
## Chain 1: Iteration:  900 / 3000 [ 30%]  (Warmup)
## Chain 1: Iteration: 1001 / 3000 [ 33%]  (Sampling)
## Chain 1: Iteration: 1300 / 3000 [ 43%]  (Sampling)
## Chain 1: Iteration: 1600 / 3000 [ 53%]  (Sampling)
## Chain 1: Iteration: 1900 / 3000 [ 63%]  (Sampling)
## Chain 1: Iteration: 2200 / 3000 [ 73%]  (Sampling)
## Chain 1: Iteration: 2500 / 3000 [ 83%]  (Sampling)
## Chain 1: Iteration: 2800 / 3000 [ 93%]  (Sampling)
```

```
## Chain 1: Iteration: 3000 / 3000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.526 seconds (Warm-up)
## Chain 1:                1.097 seconds (Sampling)
## Chain 1:                1.623 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2: Rejecting initial value:
## Chain 2:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 2:   Stan can't start sampling from this initial value.
## Chain 2:
## Chain 2: Gradient evaluation took 9.1e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.91 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 3000 [  0%]  (Warmup)
## Chain 2: Iteration:  300 / 3000 [ 10%]  (Warmup)
## Chain 2: Iteration:  600 / 3000 [ 20%]  (Warmup)
## Chain 2: Iteration:  900 / 3000 [ 30%]  (Warmup)
## Chain 2: Iteration: 1001 / 3000 [ 33%]  (Sampling)
## Chain 2: Iteration: 1300 / 3000 [ 43%]  (Sampling)
## Chain 2: Iteration: 1600 / 3000 [ 53%]  (Sampling)
## Chain 2: Iteration: 1900 / 3000 [ 63%]  (Sampling)
## Chain 2: Iteration: 2200 / 3000 [ 73%]  (Sampling)
## Chain 2: Iteration: 2500 / 3000 [ 83%]  (Sampling)
## Chain 2: Iteration: 2800 / 3000 [ 93%]  (Sampling)
## Chain 2: Iteration: 3000 / 3000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.569 seconds (Warm-up)
## Chain 2:                1.112 seconds (Sampling)
## Chain 2:                1.681 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 9.3e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.93 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 3000 [  0%]  (Warmup)
## Chain 3: Iteration:  300 / 3000 [ 10%]  (Warmup)
## Chain 3: Iteration:  600 / 3000 [ 20%]  (Warmup)
## Chain 3: Iteration:  900 / 3000 [ 30%]  (Warmup)
## Chain 3: Iteration: 1001 / 3000 [ 33%]  (Sampling)
## Chain 3: Iteration: 1300 / 3000 [ 43%]  (Sampling)
## Chain 3: Iteration: 1600 / 3000 [ 53%]  (Sampling)
## Chain 3: Iteration: 1900 / 3000 [ 63%]  (Sampling)
## Chain 3: Iteration: 2200 / 3000 [ 73%]  (Sampling)
## Chain 3: Iteration: 2500 / 3000 [ 83%]  (Sampling)
## Chain 3: Iteration: 2800 / 3000 [ 93%]  (Sampling)
## Chain 3: Iteration: 3000 / 3000 [100%]  (Sampling)
```

```
## Chain 3:
## Chain 3:  Elapsed Time: 0.536 seconds (Warm-up)
## Chain 3:                0.899 seconds (Sampling)
## Chain 3:                1.435 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 9.5e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.95 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 3000 [  0%]  (Warmup)
## Chain 4: Iteration:  300 / 3000 [ 10%]  (Warmup)
## Chain 4: Iteration:  600 / 3000 [ 20%]  (Warmup)
## Chain 4: Iteration:  900 / 3000 [ 30%]  (Warmup)
## Chain 4: Iteration: 1001 / 3000 [ 33%]  (Sampling)
## Chain 4: Iteration: 1300 / 3000 [ 43%]  (Sampling)
## Chain 4: Iteration: 1600 / 3000 [ 53%]  (Sampling)
## Chain 4: Iteration: 1900 / 3000 [ 63%]  (Sampling)
## Chain 4: Iteration: 2200 / 3000 [ 73%]  (Sampling)
## Chain 4: Iteration: 2500 / 3000 [ 83%]  (Sampling)
## Chain 4: Iteration: 2800 / 3000 [ 93%]  (Sampling)
## Chain 4: Iteration: 3000 / 3000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.548 seconds (Warm-up)
## Chain 4:                1.04 seconds (Sampling)
## Chain 4:                1.588 seconds (Total)
## Chain 4:
```

