NAME: Kaifa Lu UFID: 5205-0501
STA6348 – Bayesian Analysis for Machine Learning and Uncertainty Quantification

Problem 1: Gibbs vs composition sampling vs random walk Metropolis-Hastings (RWMH).

1.1 The composition sampler for sampling from f_{12} that iterates:

(1) Independent sampling of X_1: $X_1 \sim N(0,1)$

(2) Conditional sampling of X_2: $X_2 \sim N(\rho X_1, [1-\rho^2])$

Let $n = 1000$ and $\rho = 0.9$, we derive the sequence $(X_1^1, X_2^1), \cdots, (X_1^n, X_2^n)$ of rvs produced by the composition sampler and plot the sequence in Fig. 1. It can be easily found that the two vectors X_1, X_2 are not independent of each other.

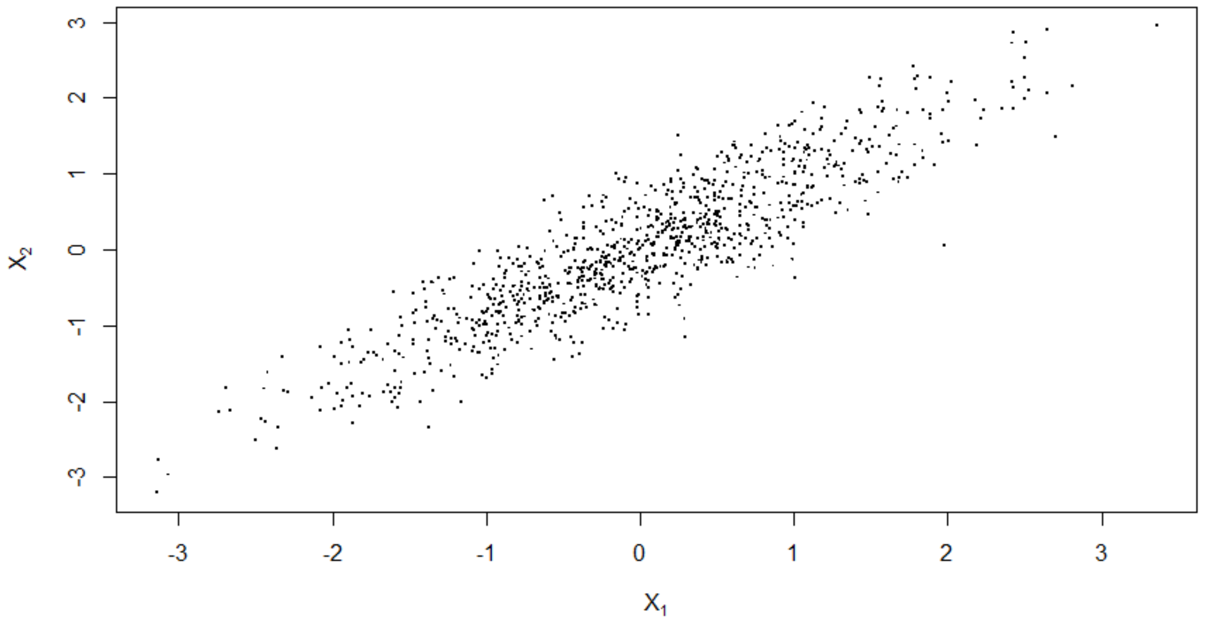


Fig. 1: Scatter plot of the two vectors X_1, X_2 produced by the composition sampler

1.2 The "Metropolized" composition sampler for sampling from f_{12} that iterates:

(1) A random walk Metropolis-Hastings sampling of X_1:

$$p(X_1) \propto exp(-X_1^2/2)$$

Use $N(X_1^t, 1)$ as the proposal, then propose $X_1^* \sim N(X_1^t, 1)$

Accept $X_1^{t+1} = X_1^*$ with probability $min\{1, r\}$ with $r = \frac{p(X_1^*)}{p(X_1^t)}$, otherwide $X_1^{t+1} = X_1^t$.

(2) Conditional sampling of X_2: $X_2 \sim N(\rho X_1, [1-\rho^2])$

Let $n = 1000$ and $\rho = 0.9$, we derive the sequence $(X_1^1, X_2^1), \cdots, (X_1^n, X_2^n)$ of rvs produced by the "Metropolized" composition sampler and plot the sequence in Fig. 2. It can be easily found that the two vectors X_1, X_2 are not independent of each other, either.

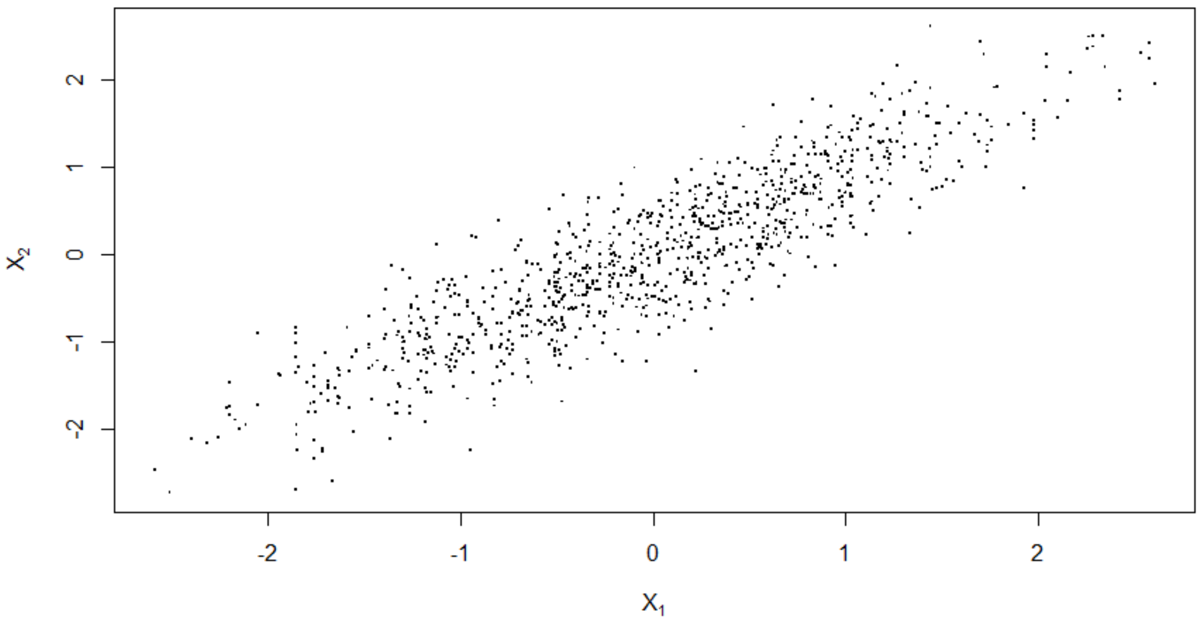


Fig. 2: Scatter plot of the two vectors X_1, X_2 produced by the "Metropolized" composition sampler

1.3 A random walk Metropolis-Hastings algorithm for jointly sampling X_1, X_2 is:

$$p(X_1, X_2) \propto exp\left(-\frac{(X_1^2 + X_2^2 - 2\rho X_1 X_2)}{2(1-\rho^2)}\right)$$

Perform a random walk Metropolis-Hastings algorithm using a normal proposal, i.e., if X_1^t, X_2^t are the current values for X_1, X_2, then

$$\begin{pmatrix} X_1^* \\ X_2^* \end{pmatrix} \sim N\left(\begin{bmatrix} X_1^t \\ X_2^t \end{bmatrix}, S\right)$$

Where S is the tuning parameter.

Based on the lectures, we know the optimal random walk tuning parameter is $2.4^2 Var(X)/d$ where $Var(X)$ is the unknown covariance matrix. We can estimate $Var(X)$ using the sample coveraiance matrix of draws. Therefore, the automatic adapting of the Metropolis-Hastings tuning parameter S is:

(1) Start with $S = diag(2)$
(2) Run 50 iterations of the MCMC using $2.4^2 S/d$
(3) Set S to the sample covariance matrix of all previous draws

Accept $\begin{pmatrix} X_1^{t+1} \\ X_2^{t+1} \end{pmatrix} = \begin{pmatrix} X_1^{*} \\ X_2^{*} \end{pmatrix}$ with probability $min\{1, r\}$ with $r = \frac{p(X_1^*, X_2^*)}{p(X_1^t, X_2^t)}$, otherwide $\begin{pmatrix} X_1^{t+1} \\ X_2^{t+1} \end{pmatrix} = \begin{pmatrix} X_1^{t} \\ X_2^{t} \end{pmatrix}$.

Let $n = 1000$ and $\rho = 0.9$, we derive the sequence $(X_1^1, X_2^1), \cdots, (X_1^n, X_2^n)$ of rvs produced by the random walk Metropolis-Hastings algorithm for joint sampling X_1, X_2 and plot the sequence in Fig. 3.

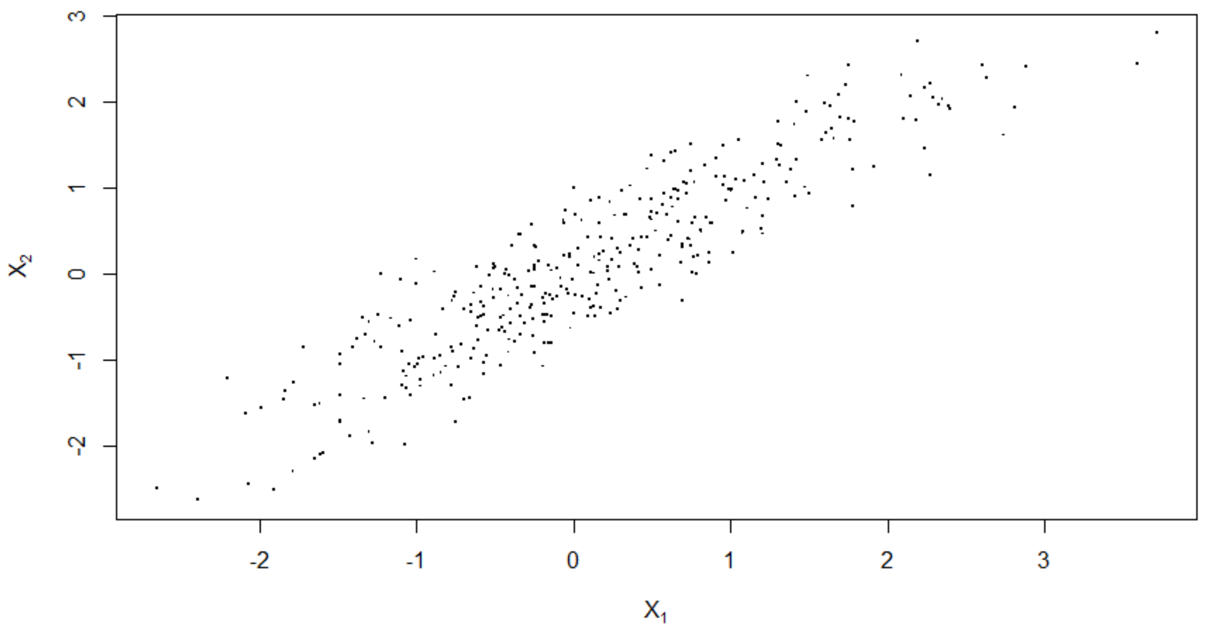


Fig. 3: Scatter plot of the two vectors X_1, X_2 produced by the random walk Metropolis-Hastings algorithm for jointly sampling X_1, X_2

1.4 The Gibbs sampler for sampling from f_{12} that iterates:

Beginning with an initial value (X_1^0, X_2^0), an iteration of the Gibbs sampler involves:

(1) Conditional sampling of X_1: $X_1^{t+1} \sim N(\rho\, X_2^{t+1}, [1 - \rho^2])$

(2) Conditional sampling of X_2: $X_2^{t+1} \sim N(\rho\, X_1^{t}, [1 - \rho^2])$

Let $n = 1000$ and $\rho = 0.9$, we derive the sequence $(X_1^1, X_2^1), \cdots, (X_1^n, X_2^n)$ of rvs produced by the Gibbs sampler and plot the sequence in Fig. 4.

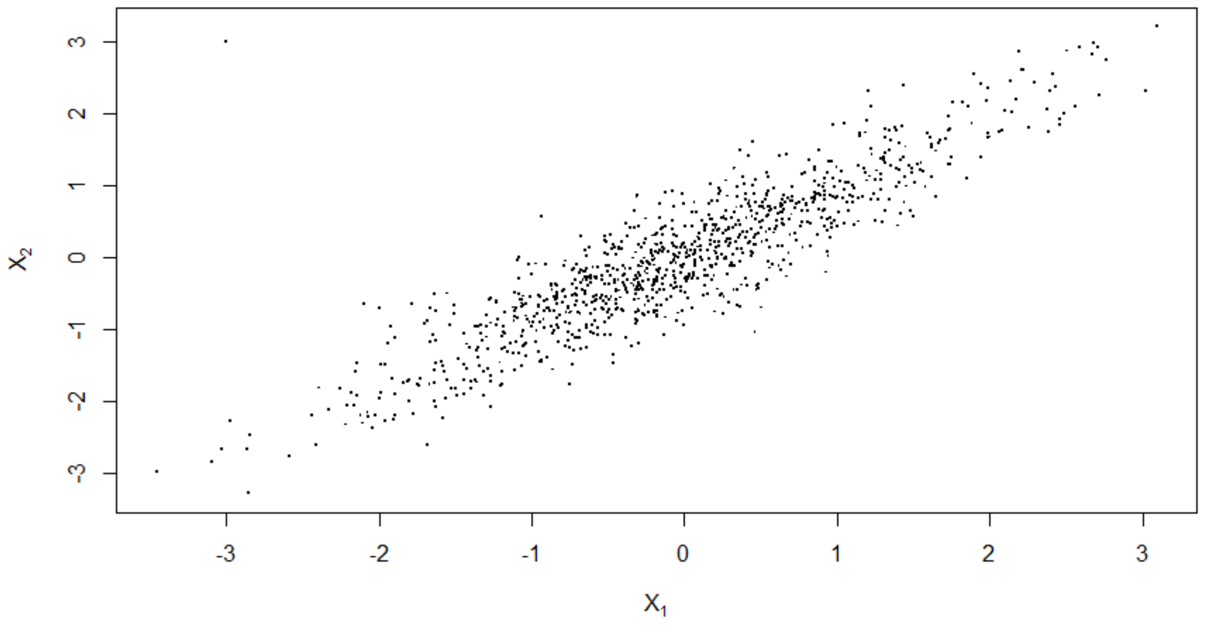


Fig. 4: Scatter plot of the two vectors X_1, X_2 produced by the Gibbs sampler

1.5 Let $n = 1000$ be the length of the Markov chain. For $\rho = 0.25, 0.5, 0.75, 0.9, 0.95, 0.99$, start the the three samplers (from 1.2, 1.3 and 1.4) at the origin, run them for n iterations. Visualize the autocorrelation in $X_1^1, X_1^2, \cdots, X_1^n$ in each chain, as shown in Fig. 5-7.

(1) The "Metropolized" composition sampler:

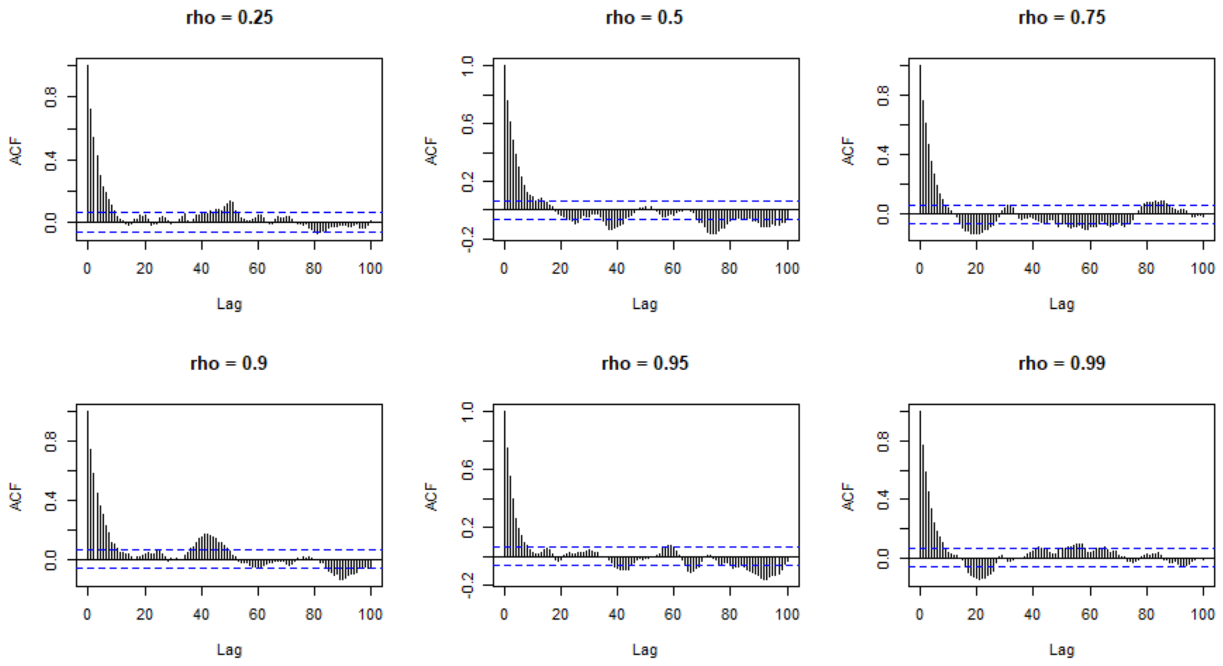


Fig. 5: Autocorrelation of the sequence X_1 produced by the "Metropolized" composition sampler

(2) A random walk Metropolis-Hastings algorithm for jointly sampling X_1, X_2:

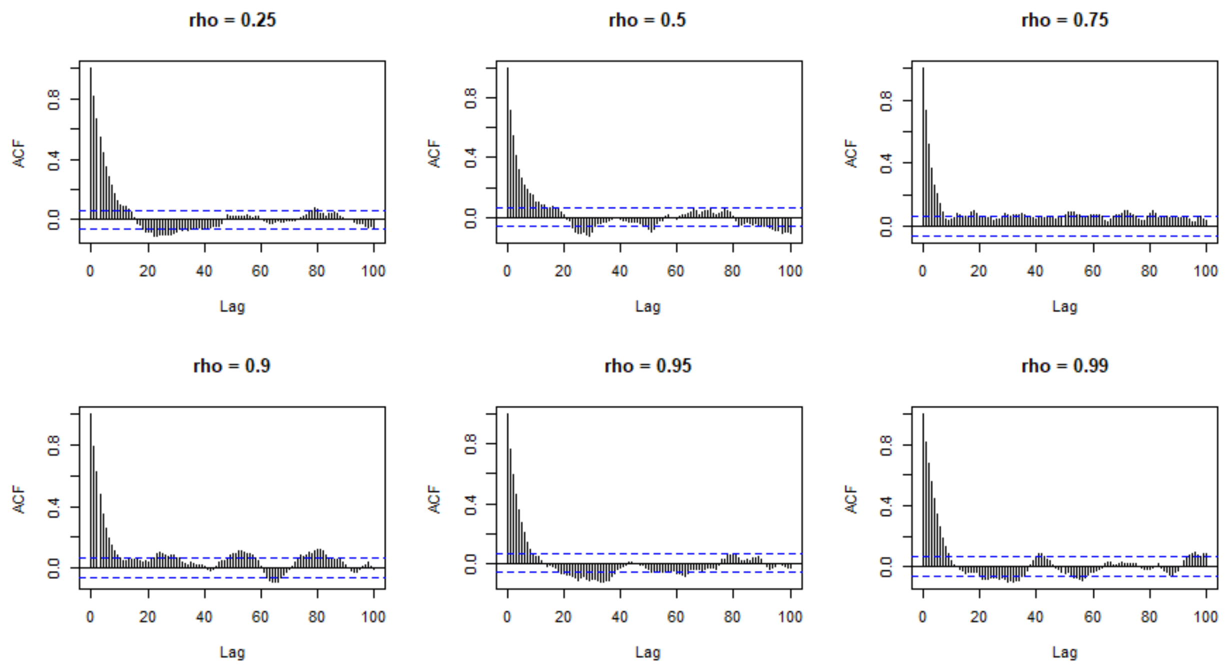


Fig. 6: Autocorrelation of the sequence X_1 produced by the random walk Metropolis-Hastings algorithm for jointly sampling X_1, X_2

(3) The Gibbs sampler:

