Worksheet 4

Foundations of Bayesian Methodology

Wenje Tu   Lea Bührer   Jerome Sepin   Zhixuan Li

Elia-Leonid Mastropietro   Jonas Raphael Füglistaler   Nico Schümperli

Spring Semester 2022

Exercise 2 (Gibbs sampler)

2(a)

$$p(x) = \exp\left(-\frac{1}{2}(ax^2 - 2bx)\right)$$

$$= \exp\left(-\frac{a}{2}\left\{x^2 - 2\frac{b}{a}x\right\}\right)$$

$$= \exp\left(-\frac{a}{2}\left\{x^2 - 2\frac{b}{a}x + \left(\frac{b}{a}\right)^2 - \left(\frac{b}{a}\right)^2\right\}\right)$$

$$= \exp\left(-\frac{a}{2}\left(x - \frac{b}{a}\right)^2 + \frac{b^2}{2a}\right)$$

$$= \underbrace{\exp\left(\frac{b^2}{2a}\right)}_{\text{costant}} \exp\left(-\frac{a}{2}\left(x - \frac{b}{a}\right)^2\right)$$

Distribution function of $p(x)$:

$$f(x) = \frac{p(x)}{\int_{-\infty}^{\infty} p(x)dx}$$

$$= \frac{\exp\left(\frac{b^2}{2a}\right)\exp\left(-\frac{a}{2}\left(x - \frac{b}{a}\right)^2\right)}{\int_{-\infty}^{\infty} \exp\left(\frac{b^2}{2a}\right)\exp\left(-\frac{a}{2}\left(x - \frac{b}{a}\right)^2\right)dx}$$

$$= \frac{\exp\left(-\frac{a}{2}\left(x - \frac{b}{a}\right)^2\right)}{\int_{-\infty}^{\infty} \exp\left(-\frac{a}{2}\left(x - \frac{b}{a}\right)^2\right)dx}$$

$$= \frac{\exp\left(-\frac{a}{2}\left(x - \frac{b}{a}\right)^2\right)}{\sqrt{\frac{2\pi}{a}}\underbrace{\int_{-\infty}^{\infty} \sqrt{\frac{a}{2\pi}}\exp\left(-\frac{a}{2}\left(x - \frac{b}{a}\right)^2\right)dx}_{\text{integrates to 1}}}$$

$$= \sqrt{\frac{a}{2\pi}}\exp\left(-\frac{a}{2}\left(x - \frac{b}{a}\right)^2\right)$$

$$X \sim \mathrm{N}\left(\frac{b}{a}, \frac{1}{a}\right)$$

2(b)