```
# Print the fitted model
print(fit,digits_summary=3)
```

```
## Inference for Stan model: anon_model.
## 4 chains, each with iter=3000; warmup=1000; thin=1;
## post-warmup draws per chain=2000, total post-warmup draws=8000.
##
##           mean se_mean    sd     2.5%      25%      50%      75%    97.5% n_eff
## mu       8.940   0.004 0.314    8.327    8.730    8.937    9.148    9.563  7331
## phi      0.229   0.001 0.062    0.108    0.188    0.229    0.270    0.353  7375
## sigma    3.783   0.002 0.170    3.468    3.664    3.777    3.894    4.136  7962
## lp__  -456.643   0.019 1.223 -459.832 -457.214 -456.317 -455.740 -455.249  4258
##        Rhat
## mu    1.001
## phi   1.000
## sigma 1.000
## lp__  1.000
##
## Samples were drawn using NUTS(diag_e) at Mon May 27 16:43:20 2024.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```r
# Extract posterior samples
postDraws <- extract(fit)

#phi = 0.97
data2 = list(T = T_, x = y)
fit2 = stan(model_code = StanModel, data = data2, warmup = 1000, iter = 3000, chains = 4)
```

```
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 6.9e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.69 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 3000 [  0%]  (Warmup)
## Chain 1: Iteration:  300 / 3000 [ 10%]  (Warmup)
## Chain 1: Iteration:  600 / 3000 [ 20%]  (Warmup)
## Chain 1: Iteration:  900 / 3000 [ 30%]  (Warmup)
## Chain 1: Iteration: 1001 / 3000 [ 33%]  (Sampling)
## Chain 1: Iteration: 1300 / 3000 [ 43%]  (Sampling)
## Chain 1: Iteration: 1600 / 3000 [ 53%]  (Sampling)
## Chain 1: Iteration: 1900 / 3000 [ 63%]  (Sampling)
## Chain 1: Iteration: 2200 / 3000 [ 73%]  (Sampling)
## Chain 1: Iteration: 2500 / 3000 [ 83%]  (Sampling)
## Chain 1: Iteration: 2800 / 3000 [ 93%]  (Sampling)
## Chain 1: Iteration: 3000 / 3000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 1.124 seconds (Warm-up)
## Chain 1:                0.895 seconds (Sampling)
## Chain 1:                2.019 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 6.2e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.62 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 3000 [  0%]  (Warmup)
## Chain 2: Iteration:  300 / 3000 [ 10%]  (Warmup)
## Chain 2: Iteration:  600 / 3000 [ 20%]  (Warmup)
## Chain 2: Iteration:  900 / 3000 [ 30%]  (Warmup)
## Chain 2: Iteration: 1001 / 3000 [ 33%]  (Sampling)
## Chain 2: Iteration: 1300 / 3000 [ 43%]  (Sampling)
## Chain 2: Iteration: 1600 / 3000 [ 53%]  (Sampling)
## Chain 2: Iteration: 1900 / 3000 [ 63%]  (Sampling)
## Chain 2: Iteration: 2200 / 3000 [ 73%]  (Sampling)
## Chain 2: Iteration: 2500 / 3000 [ 83%]  (Sampling)
## Chain 2: Iteration: 2800 / 3000 [ 93%]  (Sampling)
## Chain 2: Iteration: 3000 / 3000 [100%]  (Sampling)
## Chain 2:
```