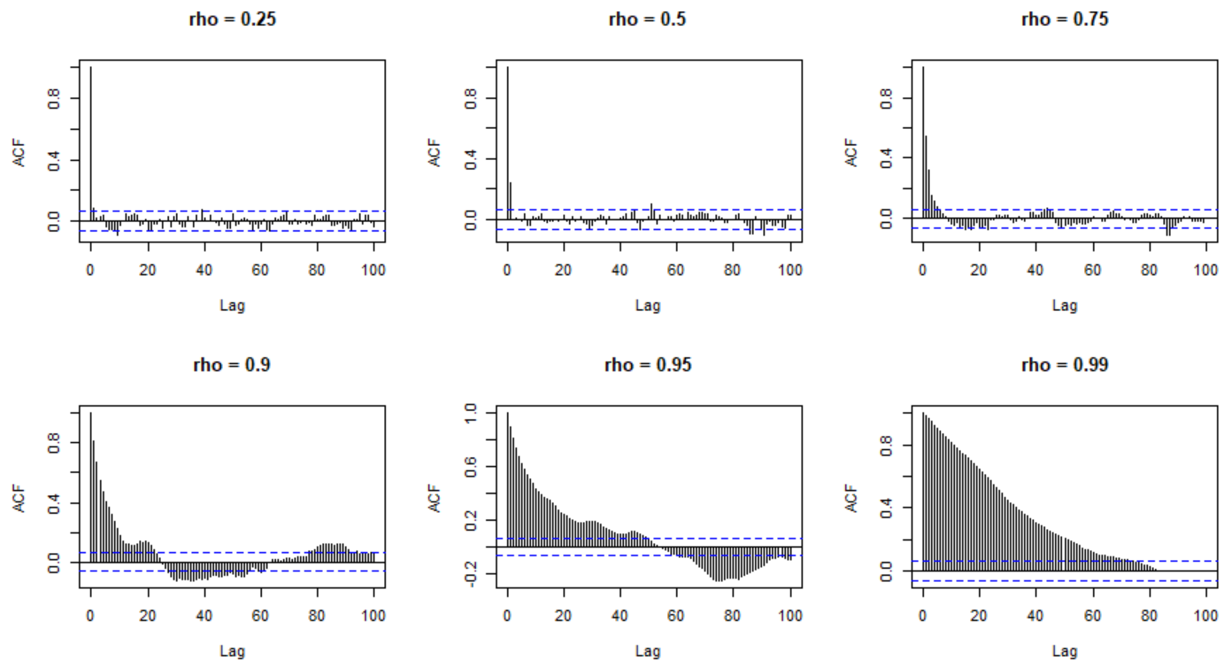


Fig. 7: Autocorrelation of the sequence X_1 produced by the Gibbs sampler

Table 1: Effective sample sizes of the sequence X_1 ($n = 1000$) produced by different samplers

rho	"Metropolized" composition sampler	A random-walk Metropolis-Hastings	Gibbs sampler
0.25	739	319	1000
0.5	692	430	1000
0.75	711	338	1000
0.9	719	372	1000
0.95	692	328	1000
0.99	723	243	1000

Based on the results from Fig. 5-7 and Table 1, we can find that when sampling the sequences X_1, X_2 in each chain, the Gibbs sampler performs better than the "Metropolized" composition sampler and the random-walk Metropolis-Hastings sampler for jointly sampling X_1, X_2, in terms of effective sample size. On the other hand, the ρ values hardly affect the effective sample sizes of the sequences X_1, X_2 in each chain, but they can significantly influence the autocorrelation of the sequences X_1, X_2 generated by the Gibbs sampler (i.e., larger ρ values correspond to stronger autocorrelation). In contrast, the autocorrelation of the sequences X_1, X_2 produced by the other two samplers is less affected by the ρ values.

1.6 Start the samplers far from the origin $\begin{pmatrix} X_1^0 \\ X_2^0 \end{pmatrix} = \begin{pmatrix} -10 \\ 10 \end{pmatrix}$ for $\rho = 0.25, 0.5, 0.99$. Then we plot the vector of states of X_2 against that of states of X_1 to visualize how the three samplers explore the support of the distribution and overplay it with the contours of the joint pdf. Also, we plot the states of each variable against the iteration number, as shown in Fig. 8-10 separately for three samplers.

(1) The "Metropolized" composition sampler:

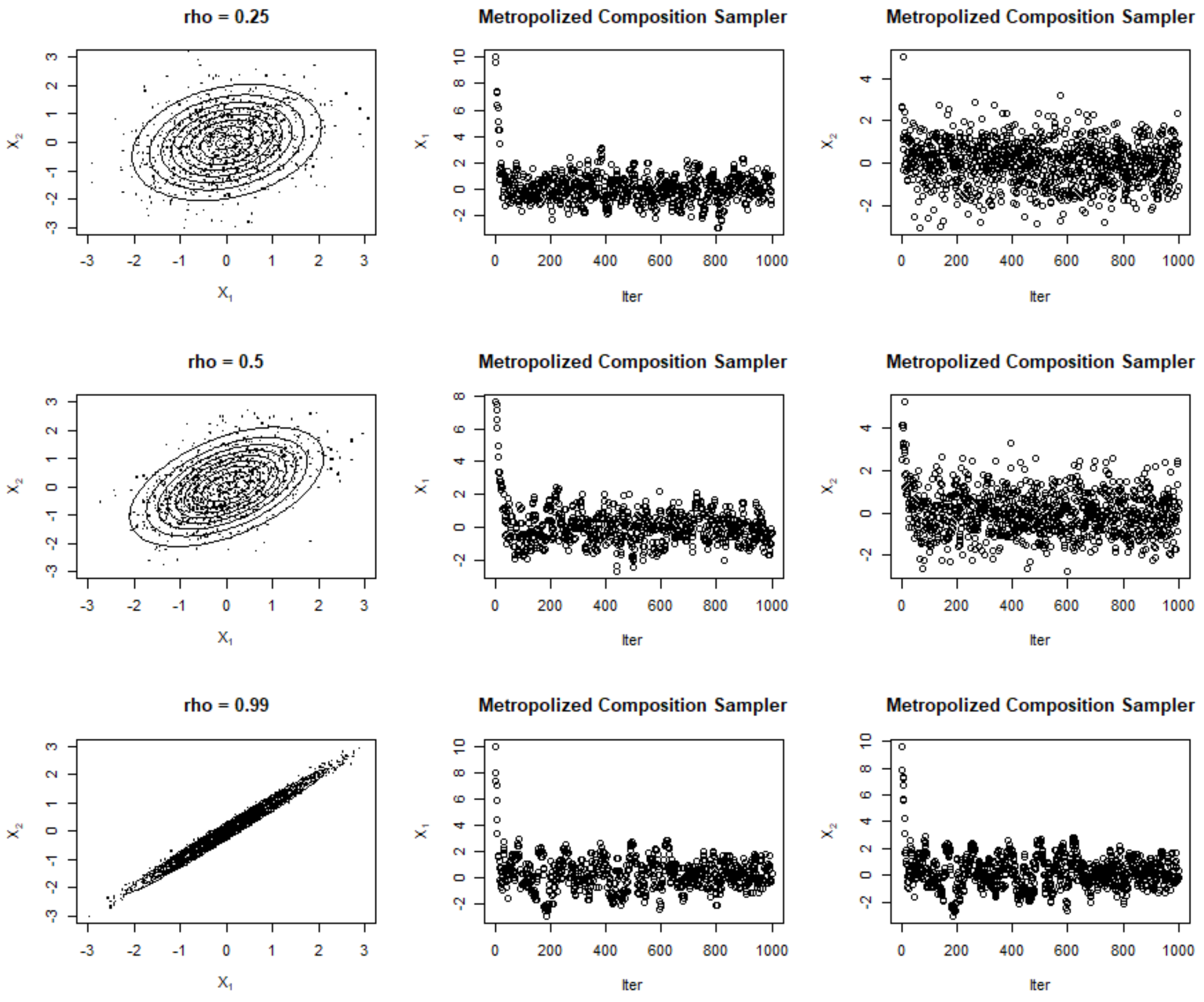


Fig. 8: Plots of the sequences X_1, X_2 produced by the “Metropolized” composition sampler: (a) The three rows separately stand for the scenarios of $\rho = 0.25, 0.5, 0.99$; (b) The first column refers to the support of the distribution and the contours of the joint pdf; (c) The second column refers to the states of X_1 against the iteration number; (d) The third column refers to the states of X_2 against the iteration number.

For the “Metropolized” composition sampler, most of the states of X_1, X_2 are overlaid with the contour plots of X_1, X_2. When the starting point of X_1 is far away from the origin, it just takes a few number of iterations to reach the high-probability density region of X_1. As ρ increases from 0.25 to 0.9, it takes a gradually increasing number of iterations to reach the high-probability density region of X_2.

(2) A random walk Metropolis-Hastings algorithm for jointly sampling X_1, X_2:

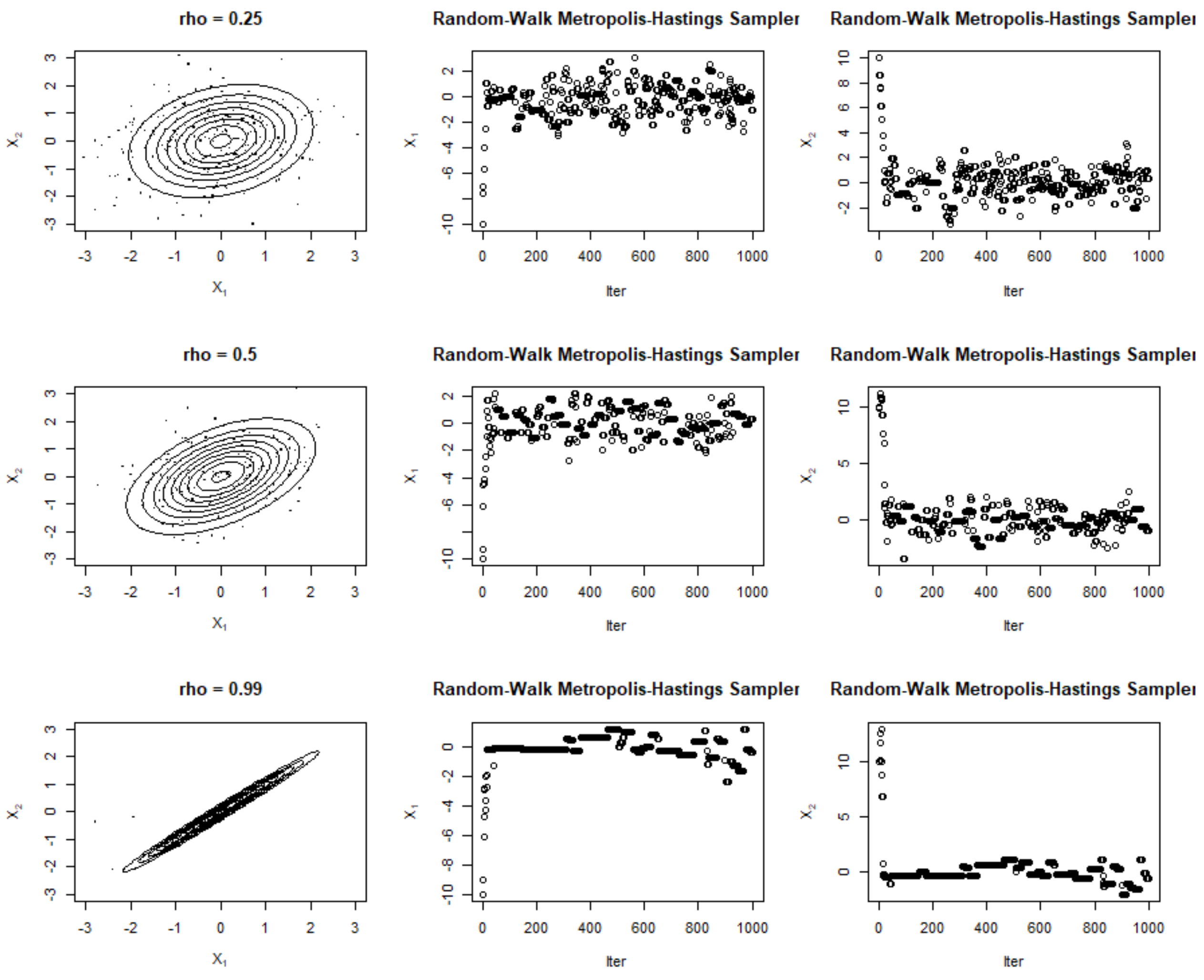


Fig. 9: Plots of the sequences X_1, X_2 produced by the random-walk Metropolis-Hastings sampler: (a) The three rows separately stand for the scenarios of $\rho = 0.25, 0.5, 0.99$; (b) The first column refers to the support of the distribution and the contours of the joint pdf; (c) The second column refers to the states of X_1 against the iteration number; (d) The third column refers to the states of X_2 against the iteration number.

For the random-walk Metropolis-Hastings sampler, it exhibits almost the same patterns as the "Metropolized" composition sampler, in terms of sampling X_1, X_2. The main difference is that the effective sample size of the random-walk Metropolis-Hastings sampler is significantly less than that of the "Metropolized" composition sampler.

(3) The Gibbs sampler:

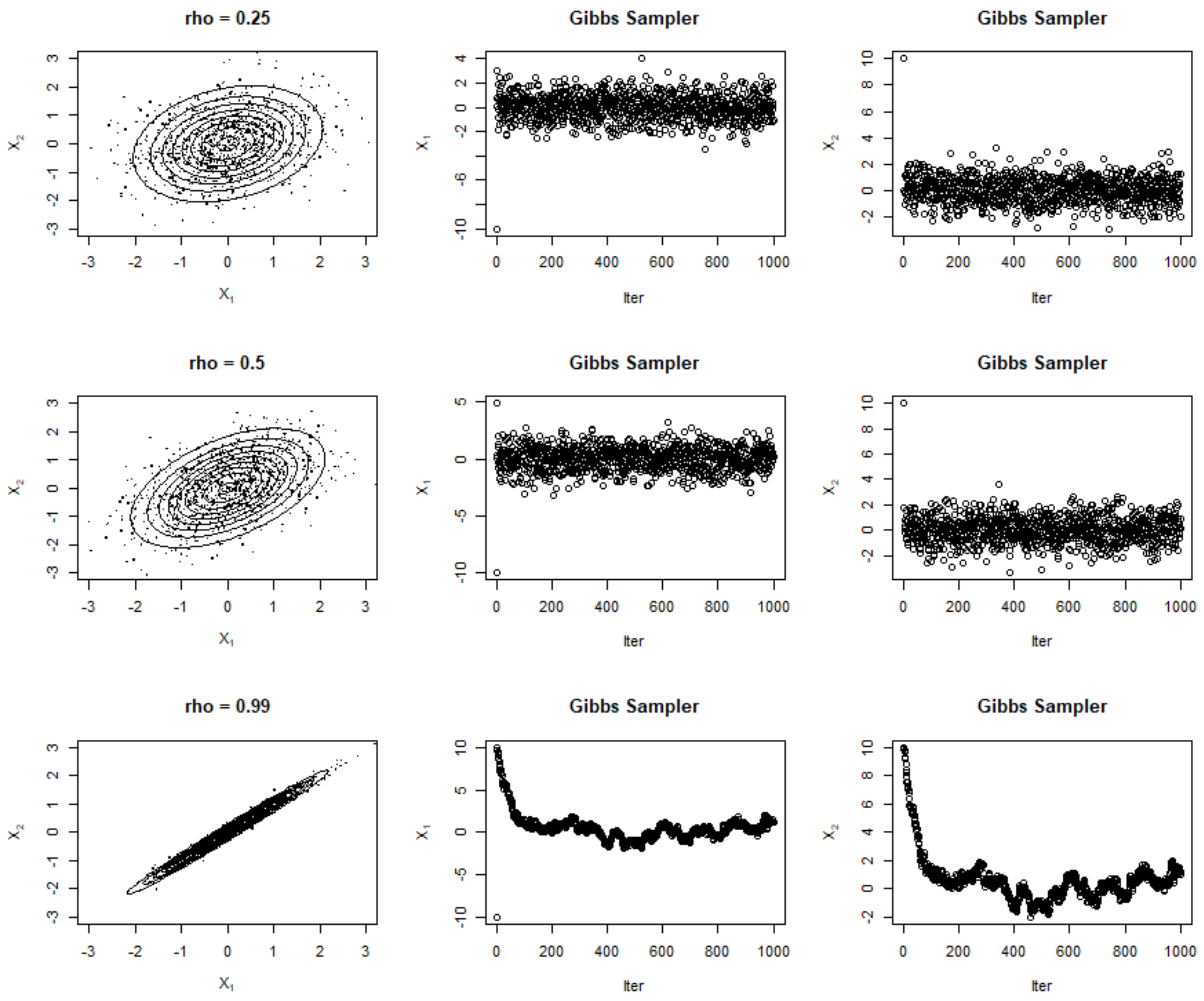
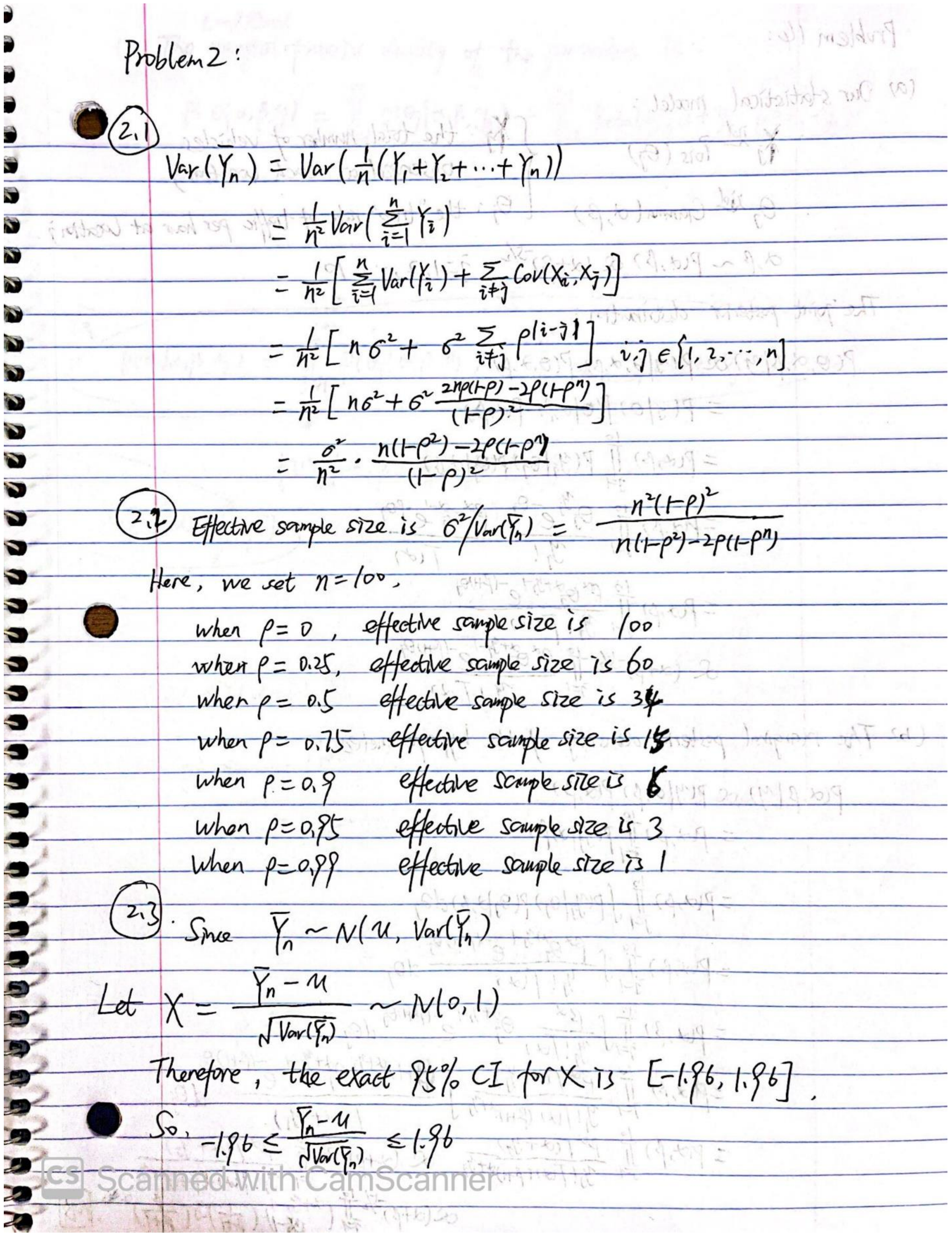


Fig. 10: Plots of the sequences X_1, X_2 produced by the Gibbs sampler: (a) The three rows stand for the scenarios of $\rho = 0.25, 0.5, 0.99$, respectively; (b) The first column refers to the support of the distribution and the contours of the joint pdf; (c) The second column refers to the states of X_1 against the iteration number; (d) The third column refers to the states of X_2 against the iteration number.