Generate data: a random normal sample by assuming `set.seed(44566)`, $n = 30$, $\mu = 4$ and $\sigma^2 = 16$.

```r
# Generate data
set.seed(44566)
mu <- 4
sigma2 <- 16
n <- 30
y <- rnorm(n=n, mean=mu, sd=sqrt(sigma2))
```

```r
# Define the parameters of the prior distributions
mu0 <- -3
sigma2_0 <- 4
a0 <- 1.6
b0 <- 0.4
```

```r
#####################
## Gibbs sampler (1 chain)
#####################

# initialisation
set.seed(44566)

n.iter <- 10000
n.burnin <- 4000
n.thin <- 1
#n.thin <- floor((n.iter-n.burnin)/500)
n.chains <- 1
parameters <- c("mu", "sigma2", "inv_sigma2")
n.parameters <- length(parameters)

n.tot <- n.burnin + n.iter*n.thin

gibbs_samples <- matrix(NA, nrow = n.iter, ncol = n.parameters)
colnames(gibbs_samples) <- parameters

mu.sim <- rep(NA, length = n.tot)
sigma2.sim <- rep(NA, length = n.tot)
inv.sigma2.sim <- rep(NA, length = n.tot)

# Set the initial value
sigma2.sim[1] <- 1/runif(n.chains)

# set the counter
k <- 1

#Run the for loop (only one chain)
for(i in 2:(n.burnin+n.iter*n.thin)){

  mu.sim[i] <- rnorm(1,
                  mean = (sum(y)/sigma2.sim[i-1] + mu0/sigma2_0) /
                    (n/sigma2.sim[i-1] + 1/sigma2_0),
                  sd = sqrt(1/(n/sigma2.sim[i-1] + 1/sigma2_0)))

  sigma2.sim[i] <- 1/rgamma(1, shape = n/2 + a0,
                        scale = 1 / (sum((y-mu.sim[i])^2)/2 + b0))

  inv.sigma2.sim[i] <- 1/sigma2.sim[i]
```

```
  # after the burnin save every n.thin'th sample
  if((i > n.burnin) && (i%%n.thin == 0)){
    gibbs_samples[k,] <- c(mu.sim[i], sigma2.sim[i], inv.sigma2.sim[i])
    k <- k + 1
  }

  if(i%%1000 == 0){
    # report on the fly in which iteration the chain is
    cat(i, "\n")
  }

}
```

```
## 1000
## 2000
## 3000
## 4000
## 5000
## 6000
## 7000
## 8000
## 9000
## 10000
## 11000
## 12000
## 13000
## 14000
```

```
# n.iter samples after n.burnin taking every n.thin'th sample
dim(gibbs_samples)
```

```
## [1] 10000      3
```

```
mu_gibbs_samples <- gibbs_samples[,"mu"]
sigma2_gibbs_samples <- gibbs_samples[,"sigma2"]