```
## Chain 2:  Elapsed Time: 1.32 seconds (Warm-up)
## Chain 2:                1.18 seconds (Sampling)
## Chain 2:                2.5 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3: Rejecting initial value:
## Chain 3:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 3:   Stan can't start sampling from this initial value.
## Chain 3:
## Chain 3: Gradient evaluation took 6.6e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.66 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 3000 [  0%]  (Warmup)
## Chain 3: Iteration:  300 / 3000 [ 10%]  (Warmup)
## Chain 3: Iteration:  600 / 3000 [ 20%]  (Warmup)
## Chain 3: Iteration:  900 / 3000 [ 30%]  (Warmup)
## Chain 3: Iteration: 1001 / 3000 [ 33%]  (Sampling)
## Chain 3: Iteration: 1300 / 3000 [ 43%]  (Sampling)
## Chain 3: Iteration: 1600 / 3000 [ 53%]  (Sampling)
## Chain 3: Iteration: 1900 / 3000 [ 63%]  (Sampling)
## Chain 3: Iteration: 2200 / 3000 [ 73%]  (Sampling)
## Chain 3: Iteration: 2500 / 3000 [ 83%]  (Sampling)
## Chain 3: Iteration: 2800 / 3000 [ 93%]  (Sampling)
## Chain 3: Iteration: 3000 / 3000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 1.82 seconds (Warm-up)
## Chain 3:                0.812 seconds (Sampling)
## Chain 3:                2.632 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
## Chain 4: Rejecting initial value:
## Chain 4:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 4:   Stan can't start sampling from this initial value.
## Chain 4:
## Chain 4: Gradient evaluation took 8.9e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.89 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 3000 [  0%]  (Warmup)
## Chain 4: Iteration:  300 / 3000 [ 10%]  (Warmup)
## Chain 4: Iteration:  600 / 3000 [ 20%]  (Warmup)
## Chain 4: Iteration:  900 / 3000 [ 30%]  (Warmup)
## Chain 4: Iteration: 1001 / 3000 [ 33%]  (Sampling)
## Chain 4: Iteration: 1300 / 3000 [ 43%]  (Sampling)
## Chain 4: Iteration: 1600 / 3000 [ 53%]  (Sampling)
## Chain 4: Iteration: 1900 / 3000 [ 63%]  (Sampling)
## Chain 4: Iteration: 2200 / 3000 [ 73%]  (Sampling)
## Chain 4: Iteration: 2500 / 3000 [ 83%]  (Sampling)
## Chain 4: Iteration: 2800 / 3000 [ 93%]  (Sampling)
```

```
## Chain 4: Iteration: 3000 / 3000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 1.323 seconds (Warm-up)
## Chain 4:                0.722 seconds (Sampling)
## Chain 4:                2.045 seconds (Total)
## Chain 4:
```

```
## Warning: There were 689 divergent transitions after warmup. See
## https://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
## to find out why this is a problem and how to eliminate them.
```

```
## Warning: Examine the pairs() plot to diagnose sampling problems
```

```
# Print the fitted model
print(fit2,digits_summary=3)
```

```
## Inference for Stan model: anon_model.
## 4 chains, each with iter=3000; warmup=1000; thin=1;
## post-warmup draws per chain=2000, total post-warmup draws=8000.
##
##           mean se_mean    sd     2.5%      25%      50%      75%    97.5% n_eff
## mu      17.999   0.335 9.443    0.627   13.356   17.355   21.888   38.829   796
## phi      0.959   0.000 0.020    0.917    0.945    0.959    0.973    0.995  1887
## sigma    4.028   0.003 0.183    3.694    3.901    4.019    4.147    4.406  4116
## lp__  -472.589   0.033 1.239 -475.621 -473.261 -472.303 -471.646 -471.086  1397
##        Rhat
## mu    1.002
## phi   1.001
## sigma 1.000
## lp__  1.002
##
## Samples were drawn using NUTS(diag_e) at Mon May 27 16:43:30 2024.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```
# Extract posterior samples
postDraws2 <- extract(fit2)
```

i)

Report the posterior mean, 95% credible intervals and the number of effective posterior samples for the three inferred parameters for each of the simulated AR(1)-process. Are you able to estimate the true values?