For the Gibbs sampler, similarly, most of the states of X_1, X_2 are overlaid with the contour plots of X_1, X_2. However, when the starting point of X_1, X_2 is far away from the origin, it takes a larger number of iterations to reach the high-probability density region of X_1, X_2, particulalry when ρ increases from 0.25 to 0.9.

Problem 2: Effective Sample Size in an AR(1) model.



Problem 2:

2.1

$$Var(\bar{Y}_n) = Var\left(\frac{1}{n}(Y_1+Y_2+\cdots+Y_n)\right)$$
$$= \frac{1}{n^2} Var\left(\sum_{i=1}^{n} Y_i\right)$$
$$= \frac{1}{n^2}\left[\sum_{i=1}^{n} Var(Y_i) + \sum_{i\neq j} Cov(X_i, X_j)\right]$$
$$= \frac{1}{n^2}\left[n\sigma^2 + \sigma^2 \sum_{i\neq j} \rho^{|i-j|}\right] \quad i,j \in \{1,2,\cdots,n\}$$
$$= \frac{1}{n^2}\left[n\sigma^2 + \sigma^2 \frac{2n\rho(1-\rho) - 2\rho(1-\rho^n)}{(1-\rho)^2}\right]$$
$$= \frac{\sigma^2}{n^2} \cdot \frac{n(1-\rho^2) - 2\rho(1-\rho^n)}{(1-\rho)^2}$$

2.2 Effective sample size is $\sigma^2 / Var(\bar{Y}_n) = \frac{n^2(1-\rho)^2}{n(1-\rho^2) - 2\rho(1-\rho^n)}$

Here, we set $n = 100$.

when $\rho = 0$, effective sample size is 100
when $\rho = 0.25$, effective sample size is 60
when $\rho = 0.5$ effective sample size is 34
when $\rho = 0.75$ effective sample size is 15
when $\rho = 0.9$ effective sample size is 6
when $\rho = 0.95$ effective sample size is 3
when $\rho = 0.99$ effective sample size is 1

2.3 Since $\bar{Y}_n \sim N(u, Var(\bar{Y}_n))$

Let $X = \frac{\bar{Y}_n - u}{\sqrt{Var(\bar{Y}_n)}} \sim N(0,1)$

Therefore, the exact 95% CI for X is $[-1.96, 1.96]$.

So, $-1.96 \le \frac{\bar{Y}_n - u}{\sqrt{Var(\bar{Y}_n)}} \le 1.96$

So the exact 95% CI for u is

$$\bar{Y}_n - 1.96\sqrt{Var(\bar{Y}_n)} \le u \le \bar{Y}_n + 1.96\sqrt{Var(\bar{Y}_n)}$$

$$\bar{Y}_n - 1.96\sigma\sqrt{\frac{n(1-\rho^2)-2\rho(1-\rho^n)}{n^2(1-\rho)^2}} \le u \le \bar{Y}_n + 1.96\sigma\sqrt{\frac{n(1-\rho^2)-2\rho(1-\rho^n)}{n^2(1-\rho)^2}}$$

So. the exact 95% CI for u is

$$\left[\bar{Y}_n - 1.96\sigma\sqrt{\frac{n(1-\rho^2)-2\rho(1-\rho^n)}{n^2(1-\rho)^2}}, \ \bar{Y}_n + 1.96\sigma\sqrt{\frac{n(1-\rho^2)-2\rho(1-\rho^n)}{n^2(1-\rho)^2}}\right]$$

where $\bar{Y}_n = \frac{1}{n}\sum_{i=1}^{n} Y_i$

(2.4) When n is held fixed and $\rho \to 1$,

$$\lim_{\rho\to1} \frac{n(1-\rho^2)-2\rho(1-\rho^n)}{n^2(1-\rho)^2} = \lim_{\rho\to1} \frac{-2n\rho-2+2(n+1)\rho^n}{2n^2(\rho-1)} = \lim_{\rho\to1} \frac{-2n+2n(n+1)\rho^{n-1}}{2n^2}$$

$$= \lim_{\rho\to1} \frac{-2n+2n^2+2n}{2n^2}$$

$$= 1$$

So when n is held fixed and $\rho \to 1$,

the exact 95% CI for u is approaching $[\bar{Y}_n - 1.96\sigma, \bar{Y}_n + 1.96\sigma]$

where $\bar{Y}_n = \frac{1}{n}\sum_{i=1}^{n} Y_i$.

(2.5) Here, when the rvs are positively correlated, we do not use conventional CIs. Instead, we have to consider the effect of correlations between the rvs. The main reason lies in that the data of the rvs are not independent of each other, instead, they are highly correlated.

2.6 Here, we simulate a time series of length n from a Gaussian $AR(1)$ process. Specifically, we separately set $n = 10^2, 10^3, 10^4$ and $\rho = 0{,}0.25{,}0.5{,}0.75{,}0.9{,}0.95{,}0.99$ to calculate and compare the effective sample size based on three different calculation methods, as presented in Table 2.

Table 2: Comparison of effective sample size by different calculation methods and n values

rho	0	0.25	0.5	0.75	0.9	0.95	0.99
Method	$n = 100$						
(i)	100	60	34	15	6	3	1
(ii)	35	28	22	14	10	8	7
(iii)	101	57	70	22	11	3	5
Method	$n = 1000$						
(i)	1,000	600	334	143	53	26	6
(ii)	333	258	222	140	99	77	61
(iii)	1,001	540	360	149	44	22	11
Method	$n = 10000$						
(i)	10,000	6,000	3,334	1,429	527	257	51
(ii)	3,317	2,762	2,100	1,423	942	753	609
(iii)	10,001	5,722	3,328	1,416	545	267	46

Note: (i) means the formula: effective sample size = $\sigma^2/Var(\bar{Y}_n)$;

(ii) refers to the "theoretical" formula: effective sample size = $n/(1 + 2\sum_{k=1}^{\infty} \rho_k)$ where ρ_k is the lag-k autocorrelation;

(iii) denotes that effective sample size is equal to the number of independent samples based on estimates from lecture slides.

Table 2 suggests that as ρ increases from 0 to 0.99, the effective sample size calculated whether method (i), (ii), or (iii) all decrease in general, particularly for those produced by method (i) and (iii). On the other hand, when $0 < \rho < 0.75$, the effective sample size calculated by method (i) and (iii) is significantly larger than that computed by method (ii); When $0.75 < \rho < 1$, the effective sample size calculated by method (ii) is larger than that computed by method (i) and (iii). In addition, the values of n just change the effective sample size, but hardly affect the ratio of effective sample size to the total samples.

Problem 3: Bayesian model selection in logistic regression.

The full specification of a *Logistic Regression* model can be summarized as follows:

$$y_i = Binomial(u_i)$$

$$ln\frac{u_i}{1-u_i} = \beta_{\gamma_0} + \beta_{\gamma_1}x_1 + \beta_{\gamma_2}x_2 + \beta_{\gamma_3}x_3 + \beta_{\gamma_4}x_4$$

Here, we define 16 *Logistic Regression* models with different variable selection from M_0 (intercept-only model) to M_{15} (intercept and X_1, X_2, X_3, X_4 included), and put a flat prior over over the model space consisting of the 16 models with $P(M_\gamma) = 1/16$. So, the log-posterior is:

$$logP(M_\gamma|Y) = log\frac{P(Y|M_\gamma)P(M_\gamma)}{P(Y)} = logP(Y|M_\gamma) + logP(M_\gamma) - logP(Y)$$

Here, we let M_γ be the model indexed by γ and γ is a vector of 4 switches that determine the variable selection of the model. As a result, γ varies from $(0,0,0,0)$ to $(1,1,1,1)$. β_γ denotes the vector of coefficients $\beta_{\gamma_0}, \beta_{\gamma_1}, \cdots$ in model M_γ and the dimension of β_γ ranges from 1 to 5 depending on how many variables are selected for the Logistic Regression model. Also, we put independent normal priors on β_γ, i.e., $\beta_\gamma \sim N(0, \sigma^2 I_d)$, where $d = 1 + \sum_j \gamma_j$ .In terms of the Random-Walk Metropolis-Hastings (RWMH) algorithm, we use the proposal pdf as follows:

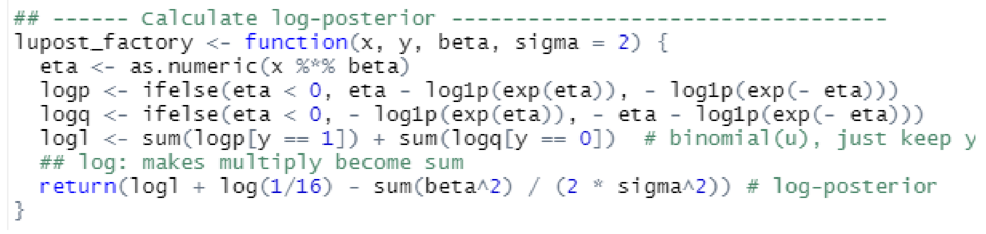
$$(\beta_\gamma^*) \sim N(\beta_\gamma^t, scale^2 I_d) \text{ where } d = 1 + \sum_j \gamma_j$$

3.1 Scenario 1: $\boldsymbol{\sigma = 2}$ and $\boldsymbol{\gamma = (1,1,1,1)}$: To estimate the model posterior probabilities, we first assume $\sigma = 2$ and $\gamma = (1,1,1,1)$ as an example to demonstrate the implementation of the log-posterior. Specifically, we conduct MCMC simulations to generate k (i.e., $k = 1$) chains, initialize it at the MLE for a given model, and then estimate the mean and variance of β_γ, as well as the values of the log-posterior at each of the MCMC samples.

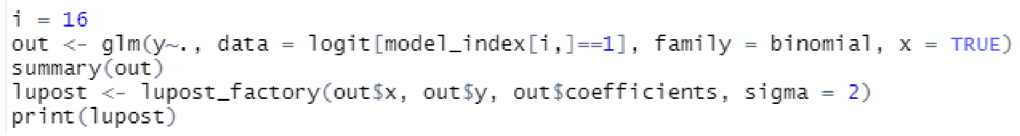
Step 1: Calculate the MLE and the log-posterior at the MLE for the M_{15} Logistic Regression model (intercept and X_1, X_2, X_3, X_4 included) as follows:

Coefficients: $\beta_{15} = [0.6328, 0.7390, 1.1137, 0.4781, 0.6944]$ for intercept and X_1, X_2, X_3, X_4

The *log-posterior probability* is -46.9689 at the MLE

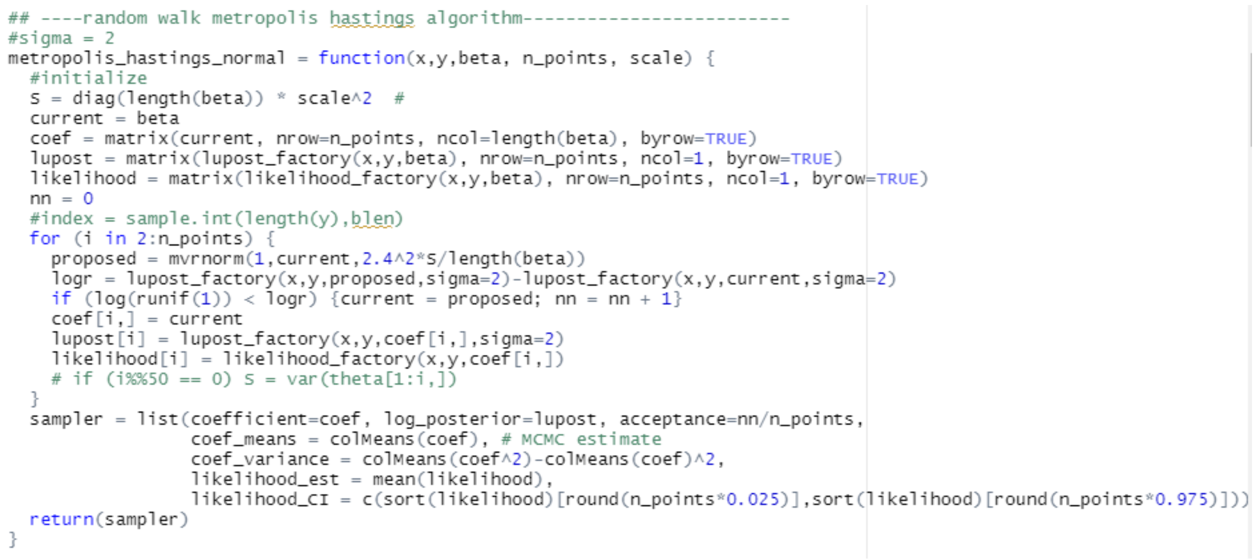


```
## ------ Calculate log-posterior ---------------------------------
lupost_factory <- function(x, y, beta, sigma = 2) {
  eta <- as.numeric(x %*% beta)
  logp <- ifelse(eta < 0, eta - log1p(exp(eta)), - log1p(exp(- eta)))
  logq <- ifelse(eta < 0, - log1p(exp(eta)), - eta - log1p(exp(- eta)))
  logl <- sum(logp[y == 1]) + sum(logq[y == 0])  # binomial(u), just keep y
  ## log: makes multiply become sum
  return(logl + log(1/16) - sum(beta^2) / (2 * sigma^2)) # log-posterior
}
```

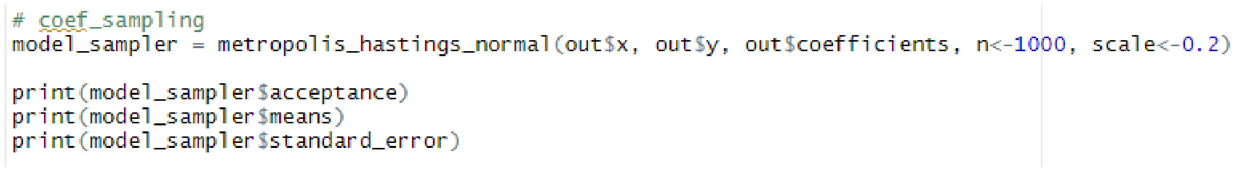


```
i = 16
out <- glm(y~., data = logit[model_index[i,]==1], family = binomial, x = TRUE)
summary(out)
lupost <- lupost_factory(out$x, out$y, out$coefficients, sigma = 2)
print(lupost)
```

Step 2: Use the Random Walk Metropolis-Hastings (RWMH) sampling method to perform the MCMC simulations with the MLE as an initial value and $n = 1000$. The proposal pdf g is $\beta_{15} \sim N(0, diag(scale^2))$ where we let the value of scale be 0.2 to guarantee a larger acceptance probability of β_{15}.



```
## ----random walk metropolis hastings algorithm------------------------
#sigma = 2
metropolis_hastings_normal = function(x,y,beta, n_points, scale) {
  #initialize
  S = diag(length(beta)) * scale^2  #
  current = beta
  coef = matrix(current, nrow=n_points, ncol=length(beta), byrow=TRUE)
  lupost = matrix(lupost_factory(x,y,beta), nrow=n_points, ncol=1, byrow=TRUE)
  likelihood = matrix(likelihood_factory(x,y,beta), nrow=n_points, ncol=1, byrow=TRUE)
  nn = 0
  #index = sample.int(length(y),blen)
  for (i in 2:n_points) {
    proposed = mvrnorm(1,current,2.4^2*S/length(beta))
    logr = lupost_factory(x,y,proposed,sigma=2)-lupost_factory(x,y,current,sigma=2)
    if (log(runif(1)) < logr) {current = proposed; nn = nn + 1}
    coef[i,] = current
    lupost[i] = lupost_factory(x,y,coef[i,],sigma=2)
    likelihood[i] = likelihood_factory(x,y,coef[i,])
    # if (i%%50 == 0) S = var(theta[1:i,])
  }
  sampler = list(coefficient=coef, log_posterior=lupost, acceptance=nn/n_points,
                 coef_means = colMeans(coef), # MCMC estimate
                 coef_variance = colMeans(coef^2)-colMeans(coef)^2,
                 likelihood_est = mean(likelihood),
                 likelihood_CI = c(sort(likelihood)[round(n_points*0.025)],sort(likelihood)[round(n_points*0.975)]))
  return(sampler)
}
```



```
# coef_sampling
model_sampler = metropolis_hastings_normal(out$x, out$y, out$coefficients, n<-1000, scale<-0.2)

print(model_sampler$acceptance)
print(model_sampler$means)
print(model_sampler$standard_error)
```

Step 3: Visualize the plots of the sequences $\beta_0, \beta_1, \beta_2, \beta_3, \beta_4$ produced by the RWMH sampler and the corresponding log-posterior probability for each iteration, as shown in Fig. 11 and 12. Also, some statistical indicators are calculated as follows:

Acceptance probability: 0.482 when scale = 0.2.

MCMC estimates of $\beta_0, \beta_1, \beta_2, \beta_3, \beta_4$: [0.6568,0.7466,1.1150,0.4803,0.8190]

MCMC variance estimates of $\beta_0, \beta_1, \beta_2, \beta_3, \beta_4$: [0.0863,0.1060,0.1291,0.0867,0.1143]

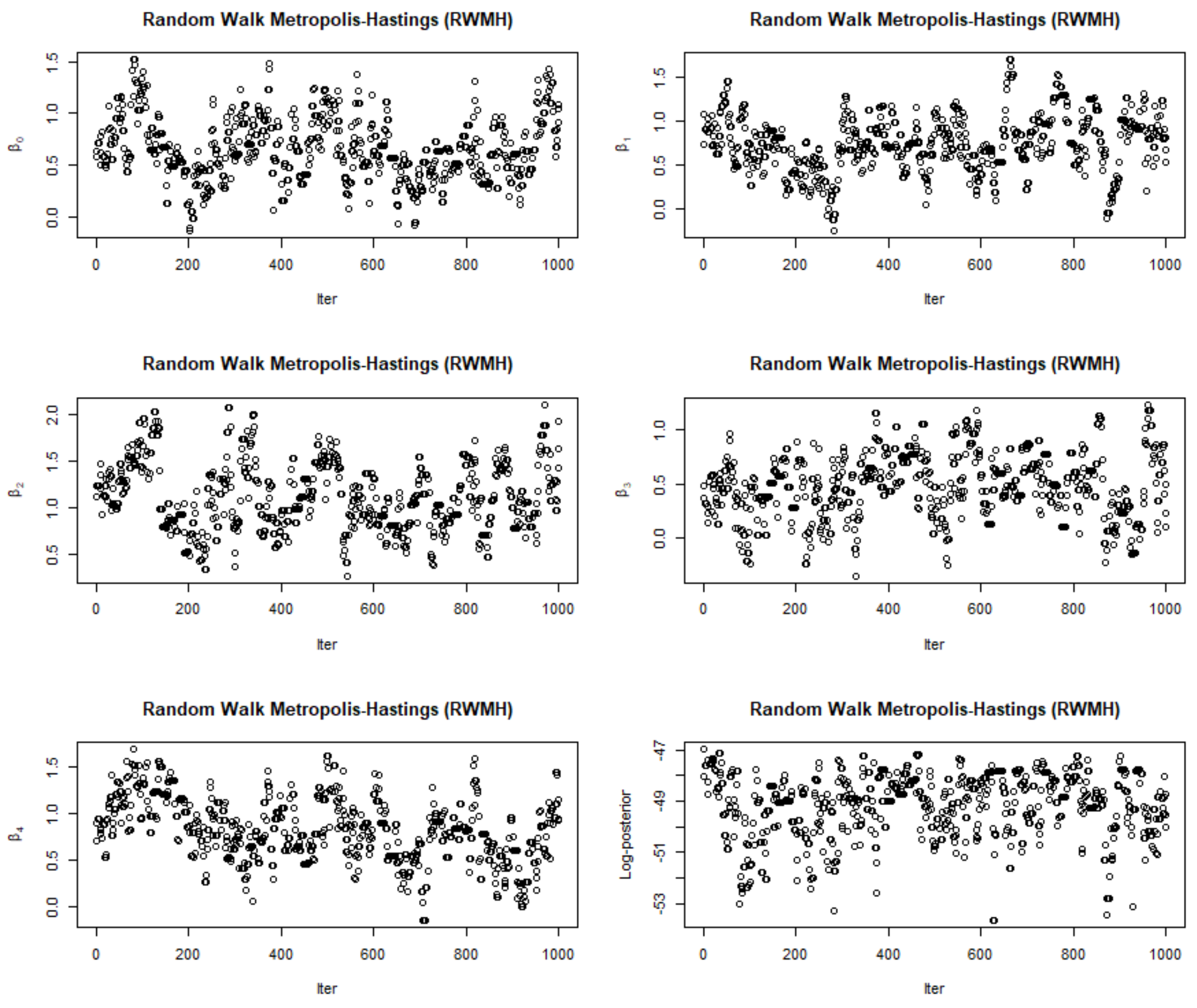


Fig. 11: Plots of the sequences $\beta_0, \beta_1, \beta_2, \beta_3, \beta_4$ produced by the RWMH sampler, as well as the corresponding log-posterior probability for each iteration.