inv_sigma2_gibbs_samples <- gibbs_samples[,"inv_sigma2"]
```

```
library(ggplot2)

d.gibbs <- data.frame(gibbs_samples)

## Traceplot of mu
ggplot(d.gibbs, aes(x=1:nrow(d.gibbs), y=mu)) +
  geom_line(color=2, alpha=0.5) +
  labs(title="Traceplot of mu", x="Iteration", y=expression(mu)) +
  theme_minimal()

## Histogram/Density of mu
ggplot(d.gibbs, aes(x=mu, y=..density..)) +
  geom_histogram(color=2, fill=2, alpha=0.5, bins=50) +
  geom_density(color=2, lwd=1) +
  labs(title="Histogram of mu", x=expression(mu)) +
  theme_minimal()

## Traceplot of sigma2
```

```
ggplot(d.gibbs, aes(x=1:nrow(d.gibbs), y=sigma2)) +
  geom_line(color=3, alpha=0.5) +
  labs(title="Traceplot of sigma2", x="Iteration", y=expression(sigma^2)) +
  theme_minimal()

## Histogram/Density of sigma2
ggplot(d.gibbs, aes(x=sigma2, y=..density..)) +
  geom_histogram(color=3, fill=3, alpha=0.5, bins=50) +
  geom_density(color=3, lwd=1) +
  labs(title="Histogram of sigma2", x=expression(sigma^2)) +
  theme_minimal()

## Traceplot of inv_sigma2
ggplot(d.gibbs, aes(x=1:nrow(d.gibbs), y=inv_sigma2)) +
  geom_line(color=4, alpha=0.5) +
  labs(title="Traceplot of inv_sigma2", x="Iteration", y=expression(1/sigma^2)) +
  theme_minimal()

## Histogram/Density of inv_sigma2
ggplot(d.gibbs, aes(x=inv_sigma2, y=..density..)) +
  geom_histogram(color=4, fill=4, alpha=0.5, bins=50) +
  geom_density(color=4, lwd=1) +
  labs(title="Histogram of inv_sigma2", x=expression(1/sigma^2)) +
  theme_minimal()
```
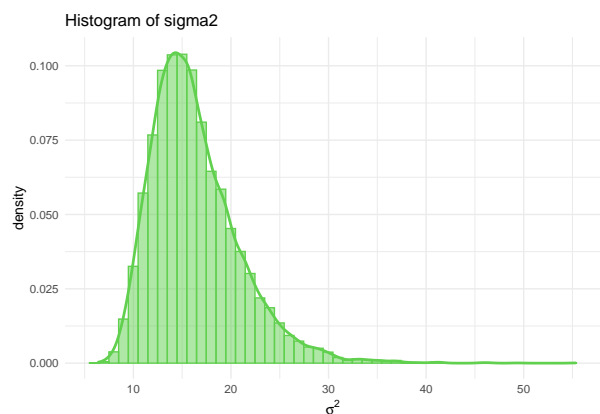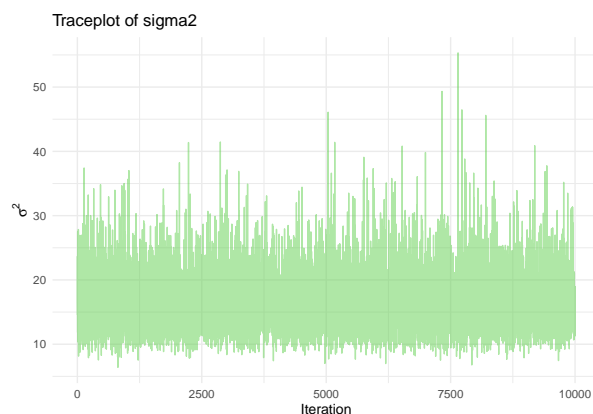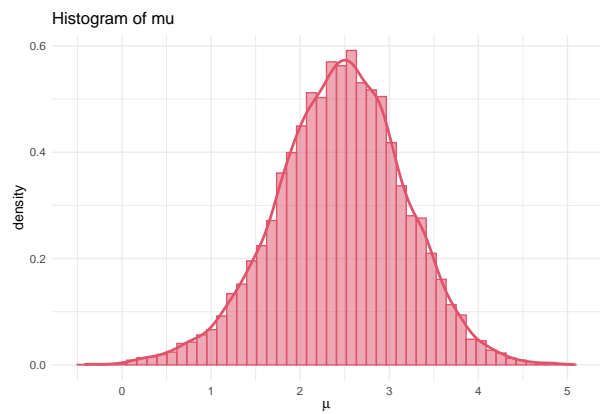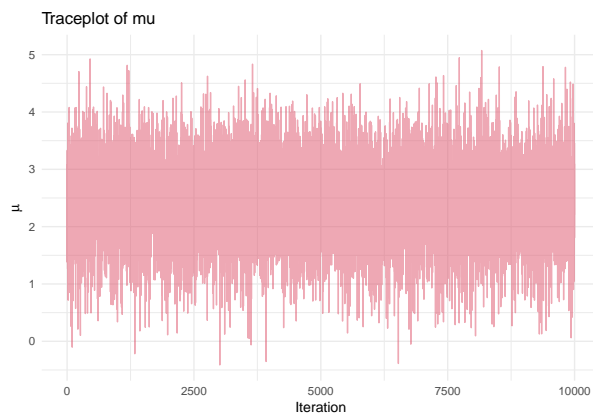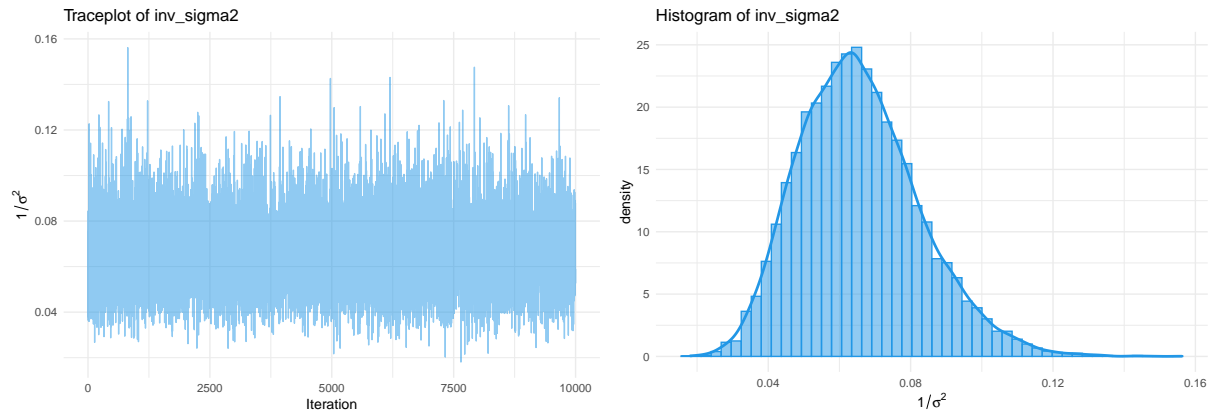
```r
d.summary <- t(
  rbind(
    colMeans(d.gibbs),
    sapply(d.gibbs, function(x) sd(x)),
    sapply(d.gibbs, function(x) quantile(x, probs=c(0.025, 0.5, 0.975)))
  )
)


d.summary <- data.frame(d.summary)
colnames(d.summary) <- c("Mean", "SD", "2.5%", "Median", "97.5%")

knitr::kable(d.summary, align="c", caption="Summary statistics of the marginal posteriors")
```