```
#For phi = 0.3
mu_mean = round(summary(fit)$summary[1,1],3)
mu_95ci = round(summary(fit)$summary[1,c(4,8)],3)
mu_eff = round(summary(fit)$summary[1,9])

phi_mean = round(summary(fit)$summary[2,1],3)
phi_95ci = round(summary(fit)$summary[2,c(4,8)],3)
phi_eff = round(summary(fit)$summary[2,9])
```

```
sigma_mean = round(summary(fit)$summary[3,1],3)
sigma_95ci = round(summary(fit)$summary[3,c(4,8)],3)
sigma_eff = round(summary(fit)$summary[3,9])

#For phi = 0.97
mu_mean2 = round(summary(fit2)$summary[1,1],3)
mu_95ci2 = round(summary(fit2)$summary[1,c(4,8)],3)
mu_eff2 = round(summary(fit2)$summary[1,9])

phi_mean2 = round(summary(fit2)$summary[2,1],3)
phi_95ci2 = round(summary(fit2)$summary[2,c(4,8)],3)
phi_eff2 = round(summary(fit2)$summary[2,9])

sigma_mean2 = round(summary(fit2)$summary[3,1],3)
sigma_95ci2 = round(summary(fit2)$summary[3,c(4,8)],3)
sigma_eff2 = round(summary(fit2)$summary[3,9])
```

Using phi = 0.3:

- For μ, the mean is 8.94, with 95% credible interval of [8.327 , 9.563]. The number of effective posterior samples is 7331.
- For ϕ, the mean is 0.229, with 95% credible interval of [0.108 , 0.353]. The number of effective posterior samples is 7375.
- For σ, the mean is 3.783, with 95% credible interval of [3.468 , 4.136]. The number of effective posterior samples is 7962.

Using phi = 0.97:

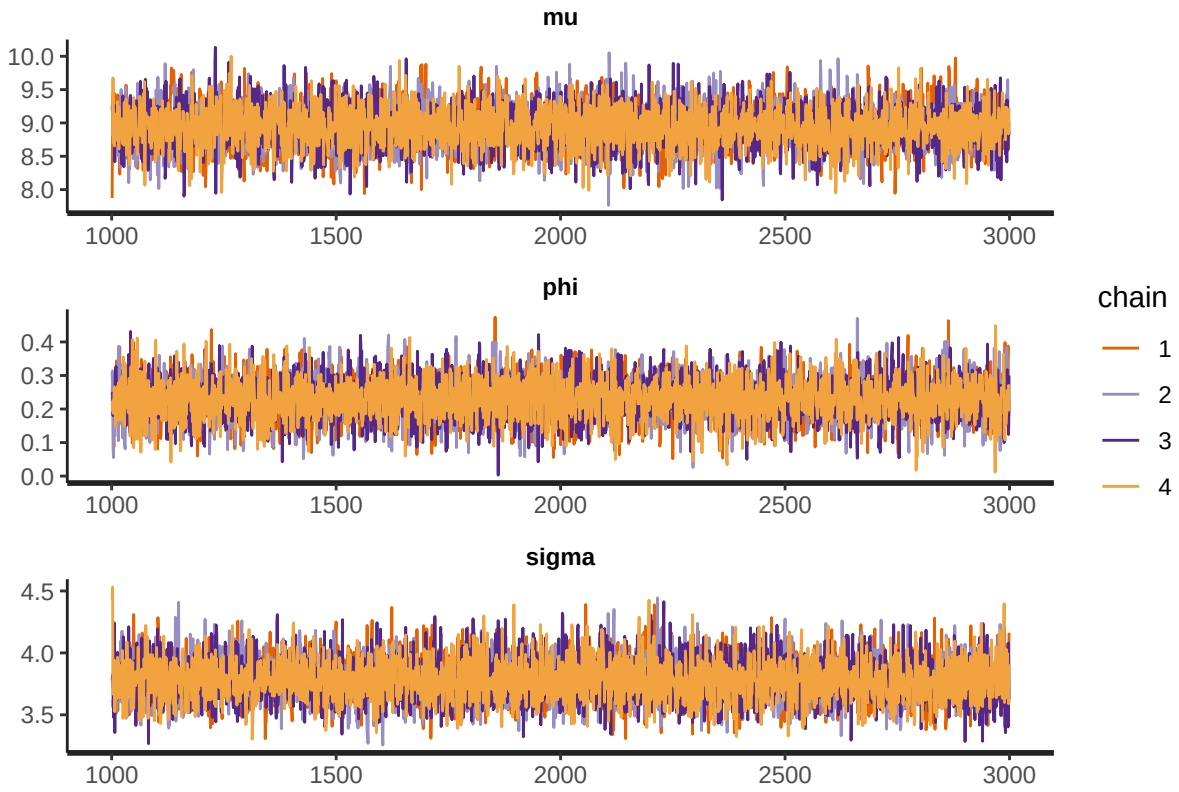
- For μ, the mean is 17.999, with 95% credible interval of [0.627 , 38.829]. The number of effective posterior samples is 796.
- For ϕ, the mean is 0.959, with 95% credible interval of [0.917 , 0.995]. The number of effective posterior samples is 1887.
- For σ, the mean is 4.028, with 95% credible interval of [3.694 , 4.406]. The number of effective posterior samples is 4116.

ii)

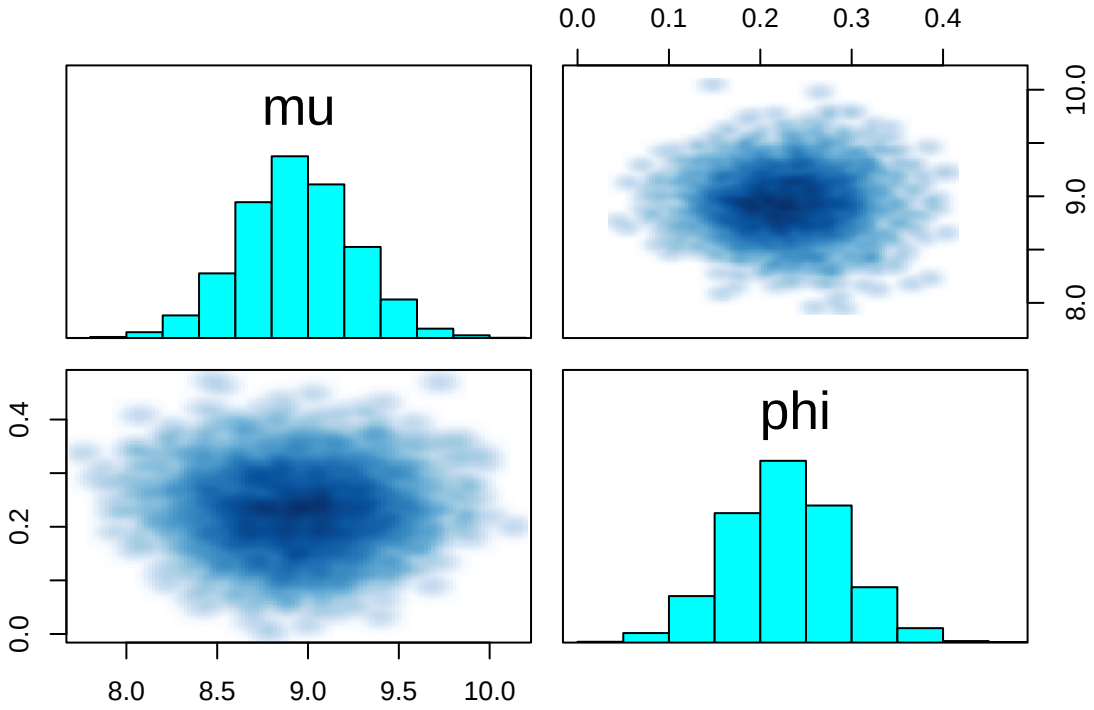
For each of the two data sets, evaluate the convergence of the samplers and plot the joint posterior of μ and ϕ. Comments?

For $\phi = 0.3$:

```
par(mfrow = c(1,3))
#phi = 0.3
traceplot(fit, nrow = 3)
```

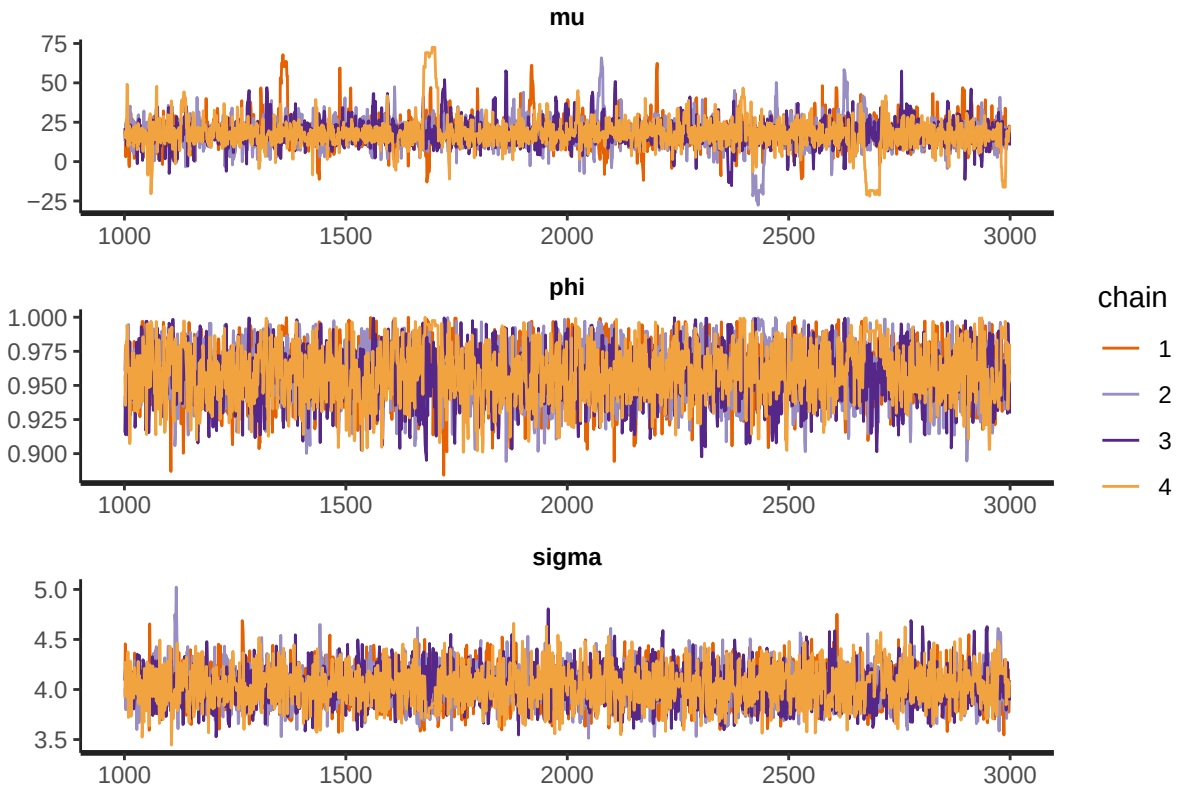