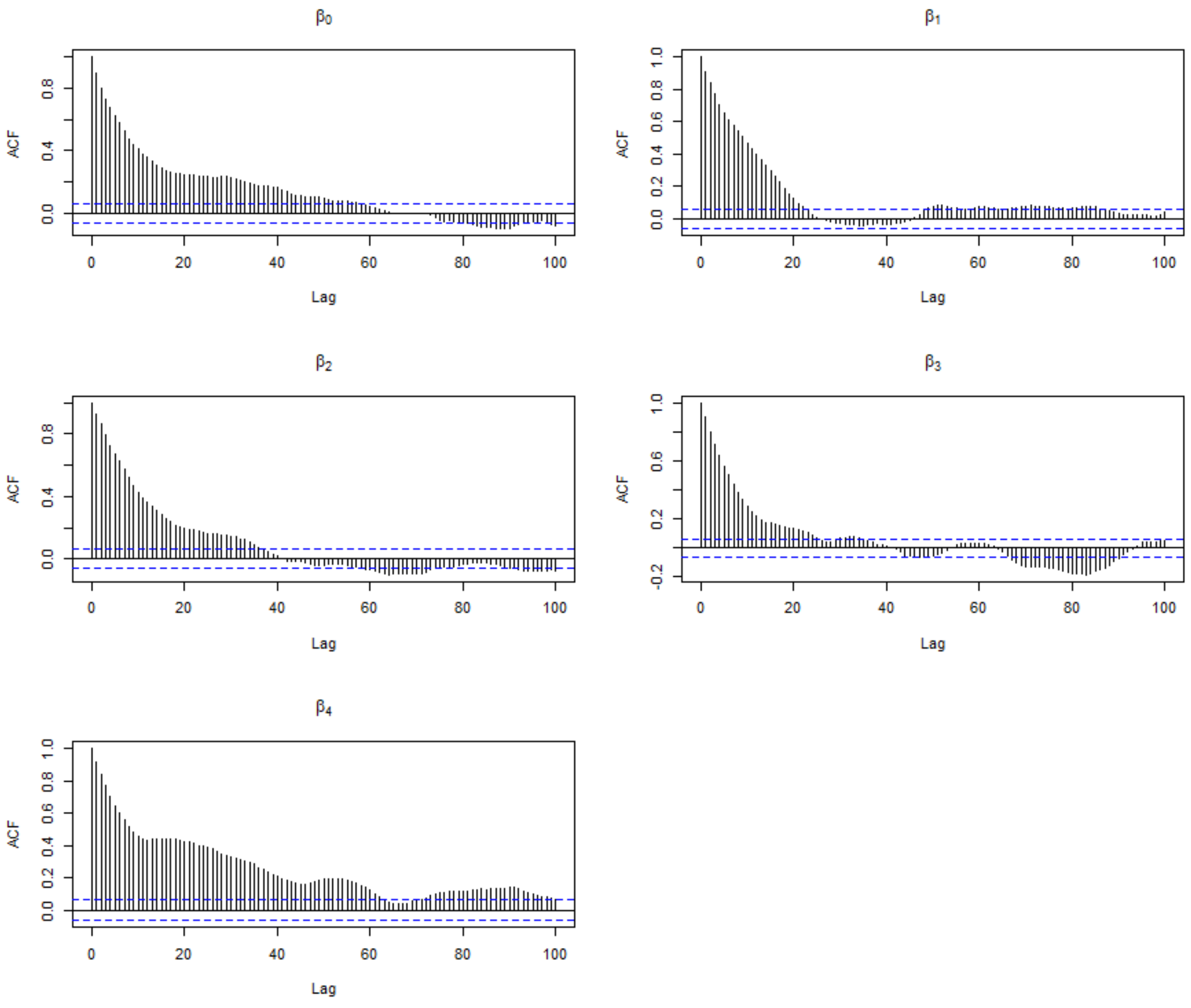


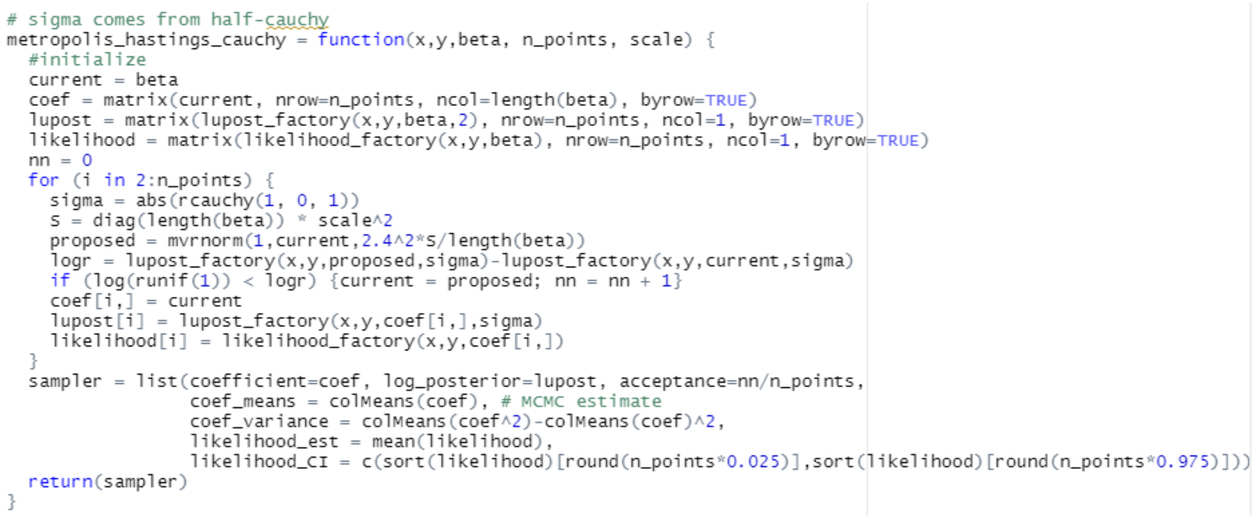
Fig. 12: Plots of autocorrelation of $\beta_0, \beta_1, \beta_2, \beta_3, \beta_4$ produced by the RWMH sampler

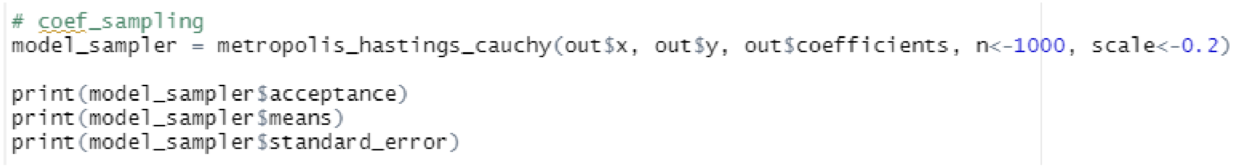
3.2 Scenario 2: $\boldsymbol{\sigma \sim Ca^+(0,1)}$ and $\boldsymbol{\gamma = (1,1,1,1)}$: Here, we assume $\sigma \sim Ca^+(0,1)$ and $\gamma = (1,1,1,1)$ to model the log-posterior probabilities and the other 15 scenarios for the *Logistic Regression* model allowing a general $\sigma \sim Ca^+(0,1)$ are implemented and shown in Problem 3.5. For this specific scenario 2, we conduct MCMC simulations to generate k (i.e., $k = 1$) chains, initialize it at the MLE for a given model, and then estimate the mean and variance of β_γ, as well as the values of the log-posterior at each of the MCMC samples.

Step 1: Calculate the MLE for the M_{15} Logistic Regression model (intercept and X_1, X_2, X_3, X_4 included) as follows:

Coefficients: $\beta_{15} = [0.6328, 0.7390, 1.1137, 0.4781, 0.6944]$ for intercept and X_1, X_2, X_3, X_4

Step 2: Use the Random Walk Metropolis-Hastings (RWMH) sampling method to perform the MCMC simulations with the MLE as an initial value and $n = 1000$. Also, we sample σ from $Ca^{+}(0,1)$ to calculate the log-posterior probabilities at each of the MCMC samples. The proposal pdf g is $\beta_{15} \sim N(0, diag(scale^2))$ where we let the value of scale be 0.2 to guarantee a larger acceptance probability of β_{15}.


```r
# sigma comes from half-cauchy
metropolis_hastings_cauchy = function(x,y,beta, n_points, scale) {
  #initialize
  current = beta
  coef = matrix(current, nrow=n_points, ncol=length(beta), byrow=TRUE)
  lupost = matrix(lupost_factory(x,y,beta,2), nrow=n_points, ncol=1, byrow=TRUE)
  likelihood = matrix(likelihood_factory(x,y,beta), nrow=n_points, ncol=1, byrow=TRUE)
  nn = 0
  for (i in 2:n_points) {
    sigma = abs(rcauchy(1, 0, 1))
    S = diag(length(beta)) * scale^2
    proposed = mvrnorm(1,current,2.4^2*S/length(beta))
    logr = lupost_factory(x,y,proposed,sigma)-lupost_factory(x,y,current,sigma)
    if (log(runif(1)) < logr) {current = proposed; nn = nn + 1}
    coef[i,] = current
    lupost[i] = lupost_factory(x,y,coef[i,],sigma)
    likelihood[i] = likelihood_factory(x,y,coef[i,])
  }
  sampler = list(coefficient=coef, log_posterior=lupost, acceptance=nn/n_points,
                 coef_means = colMeans(coef), # MCMC estimate
                 coef_variance = colMeans(coef^2)-colMeans(coef)^2,
                 likelihood_est = mean(likelihood),
                 likelihood_CI = c(sort(likelihood)[round(n_points*0.025)],sort(likelihood)[round(n_points*0.975)]))
  return(sampler)
}
```


```r
# coef_sampling
model_sampler = metropolis_hastings_cauchy(out$x, out$y, out$coefficients, n<-1000, scale<-0.2)

print(model_sampler$acceptance)
print(model_sampler$means)
print(model_sampler$standard_error)
```

Step 3: Visualize the plots of the sequences $\beta_0, \beta_1, \beta_2, \beta_3, \beta_4$ produced by the RWMH sampler and the corresponding log-posterior probability for each iteration, as shown in Fig. 13 and 14. Also, some statistical indicators are calculated as follows:

Acceptance probability: 0.453 when scale = 0.2.

MCMC estimates of $\beta_0, \beta_1, \beta_2, \beta_3, \beta_4$: [0.4622,0.7085,0.9415,0.4170,0.5764]

MCMC variance estimates of $\beta_0, \beta_1, \beta_2, \beta_3, \beta_4$: [0.0995,0.1088,0.1096,0.0999,0.1248]

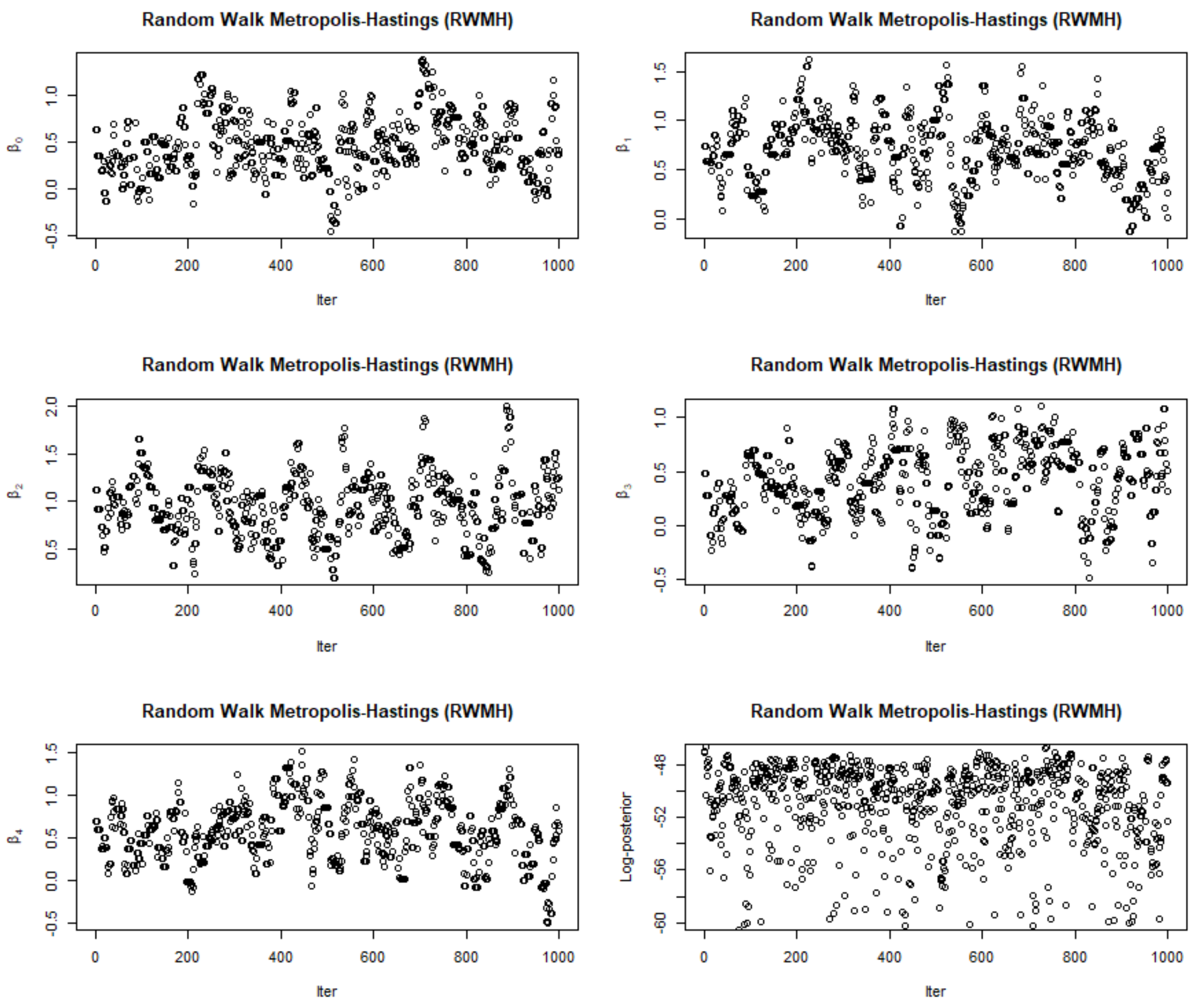


Fig. 13: Plots of the sequences $\beta_0, \beta_1, \beta_2, \beta_3, \beta_4$ produced by the RWMH sampler, as well as the corresponding log-posterior probability for each iteration.

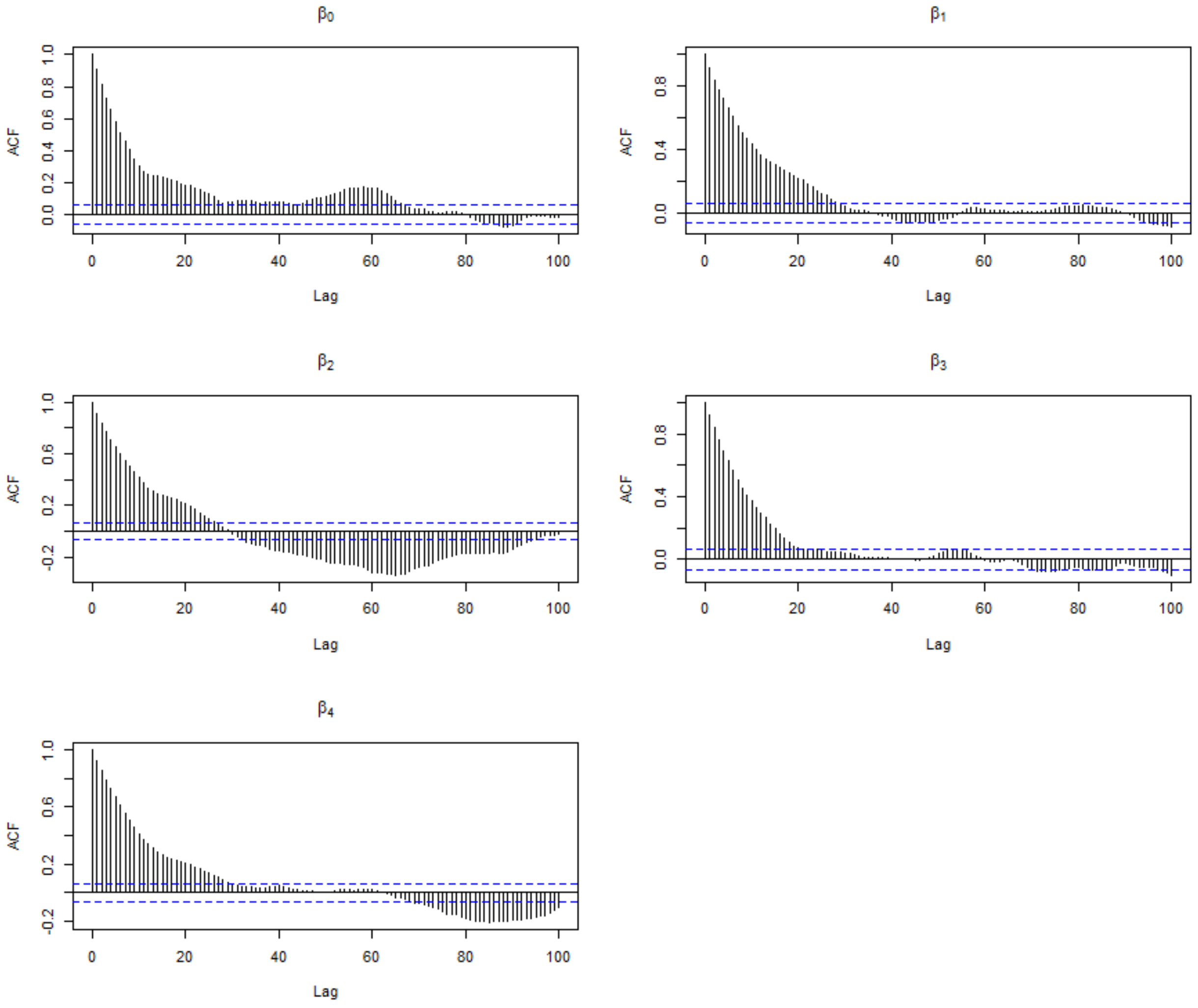


Fig. 14: Plots of autocorrelation of $\beta_0, \beta_1, \beta_2, \beta_3, \beta_4$ produced by the RWMH sampler

3.3 Reparameterize the log-posterior for implementation: Since, RWMH generally does not "like" constrained parameters, for instance, $\sigma \sim Ca^{+}(0,1)$ where σ is always postive, we replace the constrained σ with an unconstrained parameter $\tau = ln(\sigma)$ to reparameterize the log-posterior for each of the MCMC samples.

3.4 Tune RWMH: Here, we set different scales (corresponding to the covariance matrix of the proposal pdf) to check the effect of the scale on the acceptance rate of the RWMH algorithm. We vary the scale from 0.1, 0.2, 0.3, 0.4, 0.5, 1, 2, to 3 and return the corresponding acceptance rate, as shown in Table 3.

Table 3: The effect of scale of the proposal pdf on the acceptance rate of RWMH

scale	0.1	0.2	0.3	0.4	0.5	1	2	3
Acceptance rate	0.713	0.518	0.316	0.206	0.120	0.013	0.001	0

Note: The results are based on the scenario 1 where $\sigma = 2$.

As presented in Table 3, we can easily find that as the scale of the proposal pdf increases from 0.1 to 3, the acceptance rate gradually decreases until 0 when the scale is larger than 1. Therefore, across all 16 variable subsets (and two models, with known and unknown σ), we can let the scale be 0.2 to guarantee adequate acceptance rate while maintaining a large sampling range for β_γ, no exception for the full model (with $\gamma = (1,1,1,1)$ and unknown σ).