Table 1: Summary statistics of the marginal posteriors

	Mean	SD	2.5%	Median	97.5%
mu	2.4479404	0.7206035	0.9436683	2.4692436	3.808022
sigma2	16.3331563	4.5483340	9.7333046	15.5669100	27.221752
inv_sigma2	0.0656163	0.0169525	0.0367353	0.0642388	0.102740

```r
# plots (compare with INLA)
# INLA is exact

# Question: How long should we run the MCMC Gibbs chain to get close to the exact INLA approximation
# Run for 10 min, for 30 min, for 1h...

library(INLA)
library(MASS)

formula <- y ~ 1
inla.output <- inla(formula,data=data.frame(y=y),
                    control.family = list(hyper =
                                      list(prec = list(prior="loggamma",param=c(a0,b0)))),
                    control.fixed = list(mean.intercept=mu0, prec.intercept=1/sigma2_0))


par(mfrow=c(1,1))
# plot for mean
rg <- range(inla.output$marginals.fixed$"(Intercept)"[,2])
truehist(mu_gibbs_samples, prob=TRUE, col="yellow", xlab=expression(mu))
lines(density(mu_gibbs_samples),lty=3,lwd=3, col=2)
lines(inla.output$marginals.fixed$"(Intercept)",lwd=2)
```

```
legend("topright",c("MCMC, Gibbs","INLA"),lty=c(3,1),lwd=c(2,2),col=c(2,1),cex=1.0,bty="n")

# plot for variance
m_var <-inla.tmarginal(function(x) 1/x, inla.output$marginals.hyperpar[[1]])
truehist(sigma2_gibbs_samples, prob=TRUE, col="yellow", xlab=expression(sigma^2))
lines(density(sigma2_gibbs_samples),lty=3,lwd=3, col=2)
lines(m_var,lwd=2)
legend("topright",c("MCMC, Gibbs","INLA"),lty=c(3,1),lwd=c(2,2),col=c(2,1),cex=1.0,bty="n")

# plot for precision
truehist(inv_sigma2_gibbs_samples, prob=TRUE, col="yellow", xlab=expression(1/sigma^2))
lines(density(inv_sigma2_gibbs_samples),lty=3,lwd=3, col=2)
lines(inla.output$marginals.hyperpar[[1]],lwd=2)
legend("topright",c("MCMC, Gibbs","INLA"),lty=c(3,1),lwd=c(2,2),col=c(2,1),cex=1.0,bty="n")
```
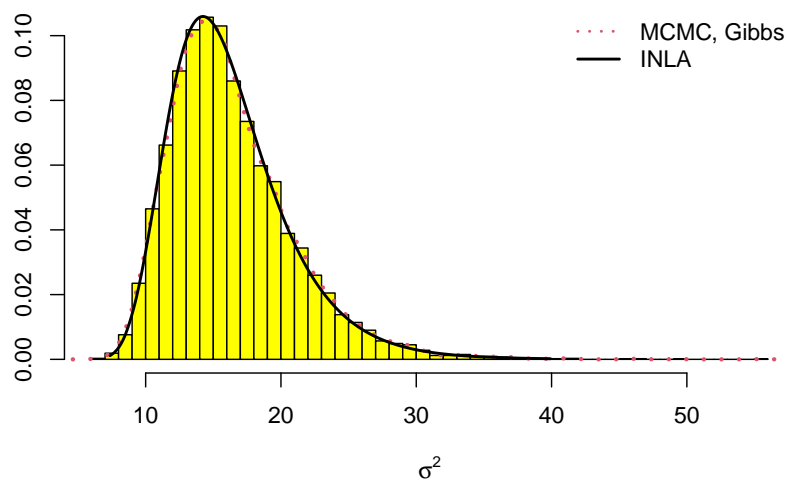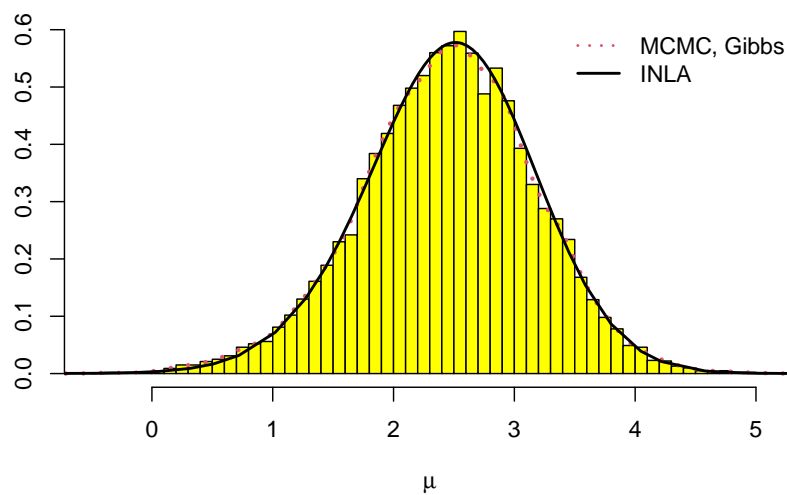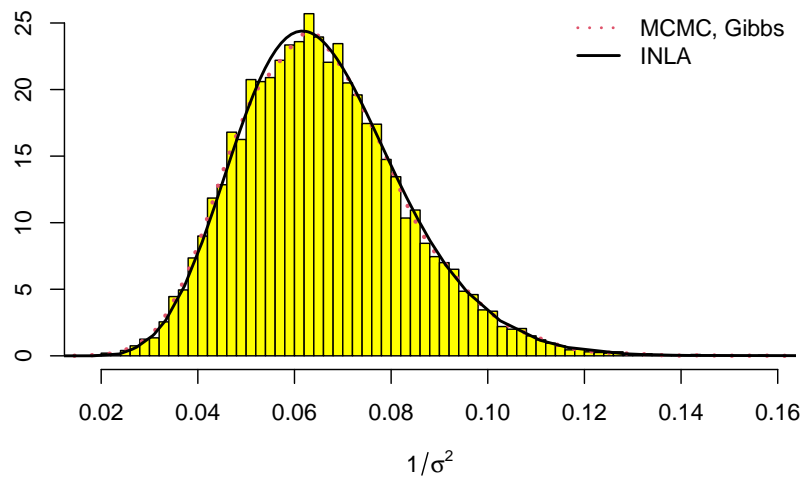
Exercise 3 (Metropolis-Hastings sampler for a logistic regression)