```
#joint posterior of mu and phi
par(mfrow = c(1,1))
pairs(fit, pars = c("mu", "phi"))
```

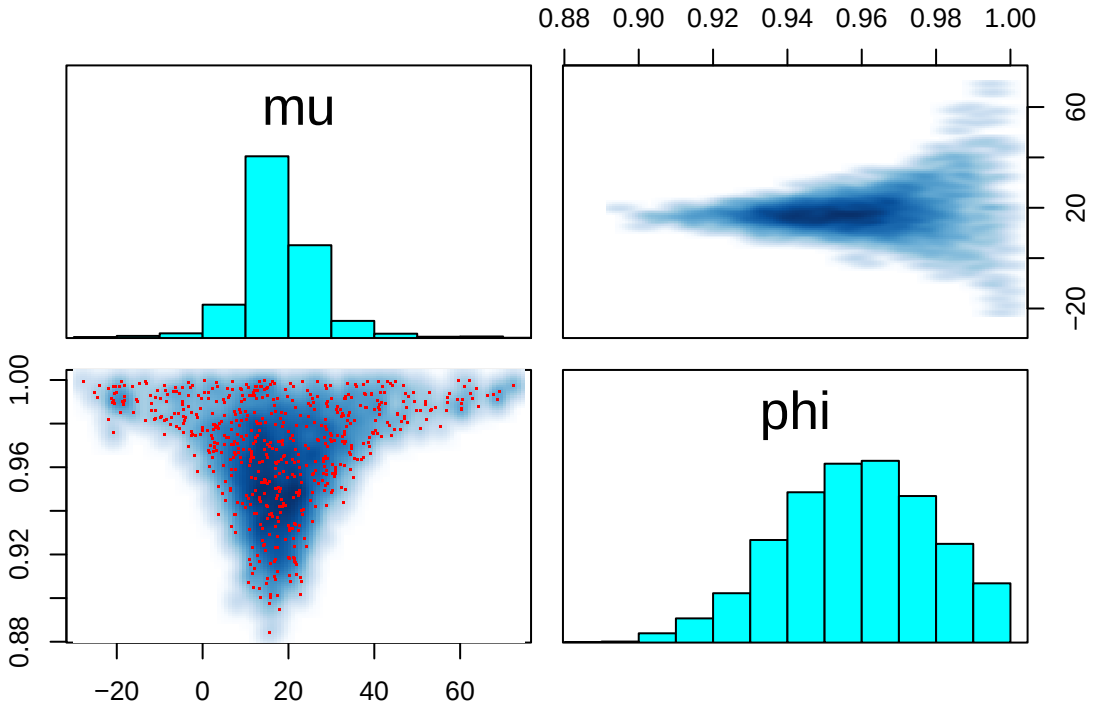


For $\phi = 0.97$:

```
par(mfrow = c(1,3))
#phi = 0.97
traceplot(fit2, nrow = 3)
```




```
#joint posterior of mu and phi
par(mfrow = c(1,1))
pairs(fit2, pars = c("mu", "phi"))
```



For both datasets, the samplers seem to have converged, although the tracesplot for phi=0.97 contains much more variation.

For $\phi = 0.3$, the posterior distribution of μ is centered around 9, which is the true value. The posterior distribution of ϕ is centered around 0.25, which is close to 0.3, which is the true value.

For $\phi = 0.97$, the joint posterior of μ and ϕ shows a strong correlation between the two parameters. The posterior distribution of μ is centered around 19, which quite far of the actual value of 9. The posterior distribution of ϕ seems to be centered around 0.97, or close to it, which is the true value. The big difference in the posterior distribution of μ is due to the fact that the data is generated from a process with a high ϕ value, which makes the process to be more dependent on the previous value of the process. It can be seen in the traceplot of the synthetic data that the blue curve (phi=0.97) is more often "above" the mean than "below", which gives an intuition as why the posterior mean of μ is higher than the true value. In the traceplot of of the chains for mu that there are some chains with high outliers, which may be the reason for the high posterior mean of μ. The number of effective posterior samples is also lower than for the other dataset.