3.5 Model/Variable selection:

➢ *Random-Walk Metropolis-Hastings (RWMH) Algorithm for sampling from a proposal pdf g that is multivariate normal distribution*

Step 1: *RWMH Algorithm*. We have known the optimal random walk tuning parameter is $2.4^2 Var(X)/d$ where $Var(X)$ is the covariance matrix. Here, $Var(X) = scale^2$ and we set $scale = 0.1$. Therefore, the implementation of the RWMH algorithm is shown below:

(1) Start with $Var(X) = diag(d) * scale^2$ where $d = 1 + \sum_j \gamma_j$

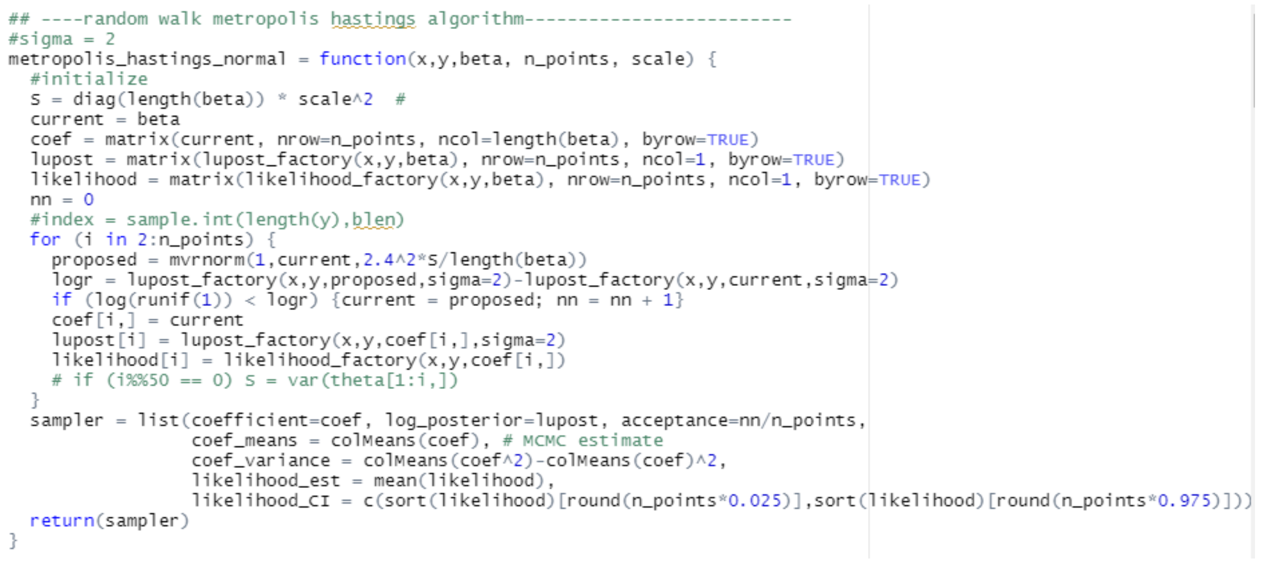
(2) Run $n = 1000$ iterations of the MCMC using $\frac{2.4^2 Var(X)}{d}$: $(\beta_\gamma^*) \sim N\left(\beta_\gamma^t, \frac{2.4^2 Var(X)}{d}\right)$

(3) Accept $(\beta_\gamma^{t+1}) = (\beta_\gamma^*)$ with probability $min\{1, r\}$ with $r = \frac{logP(M_\gamma: \beta_\gamma^*|Y)}{logP(M_\gamma: \beta_\gamma^t|Y)}$, otherwide $(\beta_\gamma^{t+1}) = (\beta_\gamma^t)$.

Step 2: Perform MCMC simulations for 16 models under two cases: $\sigma = 2$ and $\sigma \sim Ca^+(0,1)$.

Step 3: Present the statistical results of MCMC estimates for the 16 models in Table 4 and 5 with rows corresponding to different values of γ and columns referring to the MCMC estimates of mean and variance of β_γ, mean and credible interval (CI) of $P(y|M_\gamma)$, and the estimated Bayesian factor of M_γ vs M_0: $BF(M_\gamma: M_0)$.

(1) $\sigma = 2$



```
## ----random walk metropolis hastings algorithm-------------------------
#sigma = 2
metropolis_hastings_normal = function(x,y,beta, n_points, scale) {
  #initialize
  S = diag(length(beta)) * scale^2  #
  current = beta
  coef = matrix(current, nrow=n_points, ncol=length(beta), byrow=TRUE)
  lupost = matrix(lupost_factory(x,y,beta), nrow=n_points, ncol=1, byrow=TRUE)
  likelihood = matrix(likelihood_factory(x,y,beta), nrow=n_points, ncol=1, byrow=TRUE)
  nn = 0
  #index = sample.int(length(y),blen)
  for (i in 2:n_points) {
    proposed = mvrnorm(1,current,2.4^2*S/length(beta))
    logr = lupost_factory(x,y,proposed,sigma=2)-lupost_factory(x,y,current,sigma=2)
    if (log(runif(1)) < logr) {current = proposed; nn = nn + 1}
    coef[i,] = current
    lupost[i] = lupost_factory(x,y,coef[i,],sigma=2)
    likelihood[i] = likelihood_factory(x,y,coef[i,])
    # if (i%%50 == 0) S = var(theta[1:i,])
  }
  sampler = list(coefficient=coef, log_posterior=lupost, acceptance=nn/n_points,
                 coef_means = colMeans(coef), # MCMC estimate
                 coef_variance = colMeans(coef^2)-colMeans(coef)^2,
                 likelihood_est = mean(likelihood),
                 likelihood_CI = c(sort(likelihood)[round(n_points*0.025)],sort(likelihood)[round(n_points*0.975)]))
  return(sampler)
}
```

Table 4: MCMC estimates for the 16 Logistic Regression models

Model M_γ	Coefficients β_γ		Likelihood $lnP(y\|M_\gamma)$		$BF(M_\gamma : M_0)$
	Estimate	Variance	Estimate	CI	
(0,0,0,0)	[0.2154,0,0,0,0]	[0.0433,0,0,0,0]	-69.35	[-71.61,-68.81]	1
(0,0,0,1)	[0.2471,0,0,0,1.2202]	[0.0506,0,0,0,0.0960]	-58.68	[-60.82,-57.72]	4.3045e4
(0,0,1,0)	[0.2063,0,0,0.8265,0]	[0.0417,0,0,0.0576,0]	-63.41	[-65.66,-62.50]	3.7993e2
(0,0,1,1)	[0.2937,0,0,0.5803,1.1183]	[0.0611,0,0,0.0747,0.0908]	-56.99	[-59.36,-55.69]	2.3328e5
(0,1,0,0)	[0.4824,0,1.4769,0,0]	[0.0625,0,0.1107,0,0]	-53.18	[-56.28,-52.17]	1.0533e7
(0,1,0,1)	[0.5261,0,1.2458,0,0.9477]	[0.0760,0,0.1085,0,0.0981]	-49.25	[-52.02,-47.94]	5.3619e8
(0,1,1,0)	[0.5697,0,1.4649,0.8255,0]	[0.0674,0,0.1055,0.1458,0]	-51.05	[-54.56,-49.55]	8.8632e7
(0,1,1,1)	[0.6212,0,1.3246,0.6784, 0.9376]	[0.0593,0,0.1365,0.1157, 0.1116]	-47.97	[-51.20,-46.35]	1.9285e9
(1,0,0,0)	[0.3086,1.5316,0,0,0]	[0.0537,0.1507,0,0,0]	-55.17	[-58.41,-54.05]	1.4398e6
(1,0,0,1)	[0.3648,1.2513,0,0,0.8172]	[0.0538,0.1069,0,0,0.1232]	-52.21	[-55.87,-50.96]	2.7785e7
(1,0,1,0)	[0.3267,1.3549,0,0.4552,0]	[0.0656,0.1250,0,0.0861,0]	-54.22	[-56.96,-52.78]	3.7229e6
(1,0,1,1)	[0.3490,1.0380,0,0.3894, 0.8049]	[0.0559,0.1344,0,0.0903, 0.1629]	-52.03	[-55.46,-50.27]	3.3264e7
(1,1,0,0)	[0.5808,1.1003,1.1791,0,0]	[0.0717,0.0998,0.1162,0,0]	-47.81	[-50.59,-46.57]	2.2631e9
(1,1,0,1)	[0.6039,0.8048,1.1341,0, 0.7721]	[0.0886,0.1188,0.1187,0, 0.1512]	-46.69	[-49.43,-45.09]	6.9361e9
(1,1,1,0)	[0.6759,0.9033,1.3236, 0.5219,0]	[0.0887,0.1340,0.1465, 0.1095,0]	-47.51	[-51.80,-45.70]	3.0549e9
(1,1,1,1)	[0.7153,0.8049,1.1465, 0.4571,0.7817]	[0.1066,0.1159,0.1326, 0.1659,0.1392]	-46.39	[-50.22,-44.26]	9.3627e9

(2) $\sigma \sim Ca^{+}(0,1)$

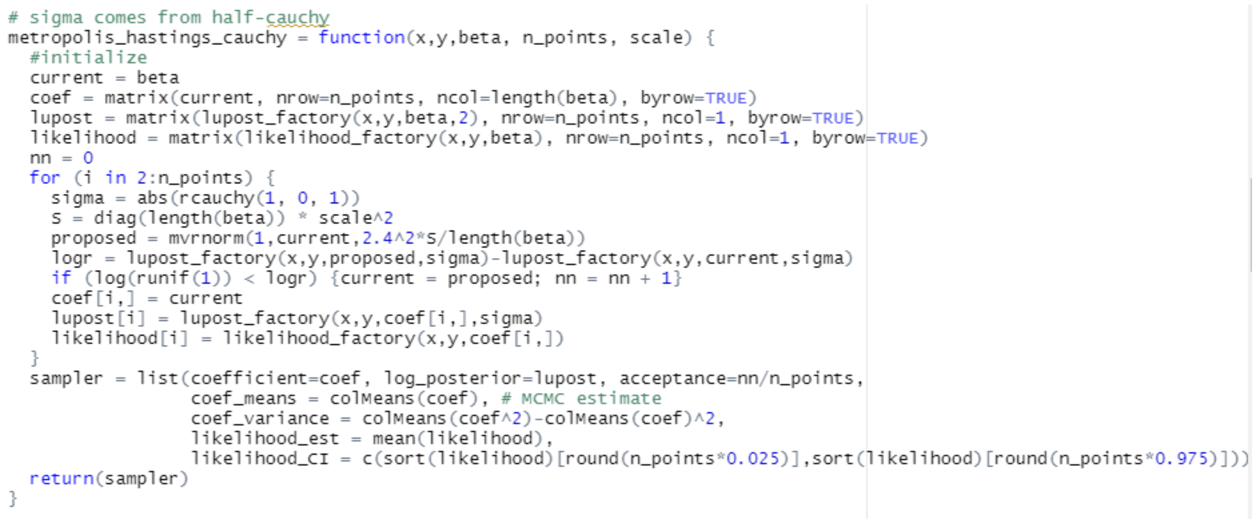

```
# sigma comes from half-cauchy
metropolis_hastings_cauchy = function(x,y,beta, n_points, scale) {
  #initialize
  current = beta
  coef = matrix(current, nrow=n_points, ncol=length(beta), byrow=TRUE)
  lupost = matrix(lupost_factory(x,y,beta,2), nrow=n_points, ncol=1, byrow=TRUE)
  likelihood = matrix(likelihood_factory(x,y,beta), nrow=n_points, ncol=1, byrow=TRUE)
  nn = 0
  for (i in 2:n_points) {
    sigma = abs(rcauchy(1, 0, 1))
    S = diag(length(beta)) * scale^2
    proposed = mvrnorm(1,current,2.4^2*S/length(beta))
    logr = lupost_factory(x,y,proposed,sigma)-lupost_factory(x,y,current,sigma)
    if (log(runif(1)) < logr) {current = proposed; nn = nn + 1}
    coef[i,] = current
    lupost[i] = lupost_factory(x,y,coef[i,],sigma)
    likelihood[i] = likelihood_factory(x,y,coef[i,])
  }
  sampler = list(coefficient=coef, log_posterior=lupost, acceptance=nn/n_points,
                 coef_means = colMeans(coef), # MCMC estimate
                 coef_variance = colMeans(coef^2)-colMeans(coef)^2,
                 likelihood_est = mean(likelihood),
                 likelihood_CI = c(sort(likelihood)[round(n_points*0.025)],sort(likelihood)[round(n_points*0.975)]))
  return(sampler)
}
```

Table 5: MCMC estimates for the 16 Logistic Regression models

Model M_γ	Coefficients β_γ		Likelihood $lnP(y\|M_\gamma)$		$BF(M_\gamma : M_0)$
	Estimate	Variance	Estimate	CI	
(0,0,0,0)	[0.1919,0,0,0,0]	[0.0461,0,0,0,0]	-69.38	[-71.59,-68.81]	1
(0,0,0,1)	[0.2006,0,0,0,0.9396]	[0.0501,0,0,0,0.1183]	-59.50	[-68.46,-57.73]	1.9536e4
(0,0,1,0)	[0.2048,0,0,0.6879,0]	[0.0482,0,0,0.0682,0]	-63.72	[-67.47,-62.53]	2.8715e2
(0,0,1,1)	[0.2422,0,0,0.4260,0.9228]	[0.0606,0,0,0.0820,0.1433]	-57.92	[-64.09,-55.65]	9.4845e4
(0,1,0,0)	[0.3403,0,1.0826,0,0]	[0.0660,0,0.1424,0,0]	-54.51	[-62.01,-52.18]	2.8705e6
(0,1,0,1)	[0.4388,0,1.0395,0,0.8299]	[0.0934,0,0.1322,0,0.1106]	-50.16	[-56.12,-47.95]	2.2240e8
(0,1,1,0)	[0.3853,0,1.1193,0.5458,0]	[0.0711,0,0.1128,0.1345,0]	-52.02	[-58.75,-49.68]	3.4622e7
(0,1,1,1)	[0.4334,0,1.0024,0.4453, 0.7046]	[0.0946,0,0.1421,0.1202, 0.1391]	-49.50	[-55.74,-46.53]	4.3030e8
(1,0,0,0)	[0.2665,1.2798,0,0,0]	[0.0774,0.1368,0,0,0]	-55.65	[-59.74,-54.06]	9.1804e5
(1,0,0,1)	[0.2752,0.9845,0,0,0.6820]	[0.0746,0.1015,0,0,0.0963]	-52.85	[-57.61,-51.00]	1.5097e7
(1,0,1,0)	[0.2607,1.0528,0,0.3963,0]	[0.0794,0.1210,0,0.0887,0]	-55.08	[-61.28,-52.76]	1.6223e6
(1,0,1,1)	[0.3212,0.9640,0,0.3635, 0.7200]	[0.0525,0.1324,0,0.0887, 0.1061]	-52.06	[-55.84,-50.18]	3.3264e7
(1,1,0,0)	[0.4539,0.9080,0.9567,0,0]	[0.1049,0.1435,0.1514,0,0]	-49.12	[-53.90,-46.67]	6.2923e8
(1,1,0,1)	[0.4635,0.7260,0.9536,0, 0.6514]	[0.1108,0.1464,0.1454,0, 0.1265]	-47.57	[-52.30,-45.24]	2.9646e9
(1,1,1,0)	[0.3611,0.7434,0.8036, 0.4515,0]	[0.0834,0.1462,0.1025, 0.0956,0]	-48.85	[-55.93,-45.85]	8.2426e8
(1,1,1,1)	[0.4236,0.7067,0.9181, 0.4272,0.5393]	[0.0800,0.1199,0.1127, 0.0802,0.1192]	-46.66	[-51.98,-44.28]	7.3650e9

- *RWMH Algorithm for importance sampling from from a proposal pdf g that is multivariate normal with mean equal to the posterior mode and covariance matrix taken to be the sample covariance matrix from the MCMC samples from the posterior (after tuning)*

Step 1: *RWMH for importance sampling*. We have known the optimal random walk tuning parameter is $2.4^2 Var(X)/d$ where $Var(X)$ is the unknown covariance matrix. We can estimate

$Var(X)$ using the sample coveraiance matrix of draws. Therefore, the implementation of the RWMH algorithm for importance sampling is shown below:

(1) Start with $Var(X) = diag(d)$ where $d = 1 + \sum_j \gamma_j$

(2) Run 50 iterations of the MCMC using $\frac{2.4^2 Var(X)}{d}$: $(\beta_\gamma^*) \sim N\left(\beta_\gamma^t, \frac{2.4^2 Var(X)}{d}\right)$

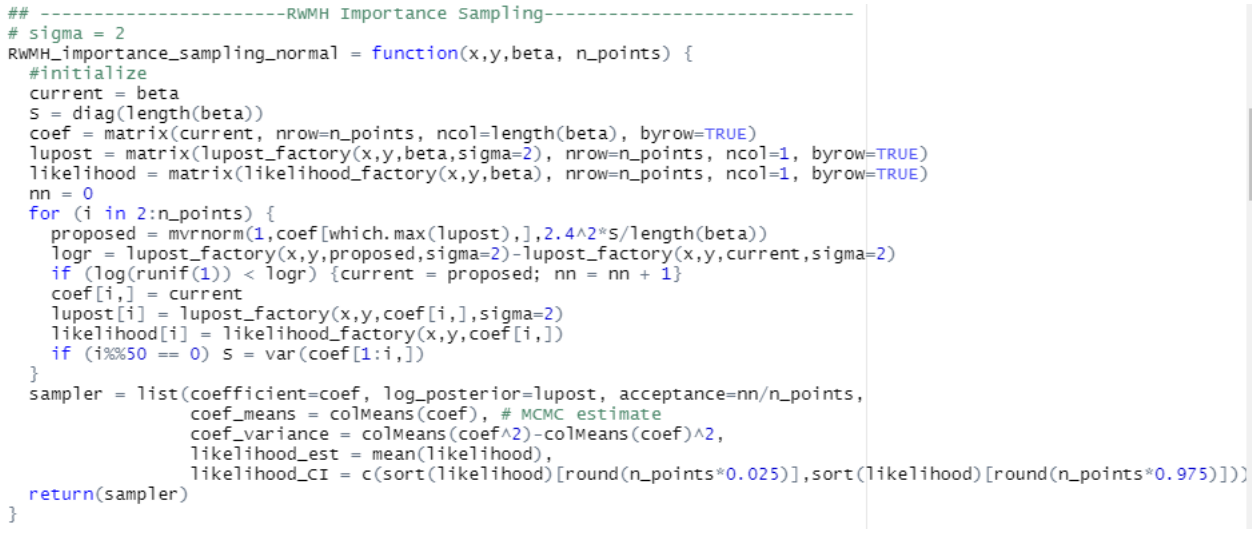
(3) Set $Var(X)$ to the sample covariance matrix of all previous draws

(4) Accept $(\beta_\gamma^{t+1}) = (\beta_\gamma^*)$ with probability $min\{1, r\}$ with $r = \frac{logP(M_\gamma{:}\beta_\gamma^*|Y)}{logP(M_\gamma{:}\beta_\gamma^t|Y)}$, otherwide $(\beta_\gamma^{t+1}) = (\beta_\gamma^t)$.

Step 2: Perform MCMC simulations for 16 models under two cases: $\sigma = 2$ and $\sigma \sim Ca^+(0,1)$.

Step 3: Present the statistical results of MCMC estimates for the 16 models in Table 6 and 7 with rows corresponding to different values of γ and columns referring to the MCMC estimates of mean and variance of β_γ, mean and credible interval (CI) of $P(y|M_\gamma)$, and the estimated Bayesian factor of M_γ vs M_0: $BF(M_\gamma : M_0)$.