3.1

Explain Metropolis-Hastings sampler

3.2

```r
MH.sampler <- function(x,          # covariate values
                       y,          # number of mice deaths
                       n,          # total number of mice
                       sigma2 = 10^(4), # variance of normal priors
                       n.iter = 10000,  # number of MCMC iterations
                       n.burnin = 4000, # burnin length
                       n.thin = 1,      # thinning parameter
                       alpha = 0,     # starting point
                       beta = 0,      # starting point
                       s_alpha = 1,   # SD for normal proposal
                       s_beta = 60    # SD for normal proposal
                       ) {

  ####################
  ## Bayesian analysis
  ####################

  # inverse logit: logit^(-1)(alpha + beta*x)
  mypi <- function(alpha, beta, x){
    tmp <- exp(alpha + beta*x)
    pi <- tmp/(1+tmp)
    return(pi)
  }




  ###################################
  ## Step 1: R: (univariate proposal) Metropolis MCMC settings
  ###################################
```

```r
alpha_samples <- c()
beta_samples <- c()
# number of accepted proposals
alpha_yes <- 0
beta_yes <- 0

# counter
count <- 0

# start the MCMC algorithm (the first iteration after the burn-in is 1)
for(i in -n.burnin:(n.iter*n.thin)){
  count <- count + 1

  ## update alpha
  # generate a new proposal for alpha
  alpha_star <- rnorm(1, alpha, sd=s_alpha)

  # NOTE: it is more stable to calculate everything on the log scale
  enum <- sum(dbinom(y, size=n, prob=mypi(alpha_star, beta, x), log=TRUE)) +
    dnorm(alpha_star, mean=0, sd=sqrt(sigma2), log=TRUE)
  denom <- sum(dbinom(y, size=n, prob=mypi(alpha, beta, x), log=TRUE))  +
    dnorm(alpha, mean=0, sd=sqrt(sigma2), log=TRUE)

  # log accetpance rate (since we use a random walk proposal there is no
  #   proposal ratio in the acceptance probability)
  logacc <- enum - denom
  if(log(runif(1)) <= logacc){
    # accept the proposed value
    alpha <- alpha_star
    alpha_yes <- alpha_yes + 1
  }

  ## update beta
  # generate a new proposal for beta
  beta_star <- rnorm(1, beta, sd=s_beta)

  enum <- sum(dbinom(y, size=n, prob=mypi(alpha, beta_star, x), log=TRUE)) +
    dnorm(beta_star, mean=0, sd=sqrt(sigma2), log=TRUE)
  denom<- sum(dbinom(y, size=n, prob=mypi(alpha, beta, x), log=TRUE)) +
    dnorm(beta, mean=0, sd=sqrt(sigma2), log=TRUE)
  # log accetpance rate
  logacc <- enum - denom

  if(log(runif(1)) <= logacc){
    # accept the proposed value
    beta <- beta_star
    beta_yes <- beta_yes + 1
  }

  # after the burnin save every kth sample
  if((i > 0) && (i%%n.thin == 0)){
    alpha_samples <- c(alpha_samples, alpha)
    beta_samples <- c(beta_samples, beta)
  }
  if(i%%1000 ==0){
    # print the acceptance rates on the fly
    cat(c(i, alpha_yes/count, beta_yes/count), "\n")
```

```
    }
  }
  return(list(alpha_samples=alpha_samples,
              beta_samples=beta_samples))
}
```

```
x_original <- c(0.0028, 0.0028, 0.0056, 0.0112, 0.0225, 0.0450)
# the centered covariate values (centered dose)
x <- x_original - mean(x_original)
# number of mice deaths
# y <- c(35, 21, 9, 6, 1)
y <- c(26, 9, 21, 9, 6, 1)
# total number of mice
# n <- c(40, 40, 40, 40, 40)
n <- c(28, 12, 40, 40, 40, 40)
```

```
d.mice <- data.frame(
  x_original, y, n, x, y/n
)
colnames(d.mice) <- c("$x$", "$y$", "$n$", "centered $x$", "$p$")
knitr::kable(d.mice, align="c", caption="Mice data from Collett (2003)")
```

Table 2: Mice data from Collett (2003)

x	y	n	centered x	p
0.0028	26	28	-0.0121833	0.9285714
0.0028	9	12	-0.0121833	0.7500000
0.0056	21	40	-0.0093833	0.5250000
0.0112	9	40	-0.0037833	0.2250000
0.0225	6	40	0.0075167	0.1500000
0.0450	1	40	0.0300167	0.0250000

```
low <- c(0.01, 1)
middle <- c(1, 100)
high <- c(50, 5000)
```

```
set.seed(44566)
cat("\n---------Low values---------\n")
low.sampler <- MH.sampler(x=x, y=y, n=n, s_alpha=low[1], s_beta=low[2])

cat("\n---------Middle values---------\n")
middle.sampler <- MH.sampler(x=x, y=y, n=n, s_alpha=middle[1], s_beta=middle[2])

cat("\n---------High values---------\n")
high.sampler <- MH.sampler(x=x, y=y, n=n, s_alpha=high[1], s_beta=high[2])
```

```
##
## ---------Low values---------
## -4000 1 0
## -3000 0.9440559 0.9090909
## -2000 0.966017 0.9355322
## -1000 0.9723426 0.9510163
## 0 0.9740065 0.9562609
## 1000 0.9696061 0.9638072
## 2000 0.9691718 0.9671721
```

```
## 3000 0.9714327 0.9705756
## 4000 0.9715036 0.9703787
## 5000 0.9726697 0.9706699
## 6000 0.9740026 0.9720028
## 7000 0.975275 0.9740933
## 8000 0.9753354 0.9750021
## 9000 0.9760018 0.9746173
## 10000 0.976216 0.974359
##
## ---------Middle values---------
## -4000 0 0
## -3000 0.2297702 0.2327672
## -2000 0.2278861 0.2358821
## -1000 0.2172609 0.2372542
## 0 0.2124469 0.2386903
## 1000 0.2167566 0.2343531
## 2000 0.2132978 0.2287952
## 3000 0.2141123 0.2286816
## 4000 0.2144732 0.2294713
## 5000 0.213754 0.229419
## 6000 0.2141786 0.230377
## 7000 0.216344 0.2305245
## 8000 0.2147321 0.2301475
## 9000 0.2136759 0.229367
## 10000 0.2134133 0.2302693
##
## ---------High values---------
## -4000 0 0
## -3000 0.003996004 0.004995005
## -2000 0.004497751 0.005497251
## -1000 0.004331889 0.005331556
## 0 0.003999 0.004248938
## 1000 0.00339932 0.00479904
## 2000 0.003832695 0.004832528
## 3000 0.004142265 0.004713612
## 4000 0.0039995 0.004624422
## 5000 0.003777358 0.004332852
## 6000 0.00369963 0.00439956
## 7000 0.004272339 0.004454141
## 8000 0.004249646 0.004582951
## 9000 0.004461195 0.004538112
## 10000 0.004713949 0.004571102
```