(1) $\sigma = 2$



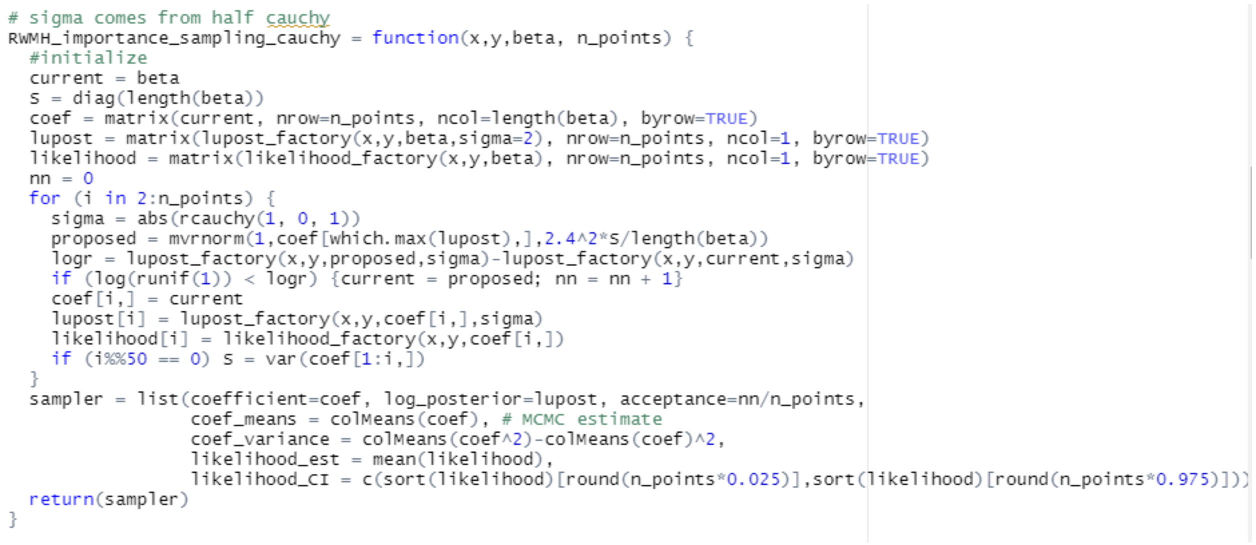
```
## -----------------------RWMH Importance Sampling----------------------------
# sigma = 2
RWMH_importance_sampling_normal = function(x,y,beta, n_points) {
  #initialize
  current = beta
  S = diag(length(beta))
  coef = matrix(current, nrow=n_points, ncol=length(beta), byrow=TRUE)
  lupost = matrix(lupost_factory(x,y,beta,sigma=2), nrow=n_points, ncol=1, byrow=TRUE)
  likelihood = matrix(likelihood_factory(x,y,beta), nrow=n_points, ncol=1, byrow=TRUE)
  nn = 0
  for (i in 2:n_points) {
    proposed = mvrnorm(1,coef[which.max(lupost),],2.4^2*S/length(beta))
    logr = lupost_factory(x,y,proposed,sigma=2)-lupost_factory(x,y,current,sigma=2)
    if (log(runif(1)) < logr) {current = proposed; nn = nn + 1}
    coef[i,] = current
    lupost[i] = lupost_factory(x,y,coef[i,],sigma=2)
    likelihood[i] = likelihood_factory(x,y,coef[i,])
    if (i%%50 == 0) S = var(coef[1:i,])
  }
  sampler = list(coefficient=coef, log_posterior=lupost, acceptance=nn/n_points,
                 coef_means = colMeans(coef), # MCMC estimate
                 coef_variance = colMeans(coef^2)-colMeans(coef)^2,
                 likelihood_est = mean(likelihood),
                 likelihood_CI = c(sort(likelihood)[round(n_points*0.025)],sort(likelihood)[round(n_points*0.975)]))
  return(sampler)
}
```

Table 6: MCMC estimates for the 16 Logistic Regression models

Model M_γ	Coefficients β_γ		Likelihood $lnP(y\|M_\gamma)$		$BF(M_\gamma : M_0)$
	Estimate	Variance	Estimate	CI	
(0,0,0,0)	[0.1892,0,0,0,0]	[0.0323,0,0,0,0]	-69.21	[-70.91,-68.81]	1
(0,0,0,1)	[0.2555,0,0,0,1.2122]	[0,0,0,0,0]	-57.70	[-57.70,-57.70]	9.9708e4
(0,0,1,0)	[0.2270,0,0,0.7968,0]	[0.0189,0,0,0.0153,0]	-62.80	[-64.02,-62.49]	6.0789e2
(0,0,1,1)	[0.2716,0,0,0.5704,1.0762]	[0,0,0,0,0]	-55.56	[-55.56,-55.56]	8.4746e5
(0,1,0,0)	[0.5039,0,1.4490,0,0]	[0.0387,0,0.0769,0,0]	-52.79	[-54.58,-52.15]	1.3524e7
(0,1,0,1)	[0.5531,0,1.2797,0,0.9368]	[0,0,0,0,0]	-47.86	[-47.86,-47.86]	1.8715e9
(0,1,1,0)	[0.5464,0,1.3983,0.6846,0]	[0,0,0,0,0]	-49.48	[-49.48,-49.48]	3.7037e8

(0,1,1,1)	[0.5990,0,1.2936,0.6042, 0.8794]	[0,0,0,0,0]	-46.17	[-46.17,-46.17]	1.0143e10
(1,0,0,0)	[0.3152,1.4634,0,0,0]	[0,0,0,0,0]	-54.03	[-54.03,-54.03]	3.9137e6
(1,0,0,1)	[0.3635,1.1525,0,0,0.8294]	[0,0,0,0,0]	-50.85	[-50.85,-50.85]	9.4112e7
(1,0,1,0)	[0.3119,1.3016,0,0.4541,0]	[0,0,0,0,0]	-52.65	[-52.65,-52.65]	1.5557e7
(1,0,1,1)	[0.3652,1.0603,0,0.4021, 0.7744]	[0,0,0,0,0]	-50.02	[-50.02,-50.02]	2.1583e8
(1,1,0,0)	[0.5582,1.0823,1.1421,0,0]	[0.0003,0.0674,0.0208,0,0]	-46.83	[-47.98,-46.50]	5.2422e9
(1,1,0,1)	[0.6043,0.8305,1.0877,0, 0.7049]	[0,0,0,0,0]	-44.79	[-44.79,-44.79]	4.0315e10
(1,1,1,0)	[0.5784,0.9363,1.1591, 0.4673,0]	[0,0,0,0,0]	-45.42	[-45.42,-45.42]	2.1472e10
(1,1,1,1)	[0.6330,0.7453,1.1005, 0.4734,0.6975]	[0.0199,0.0200,0.0855, 0.0295,0.0501]	-44.58	[-46.76,-43.87]	4.9736e10

(2) $\sigma \sim Ca^{+}(0,1)$



```
# sigma comes from half cauchy
RWMH_importance_sampling_cauchy = function(x,y,beta, n_points) {
  #initialize
  current = beta
  S = diag(length(beta))
  coef = matrix(current, nrow=n_points, ncol=length(beta), byrow=TRUE)
  lupost = matrix(lupost_factory(x,y,beta,sigma=2), nrow=n_points, ncol=1, byrow=TRUE)
  likelihood = matrix(likelihood_factory(x,y,beta), nrow=n_points, ncol=1, byrow=TRUE)
  nn = 0
  for (i in 2:n_points) {
    sigma = abs(rcauchy(1, 0, 1))
    proposed = mvrnorm(1,coef[which.max(lupost),],2.4^2*S/length(beta))
    logr = lupost_factory(x,y,proposed,sigma)-lupost_factory(x,y,current,sigma)
    if (log(runif(1)) < logr) {current = proposed; nn = nn + 1}
    coef[i,] = current
    lupost[i] = lupost_factory(x,y,coef[i,],sigma)
    likelihood[i] = likelihood_factory(x,y,coef[i,])
    if (i%%50 == 0) S = var(coef[1:i,])
  }
  sampler = list(coefficient=coef, log_posterior=lupost, acceptance=nn/n_points,
                 coef_means = colMeans(coef), # MCMC estimate
                 coef_variance = colMeans(coef^2)-colMeans(coef)^2,
                 likelihood_est = mean(likelihood),
                 likelihood_CI = c(sort(likelihood)[round(n_points*0.025)],sort(likelihood)[round(n_points*0.975)]))
  return(sampler)
}
```

Table 7: MCMC estimates for the 16 Logistic Regression models

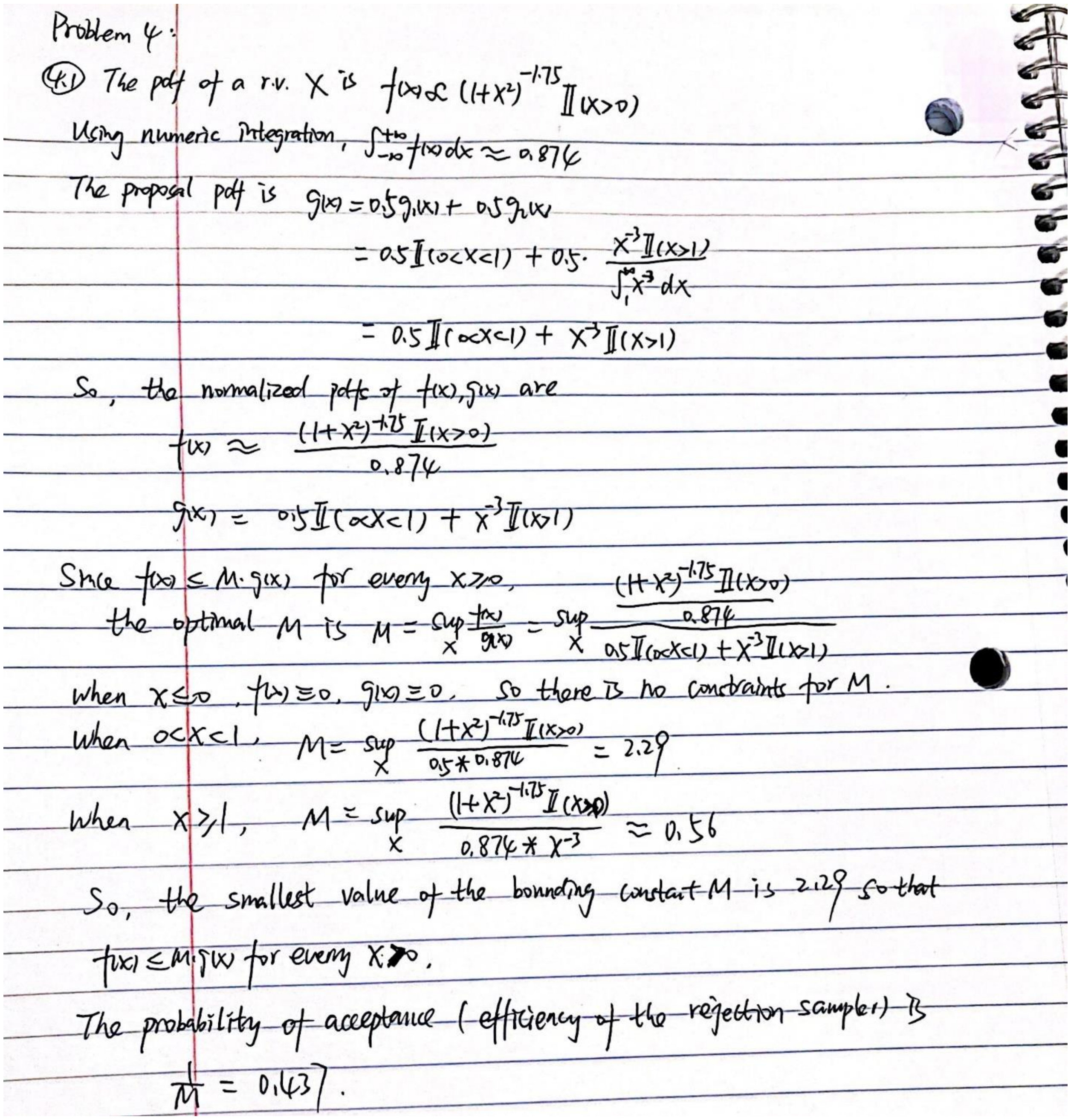
Model M_γ	Coefficients β_γ		Likelihood $lnP(y\|M_\gamma)$		$BF(M_\gamma : M_0)$
	Estimate	Variance	Estimate	CI	
(0,0,0,0)	[0.1792,0,0,0,0]	[0.0340,0,0,0,0]	-69.24	[-71.10,-68.81]	1
(0,0,0,1)	[0.2555,0,0,0,1.2122]	[0,0,0,0,0]	-57.70	[-57.70,-57.70]	1.0274e5
(0,0,1,0)	[0.1927,0,0,0.7130,0]	[0.0288,0,0,0.0595,0]	-63.39	[-66.29,-62.50]	3.4723e2
(0,0,1,1)	[0.2716,0,0,0.5704,1.0762]	[0,0,0,0,0]	-55.56	[-55.56,-55.56]	8.7327e5
(0,1,0,0)	[0.5087,0,1.4370,0,0]	[0,0,0,0,0]	-52.14	[-52.14,-52.14]	2.6695e7
(0,1,0,1)	[0.5143,0,1.2557,0,0.9479]	[0.0176,0,0.0067,0,0.0014]	-48.00	[-48.62,-47.86]	1.6765e9
(0,1,1,0)	[0.4547,0,1.2433,0.6206,0]	[0.0836,0,0.1288,0.0921,0]	-51.38	[-60.16,-49.49]	5.7082e7
(0,1,1,1)	[0.5231,0,1.2564,0.6084, 0.7646]	[0.0339,0,0.0271,0.0332, 0.0750]	-46.96	[-49.01,-46.18]	4.7433e9
(1,0,0,0)	[0.2708,1.3329,0,0,0]	[0.0712,0.1787,0,0,0]	-55.74	[-61.64,-54.06]	7.2942e5
(1,0,0,1)	[0.3230,1.0124,0,0,0.7485]	[0.0567,0.0776,0,0,0.1033]	-52.44	[-59.29,-50.97]	1.9776e7
(1,0,1,0)	[0.2837,1.2129,0,0.4350,0]	[0.0553,0.0573,0,0.0572,0]	-53.84	[-57.48,-52.65]	4.8768e6
(1,0,1,1)	[0.3652,1.0603,0,0.4021, 0.7744]	[0,0,0,0,0]	-50.02	[-50.02,-50.02]	2.2240e8
(1,1,0,0)	[0.4958,0.9898,1.0612,0,0]	[0.1127,0.1134,0.1460,0,0]	-48.63	[-56.57,-46.50]	8.9291e8

(1,1,0,1)	[0.6043,0.8305,1.0877,0, 0.7049]	[0,0,0,0,0]	-44.79	[-44.79,-44.79]	4.1543e10
(1,1,1,0)	[0.5784,0.9363,1.1591, 0.4673,0]	[0,0,0,0,0]	-45.42	[-45.42,-45.42]	2.2126e10
(1,1,1,1)	[0.6328,0.7390,1.1137, 0.4781,0.6944]	[0,0,0,0,0]	-43.83	[-43.83,-43.83]	1.0850e11

In summary, based on the results of Table 4-7, we can eaasily find that the full model (intercept and X_1, X_2, X_3, X_4 included) performs the best, as opposed to other model/variable selection, with the largest likelihood $lnP\left(y|M_{\gamma=(1,1,1,1)}\right)$ and the highest Bayesian factor $BF\left(M_{\gamma=(1,1,1,1)}: M_0\right)$. In addition, RWMH for importance sampling from a proposal pdf that is multivariate normal with mean equal to the posterior mode and covariance matrix taken to be the sample covariance matrix performs better than the classic RWMH algorithm. In general, whether the classic RWMH algorithm or the RWMH for importance sampling, the scenario of $\sigma = 2$ performs better than the scenario of $\sigma \sim Ca^+(0,1)$, in terms of the data likelihood and Bayesian factor. For a single variable, we find that the variable X_2 is more important than the variables X_1, X_4, X_3 in predicting Y. In terms of dual variables, the combination of the variables X_1, X_2 performs better than others. For the combination of three variables, the variables X_1, X_2, X_4 perform the best. However, all these combinations of different variables are worse than the full model when predicting the variable Y.

Problem 4: Rejection sampling and importance sampling

4.1 The smallest value of the bounding constant M is 2.29 so that $f(x) \le M \cdot g(x)$ for every $x > 0$. The efficiency of the rejection sampler is $\frac{1}{M} = 0.437$. Based on the simulations, we calculated the acceptance rate of the rejection sampler is 0.48.



Problem 4:

(4.1) The pdf of a r.v. X is $f(x) \propto (1+x^2)^{-1.75}\, \mathbb{I}(x>0)$

Using numeric integration, $\int_{-\infty}^{+\infty} f(x)\,dx \approx 0.874$

The proposal pdf is $g(x) = 0.5 g_1(x) + 0.5 g_2(x)$

$$= 0.5\,\mathbb{I}(0<x<1) + 0.5 \cdot \frac{x^{-3}\,\mathbb{I}(x>1)}{\int_1^{\infty} x^{-3}\,dx}$$

$$= 0.5\,\mathbb{I}(0<x<1) + x^{-3}\,\mathbb{I}(x>1)$$

So, the normalized pdfs of $f(x), g(x)$ are

$$f(x) \approx \frac{(1+x^2)^{-1.75}\,\mathbb{I}(x>0)}{0.874}$$

$$g(x) = 0.5\,\mathbb{I}(0<x<1) + x^{-3}\,\mathbb{I}(x>1)$$

Since $f(x) \le M \cdot g(x)$ for every $x \geqslant 0$,
the optimal M is $M = \sup_x \frac{f(x)}{g(x)} = \sup_x \frac{\frac{(1+x^2)^{-1.75}\,\mathbb{I}(x>0)}{0.874}}{0.5\,\mathbb{I}(0<x<1) + x^{-3}\,\mathbb{I}(x>1)}$

when $x \le 0$, $f(x) \equiv 0$, $g(x) \equiv 0$. So there is no constraints for M.

when $0<x<1$, $M = \sup_x \frac{(1+x^2)^{-1.75}\,\mathbb{I}(x>0)}{0.5 * 0.874} = 2.29$

when $x \geqslant 1$, $M = \sup_x \frac{(1+x^2)^{-1.75}\,\mathbb{I}(x>0)}{0.874 * x^{-3}} \approx 0.56$

So, the smallest value of the bounding constant M is 2.29 so that $f(x) \le M \cdot g(x)$ for every $x > 0$.

The probability of acceptance (efficiency of the rejection sampler) is

$$\frac{1}{M} = 0.437.$$

Then we use the rejection sampling method with the proposal $g(x)$ to simulate $X \sim f(x)$ and plot the samples in Fig. 15.

```
sampler_1 = function(n_points, w=0.5) {
  sample = matrix(0, nrow=n_points, ncol=1, byrow=TRUE)
  for (i in 1:n_points) {
    if (runif(1)<w) {sample[i] = runif(1)}
    else {sample[i] = sqrt(1/(1-runif(1)))}
  }
  return(sample)
}

# rejection sampling
set.seed(1)

M = 2/I_f$value
n = 10^2
d = data.frame(x = sampler_1(n),
               u = runif(n)) %>%
  mutate(u_scaled = u*M*g(x),
         accept   = u_scaled < f(x)/I_f$value)

gg = ggplot(d, aes(x=x,y=u_scaled,col=accept)) +
  geom_point()

clrs = unique(ggplot_build(gg)$data[[1]]$colour)

gg + stat_function(fun=function(x) M*g(x), col=clrs[2]) +
  stat_function(fun=function(x) f(x)/I_f$value, col=clrs[1]) +
  labs(x="sample",y=expression(paste("u M g(",x,")"))) +
  theme_bw()

cat("Observed acceptance rate was",mean(d$accept))
```

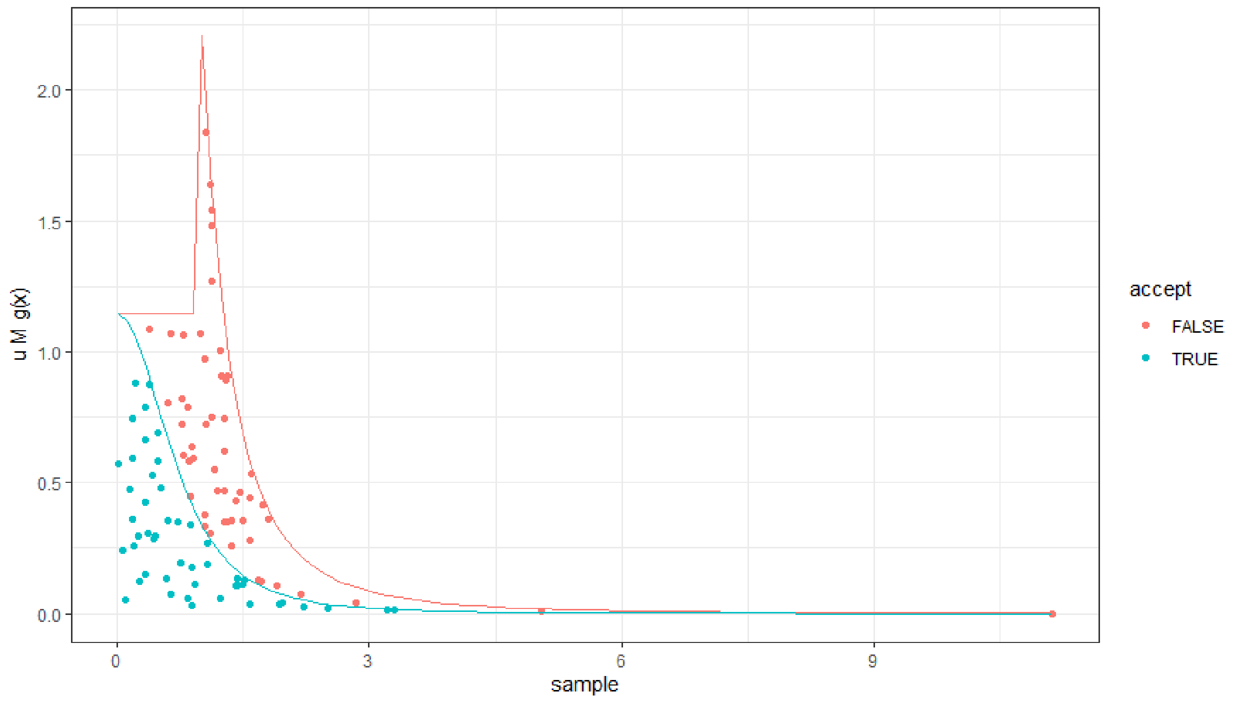
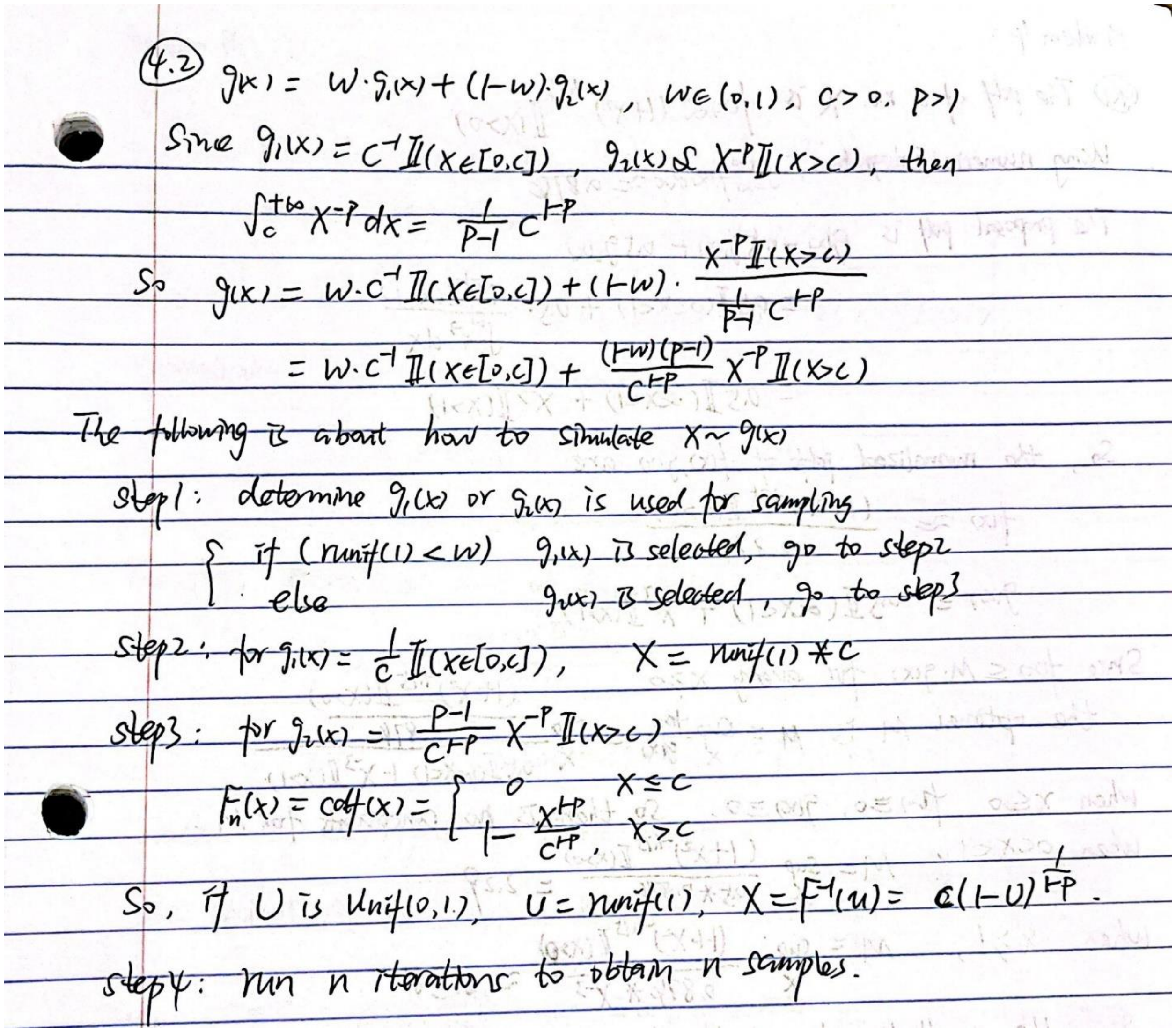


Fig. 15: Plots of samples derived from the rejection samplers

4.2



(4.2) $g(x) = w\cdot g_1(x) + (1-w)\cdot g_2(x)$, $w\in(0,1)$, $c>0$, $p>1$

Since $g_1(x) = c^{-1}\mathbb{I}(x\in[0,c])$, $g_2(x) \propto x^{-p}\mathbb{I}(x>c)$, then

$$\int_c^{+\infty} x^{-p}\,dx = \frac{1}{p-1}c^{1-p}$$

So $g(x) = w\cdot c^{-1}\mathbb{I}(x\in[0,c]) + (1-w)\cdot \dfrac{x^{-p}\mathbb{I}(x>c)}{\frac{1}{p-1}c^{1-p}}$

$= w\cdot c^{-1}\mathbb{I}(x\in[0,c]) + \dfrac{(1-w)(p-1)}{c^{1-p}}x^{-p}\mathbb{I}(x>c)$

The following is about how to simulate $X\sim g(x)$

step1: determine $g_1(x)$ or $g_2(x)$ is used for sampling

$\begin{cases} \text{if (runif(1) < w)} & g_1(x) \text{ is selected, go to step2} \\ \text{else} & g_2(x) \text{ is selected, go to step3} \end{cases}$

step2: for $g_1(x) = \frac{1}{c}\mathbb{I}(x\in[0,c])$, $X = \text{runif}(1) * c$

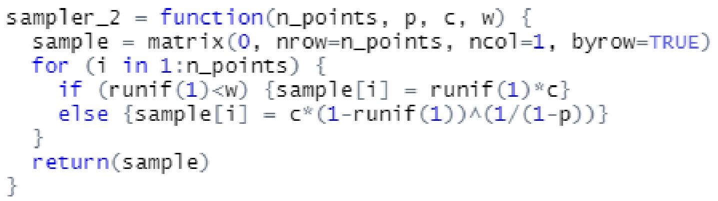
step3: for $g_2(x) = \dfrac{p-1}{c^{1-p}}x^{-p}\mathbb{I}(x>c)$

$$F_n(x) = \text{cdf}(x) = \begin{cases} 0 & x\le c \\ 1-\dfrac{x^{1-p}}{c^{1-p}}, & x>c \end{cases}$$

So, if U is Unif(0,1), $U = \text{runif}(1)$, $X = F^{-1}(u) = c(1-U)^{\frac{1}{1-p}}$.

step4: run n iterations to obtain n samples.

Specifically, the implementation of sampling from $X\sim g$ is shown below:



```r
sampler_2 = function(n_points, p, c, w) {
  sample = matrix(0, nrow=n_points, ncol=1, byrow=TRUE)
  for (i in 1:n_points) {
    if (runif(1)<w) {sample[i] = runif(1)*c}
    else {sample[i] = c*(1-runif(1))^(1/(1-p))}
  }
  return(sample)
}
```

To check the reliability of the sampler, we first assume $w = 0.5, p = 3, c = 1, n = 100$ and plot the cdf and ecdf based on the samples from of $X\sim g$ in Fig. 16. Also, we plot the absolute error between the cdf and ecdf in Fig. 17. The two figures both suggest that the difference between the cdf and ecdf is tiny.

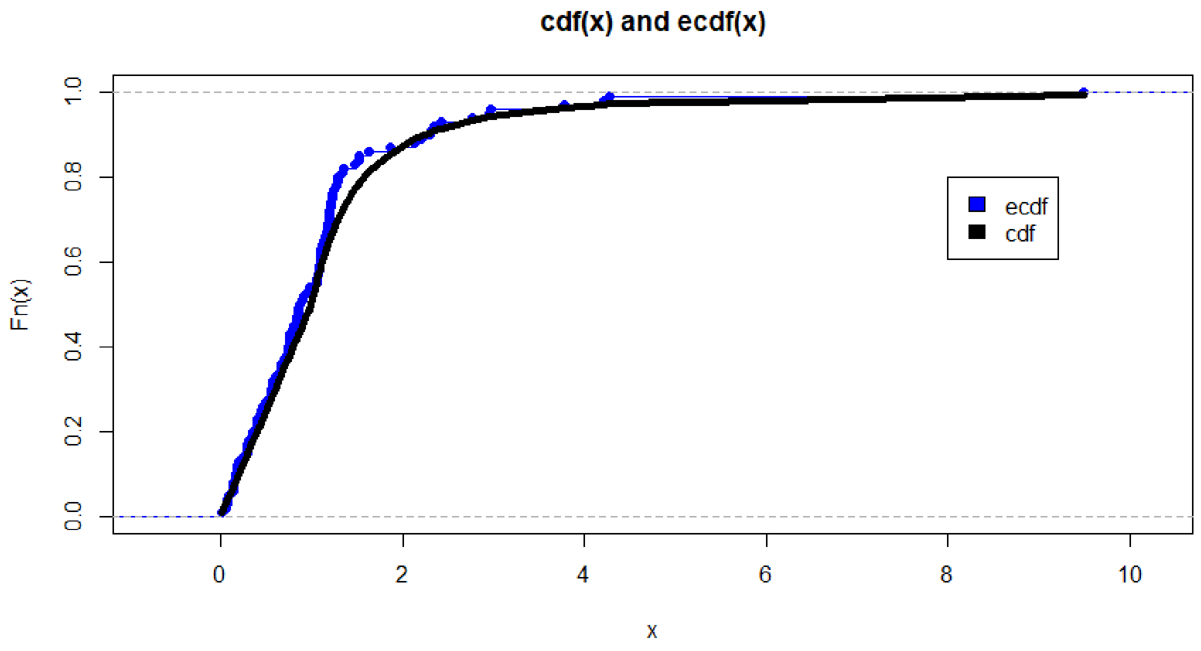


Fig. 16: Plots of the cdf and ecdf based on the simulations of $X \sim g$

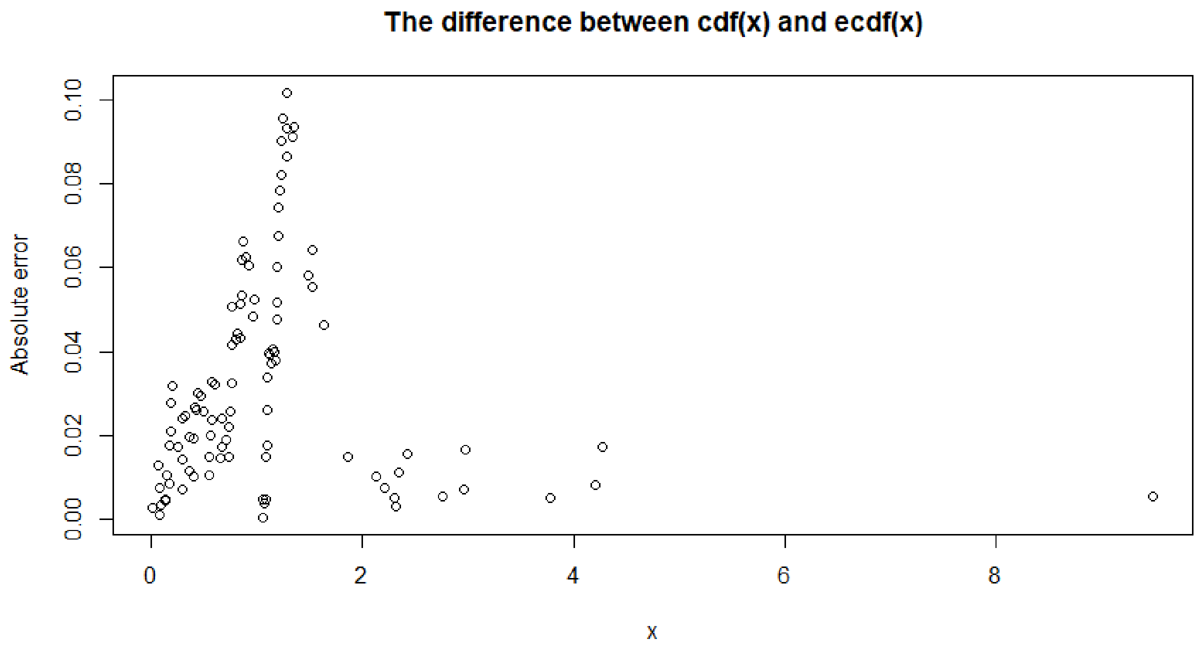


Fig. 17: Plots of absolute error between the cdf and ecdf

In addition, we use the rejection sampling method with the proposal $g(x)$ to simulate $X \sim f(x)$ and plot the samples in Fig. 18. Based on the simulations, we calculated the acceptance rate of the rejection sampler is 0.44. This figure and related results is almost the same as those in 4.1.

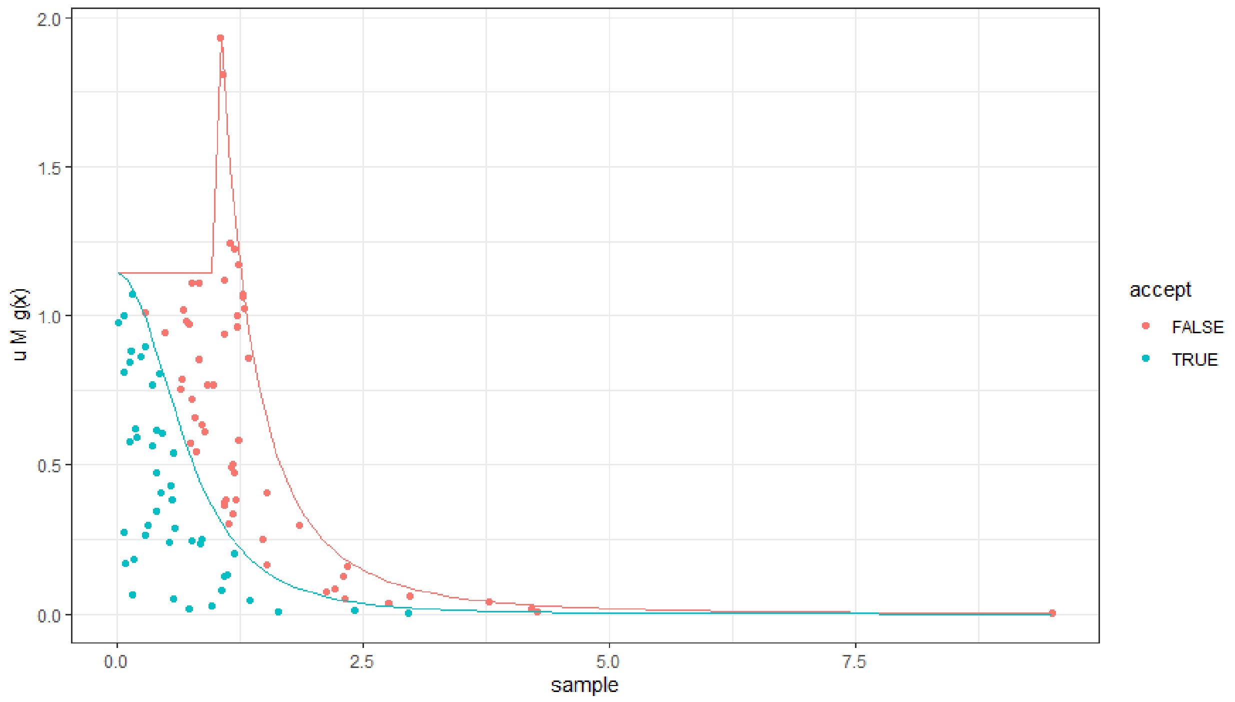
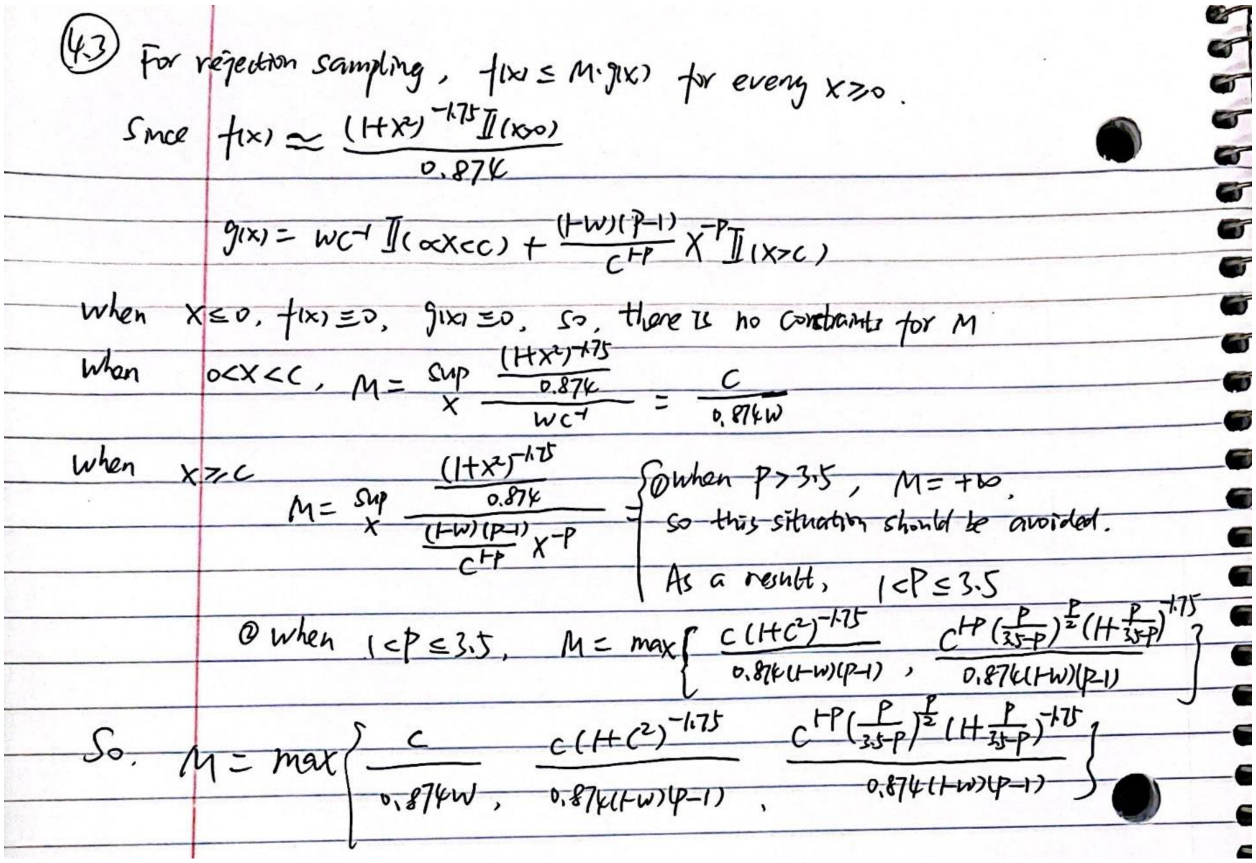


Fig. 18: Plots of samples derived from the rejection samplers

4.3



4.3 For rejection sampling, $f(x) \le M \cdot g(x)$ for every $x \geq 0$.

Since $f(x) \approx \dfrac{(1+x^2)^{-1.75}\mathbb{I}(x>0)}{0.874}$

$$g(x) = wc^{-1}\,\mathbb{I}(0<x<c) + \frac{(1-w)(p-1)}{c^{1-p}} x^{-p}\,\mathbb{I}(x>c)$$

when $x \le 0$, $f(x) \equiv 0$, $g(x) \equiv 0$, so, there is no constraints for M.

when $0<x<c$, $M = \sup\limits_x \dfrac{\frac{(1+x^2)^{-1.75}}{0.874}}{wc^{-1}} = \dfrac{c}{0.874w}$

when $x \geq c$

$$M = \sup_x \frac{\frac{(1+x^2)^{-1.75}}{0.874}}{\frac{(1-w)(p-1)}{c^{1-p}} x^{-p}} = \begin{cases} \text{① when } p>3.5, \ M=+\infty, \\ \text{so this situation should be avoided.} \\ \text{As a result, } 1<p\le 3.5 \end{cases}$$

② when $1<p\le 3.5$, $M = \max\left\{ \dfrac{c(1+c^2)^{-1.75}}{0.874(1-w)(p-1)}, \dfrac{c^{1-p}\left(\frac{p}{3.5-p}\right)^{\frac{p}{2}}\left(1+\frac{p}{3.5-p}\right)^{-1.75}}{0.874(1-w)(p-1)} \right\}$

So, $M = \max\left\{ \dfrac{c}{0.874w}, \dfrac{c(1+c^2)^{-1.75}}{0.874(1-w)(p-1)}, \dfrac{c^{1-p}\left(\frac{p}{3.5-p}\right)^{\frac{p}{2}}\left(1+\frac{p}{3.5-p}\right)^{-1.75}}{0.874(1-w)(p-1)} \right\}$

Here, we vary the values of p, c, w, separately from 1.1 to 3.4, from 1 to 10, from 0.1 to 0.9, to determine their effects on the efficiency of the rejection sampler. We find that when $c = 1, w = 0.8, p = 2.3$, the rejection sampler is the most efficient (the smallest M) among all cases.