When the tuning parameters of the proposals are set to middle $(1, 100)$, the optimal acceptance rate is attained.

```
alpha_low <- low.sampler$alpha_samples
beta_low <- low.sampler$beta_samples

alpha_middle <- middle.sampler$alpha_samples
beta_middle <- middle.sampler$beta_samples

alpha_high <- high.sampler$alpha_samples
beta_high <- high.sampler$beta_samples

plot(x=1:length(alpha_low), y=alpha_low, type="l", col=2,
     xlab="Iteration", ylab=expression(alpha), main="Traceplot of alpha (low)")
plot(x=1:length(alpha_middle), y=alpha_middle, type="l", col=3,
```

```
      xlab="Iteration", ylab=expression(alpha), main="Traceplot of alpha (middle)")
plot(x=1:length(alpha_high), y=alpha_high, type="l", col=4,
     xlab="Iteration", ylab=expression(alpha), main="Traceplot of alpha (high)")

plot(x=1:length(beta_low), y=beta_low, type="l", col=2,
     xlab="Iteration", ylab=expression(beta), main="Traceplot of beta (low)")
plot(x=1:length(beta_middle), y=beta_middle, type="l", col=3,
     xlab="Iteration", ylab=expression(beta), main="Traceplot of beta (middle)")
plot(x=1:length(beta_high), y=beta_high, type="l", col=4,
     xlab="Iteration", ylab=expression(beta), main="Traceplot of beta (high)")

acf(alpha_low, col=2, main="Autocorrelation plot of alpha (low)")
acf(alpha_middle, col=3, main="Autocorrelation plot of alpha (middle)")
acf(alpha_high, col=4, main="Autocorrelation plot of alpha (high)")

acf(beta_low, col=2, main="Autocorrelation plot of beta (low)")
acf(beta_middle, col=3, main="Autocorrelation plot of beta (middle)")
acf(beta_high, col=4, main="Autocorrelation plot of beta (high)")

ccf(alpha_low, beta_low, col=2, main="Crosscorrelation plot of alpha and beta (low)")
ccf(alpha_middle, beta_middle, col=3, main="Crosscorrelation plot of alpha and beta (middle)")
ccf(alpha_high, beta_high, col=4, main="Crosscorrelation plot of alpha and beta (high)")
```
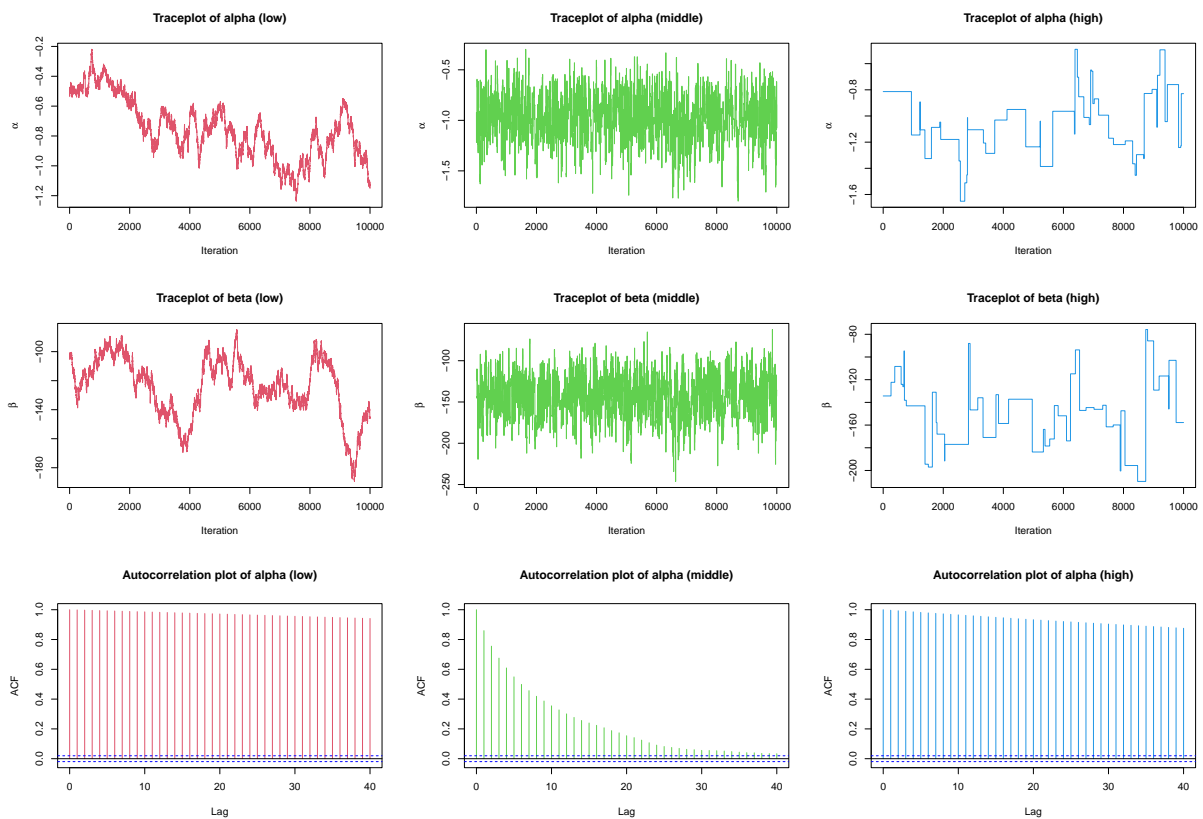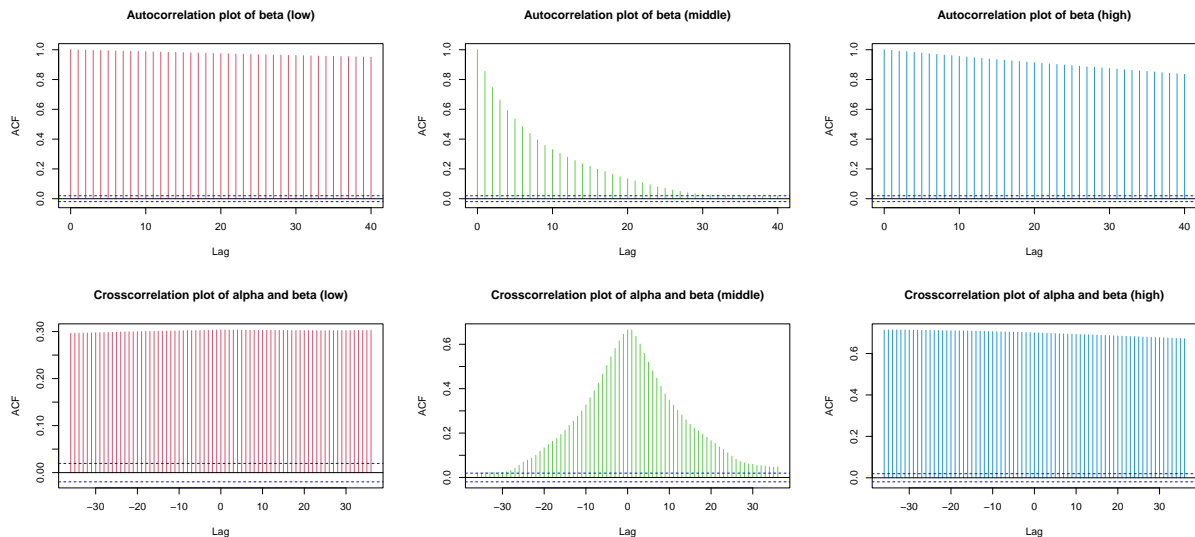
Autocorrelation plot of beta (low)   Autocorrelation plot of beta (middle)   Autocorrelation plot of beta (high)

Crosscorrelation plot of alpha and beta (low)   Crosscorrelation plot of alpha and beta (middle)   Crosscorrelation plot of alpha and beta (high)

```r
d.MH <- data.frame(
  alpha = alpha_middle,
  beta  = beta_middle
)

ggplot(d.MH, aes(x=alpha, y=..density..)) +
  geom_histogram(color=2, fill=2, alpha=0.5, bins=50) +
  geom_density(color=2, lwd=1) +
  labs(title="Histogram of alpha", x=expression(alpha)) +
  theme_minimal()

ggplot(d.MH, aes(x=beta, y=..density..)) +
  geom_histogram(color=3, fill=3, alpha=0.5, bins=50) +
  geom_density(color=3, lwd=1) +
  labs(title="Histogram of beta", x=expression(beta)) +
  theme_minimal()
```


Histogram of alpha   Histogram of beta
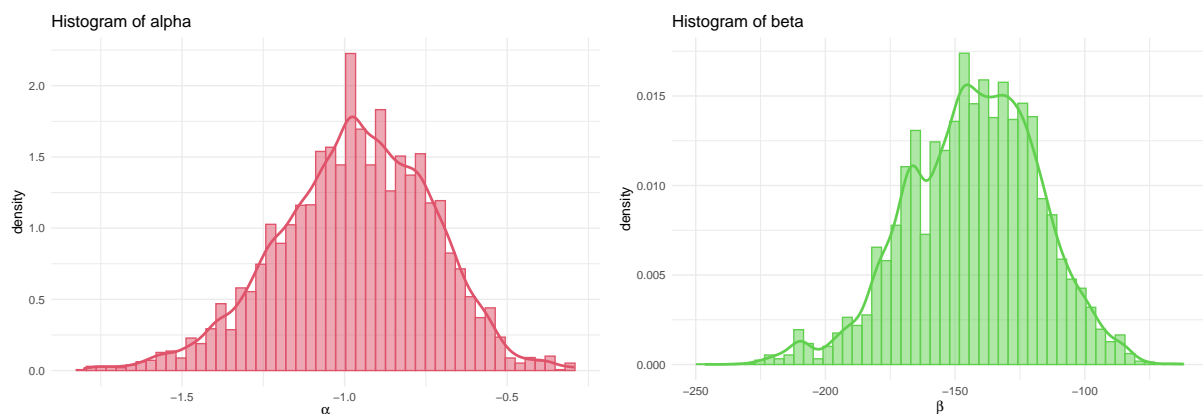
```r
d.summary <- t(
  rbind(
    colMeans(d.MH),
    sapply(d.MH, function(x) sd(x)),
    sapply(d.MH, function(x) quantile(x, probs=c(0.025, 0.5, 0.975)))
  )
)
```

```
d.summary <- data.frame(d.summary)
colnames(d.summary) <- c("Mean", "SD", "2.5%", "Median", "97.5%")

knitr::kable(d.summary, align="c",
             caption="Summary statistics of marginal posteriors (middle choice)")
```

Table 3: Summary statistics of marginal posteriors (middle choice)

	Mean	SD	2.5%	Median	97.5%
alpha	-0.9673889	0.2352573	-1.461707	-0.9592253	-0.5537754
beta	-143.0714300	25.3517136	-196.708001	-142.0147341	-96.3710245

```
med.alpha <- d.summary$Median[1]
med.beta <- d.summary$Median[2]

# inverse logit: logit^(-1)(alpha + beta*x)
mypi <- function(alpha, beta, x){
  tmp <- exp(alpha + beta*x)
  pi <- tmp/(1+tmp)
  return(pi)
}

x.grid <- seq(min(x), max(x), length.out=100)
y.pred <- mypi(alpha=med.alpha, beta=med.beta, x=x.grid)

plot(x=x, y=y/n, col=2, xlab="centered dose", ylab="response probability",
     main="Logistic curve with data")
points(x=x, y=mypi(alpha=med.alpha, beta=med.beta, x=x), col=3, pch=2)
lines(y.pred ~ x.grid, col=4)
legend("topright", legend=c("data", "predictions", "logistic curve"),
       col=2:4, lty=c(NA, NA, 1), pch=c(1, 2, NA))
```