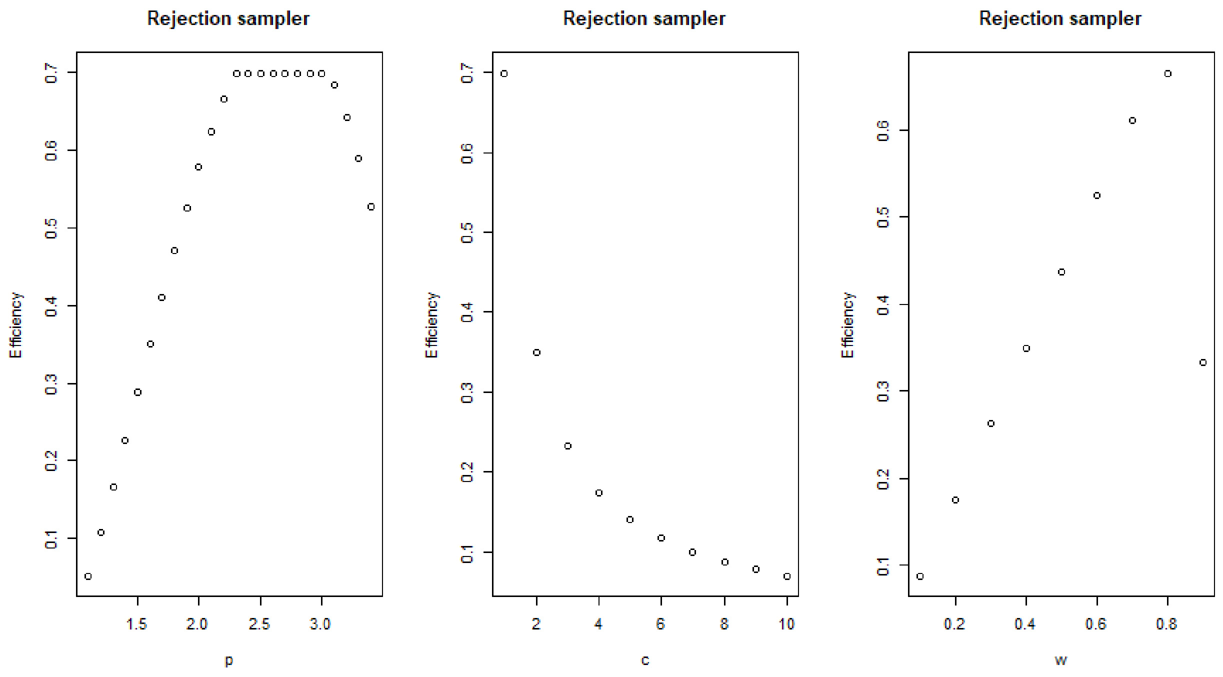


Fig. 19: Plot of effects of p, c, w on the efficiency of the rejection sampler

4.4 We use the rejection sampling method with the proposal $g(x)$ from 4.3 where $c = 1, w = 0.8, p = 2.3, J = 10$^5 to simulate $X \sim f(x)$ and plot the samples in Fig. 20. Based on the simulations, we calculated the acceptance rate of the rejection sampler is 0.70. Since most samples fall into the range of [0,10] and only a few samples are distributed in the range of $[10, \infty]$, the rejection sampling method with proposal $g(x)$ from 4.3 is not very appropriate or attractive for estimating the expectation $E\big(h(X)\big) = \int_{-\infty}^{+\infty} h(x) f(x)\, dx$ where $h(x) = x\mathrm{II}(x \geq 10)$. In addition, Fig. 21 plots how the Monte Carlo estimate and 95% credible interval of $E\big(h(X)\big)$ vary with an increase in the sample size and based on the strong law of larger numbers, the Monte Carlo estimate of $E\big(h(X)\big)$ eventually converge to 0.018, which is very close to the analytical integration 0.021.

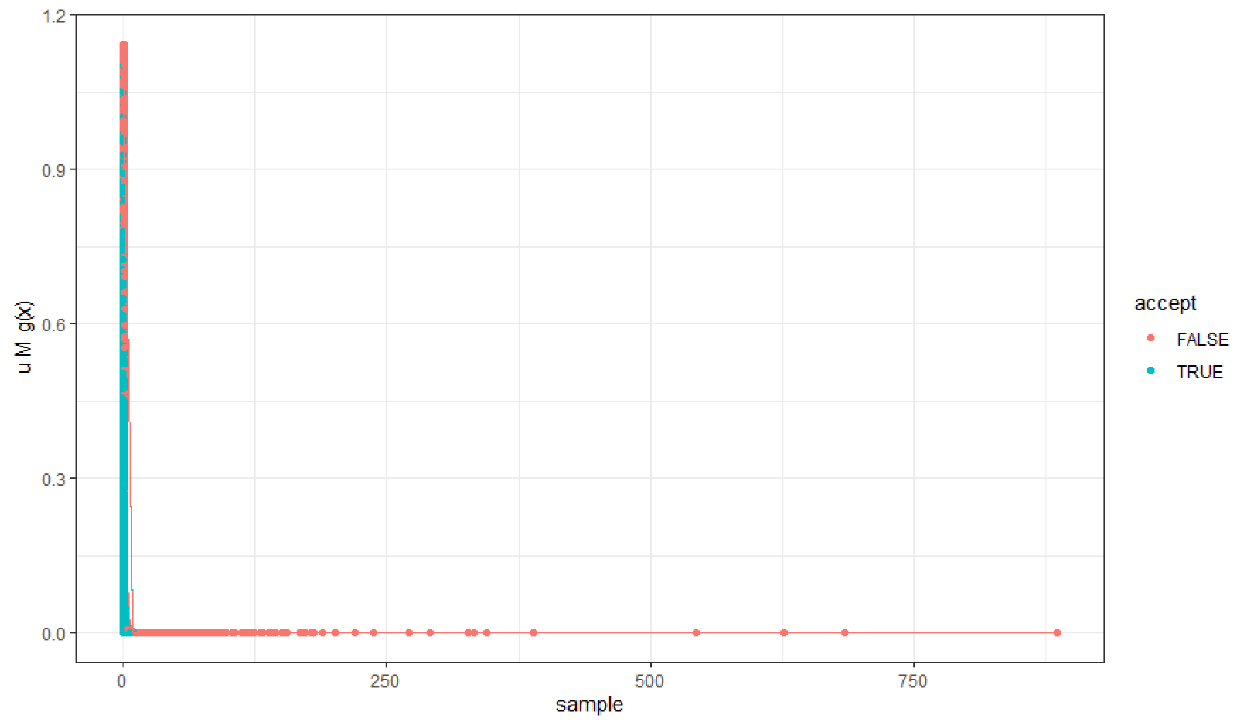


Fig. 20: Plots of samples derived from the rejection samplers with the proposal from 4.3

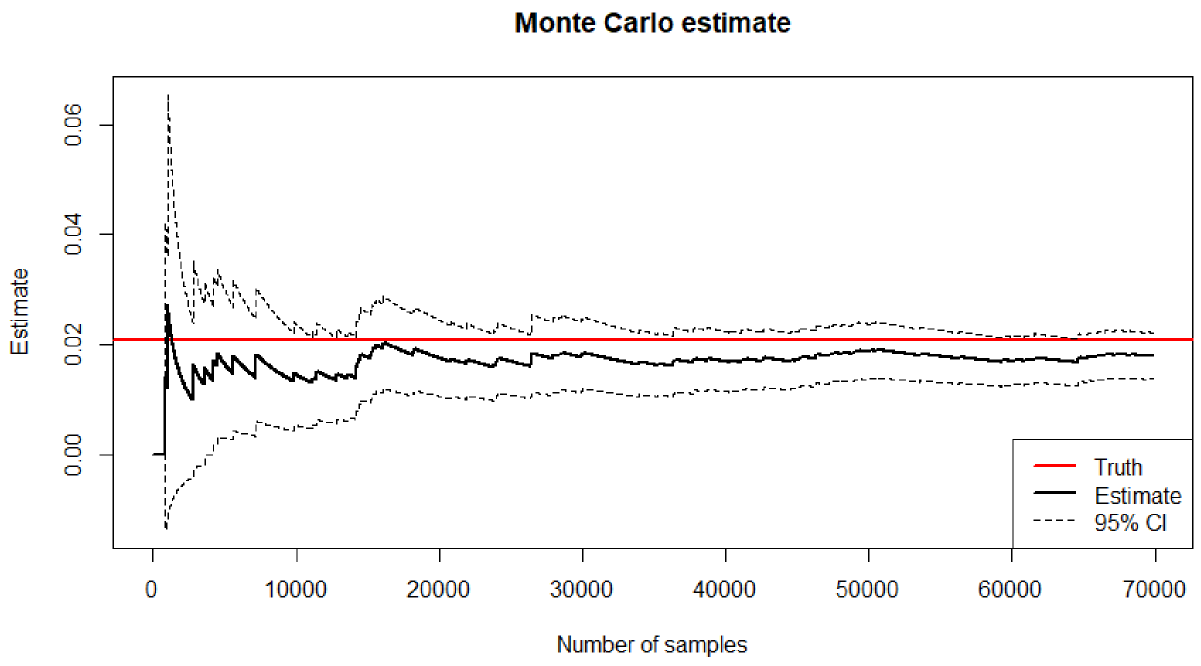


Fig. 21: Monte Carlo estimate and 95% credible interval of $E(h(X))$

When we set the sample size $J = 10\text{^}5$, we find that the minimum of relative error is still higher than 10%. In other words, the Monte Carlo sample sizes required by the rejection sampling method is larger than 10^5 if we want the desired relative error is 10% and 1%. Therefore, we increase the sample size J from 10^5 to 10^6 and plot how the relative error varies by the sample size in Fig. 22. We conclude that if we want the desired relative error is 10%, the MC sample size required by the rejection sampling method is larger than 596529 (about $6 * 10\text{^}5$). If we want the desired relative error is 1%, the MC sample size required by the rejection sampling method must be larger than $10\text{^}6$ (about $5{\sim}6 * 10\text{^}6$) because the minimum of relative error is 0.0744 when $J = 10\text{^}6$.

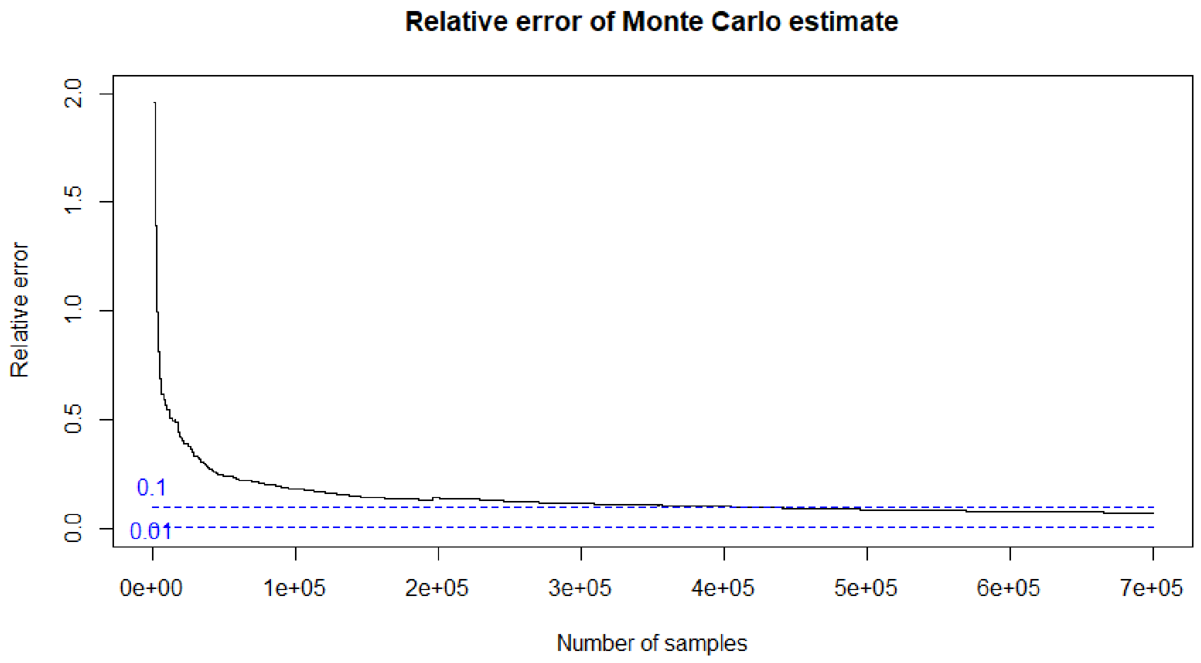
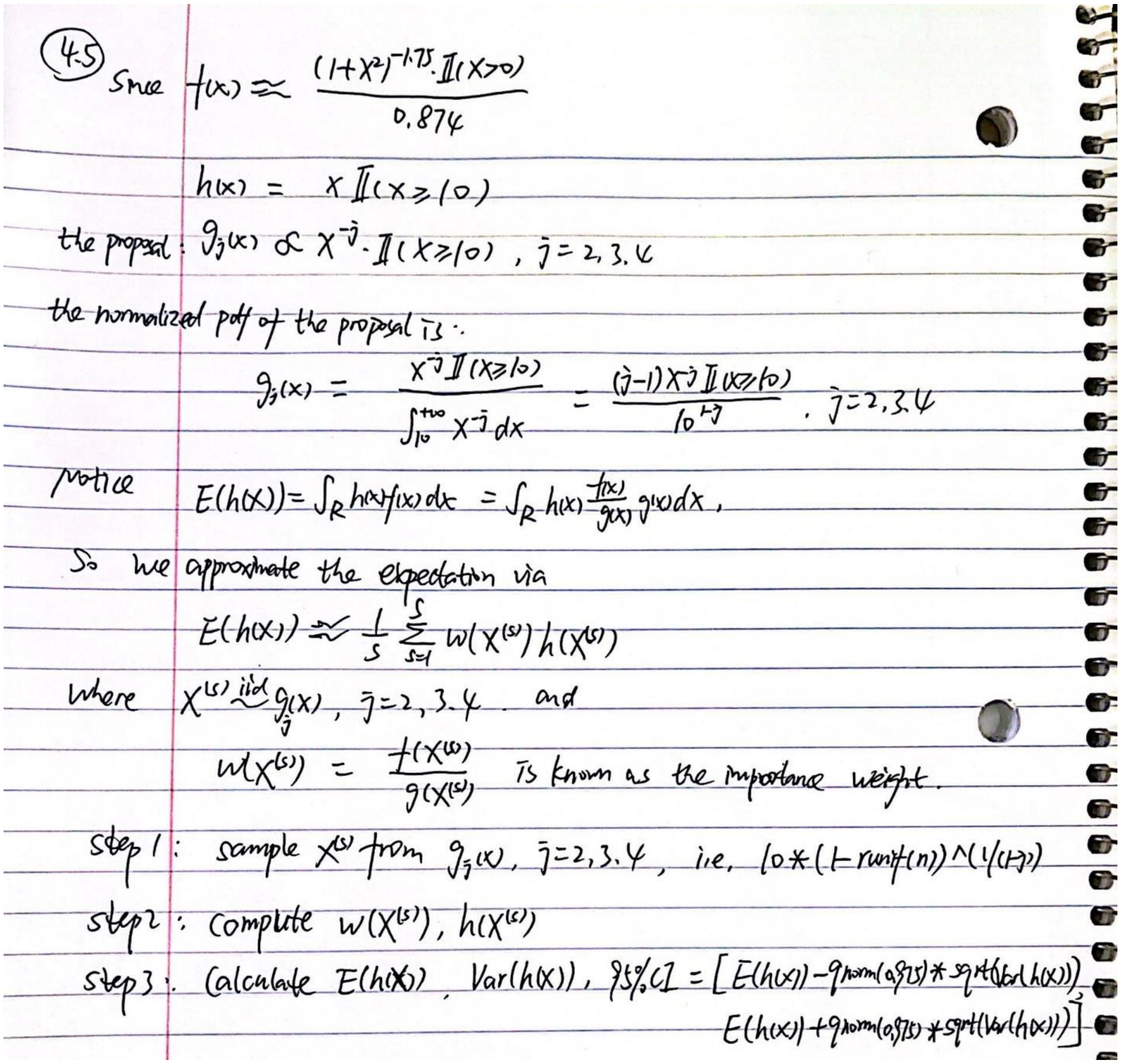


Fig. 22: Plots of change of relative error by the number of samples

4.5



$\boxed{4.5}$ Since $f(x) \approx \frac{(1+x^2)^{-1.75}\cdot\mathbb{I}(x>0)}{0.874}$

$h(x) = x\,\mathbb{I}(x \geq 10)$

the proposal: $g_j(x) \propto x^{-j}\cdot\mathbb{I}(x\geq 10)$, $j=2,3,4$

the normalized pdf of the proposal is:

$$g_j(x) = \frac{x^{-j}\,\mathbb{I}(x\geq 10)}{\int_{10}^{+\infty} x^{-j}\,dx} = \frac{(j-1)x^{-j}\,\mathbb{I}(x\geq 10)}{10^{1-j}}, \quad j=2,3,4$$

Notice $E(h(X)) = \int_R h(x)f(x)\,dx = \int_R h(x)\frac{f(x)}{g(x)}g(x)\,dx$,

So we approximate the expectation via

$$E(h(X)) \approx \frac{1}{S}\sum_{s=1}^{S} w(X^{(s)})\,h(X^{(s)})$$

where $X^{(s)} \overset{iid}{\sim} g_j(x)$, $j=2,3,4$ and

$w(X^{(s)}) = \frac{f(X^{(s)})}{g(X^{(s)})}$ is known as the importance weight.

step 1: sample $X^{(s)}$ from $g_j(x)$, $j=2,3,4$, i.e., 10*(1-runif(n))^(1/(1-j))

step2: compute $w(X^{(s)})$, $h(X^{(s)})$

step3: calculate $E(h(X))$, $Var(h(X))$, 95%CI = [E(h(x))-qnorm(0.975)*sqrt(Var(h(x))), E(h(x))+qnorm(0.975)*sqrt(Var(h(x)))]

Then we let $j = 2,3,4$, estimate $E\big(h(X)\big)$ and its accuracy, (i.e., 95% equal-tailed approximate confidence interval, CI) by importance sampling using each g_j, and further plot the estimates and the corresponding CIs in Fig. 23-25.

(1) $j = 2$:

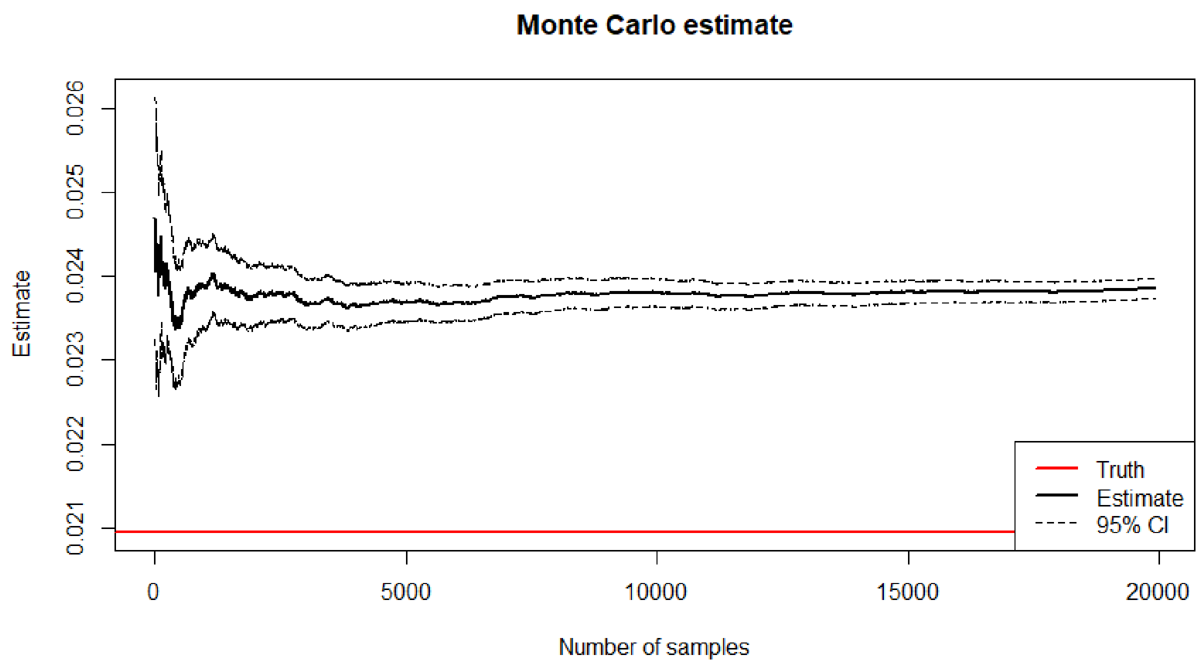


Fig. 23: MC estimates and 95% credible intervals of $E(h(X))$ by importance sampling using g_2

(2) $j = 3$:

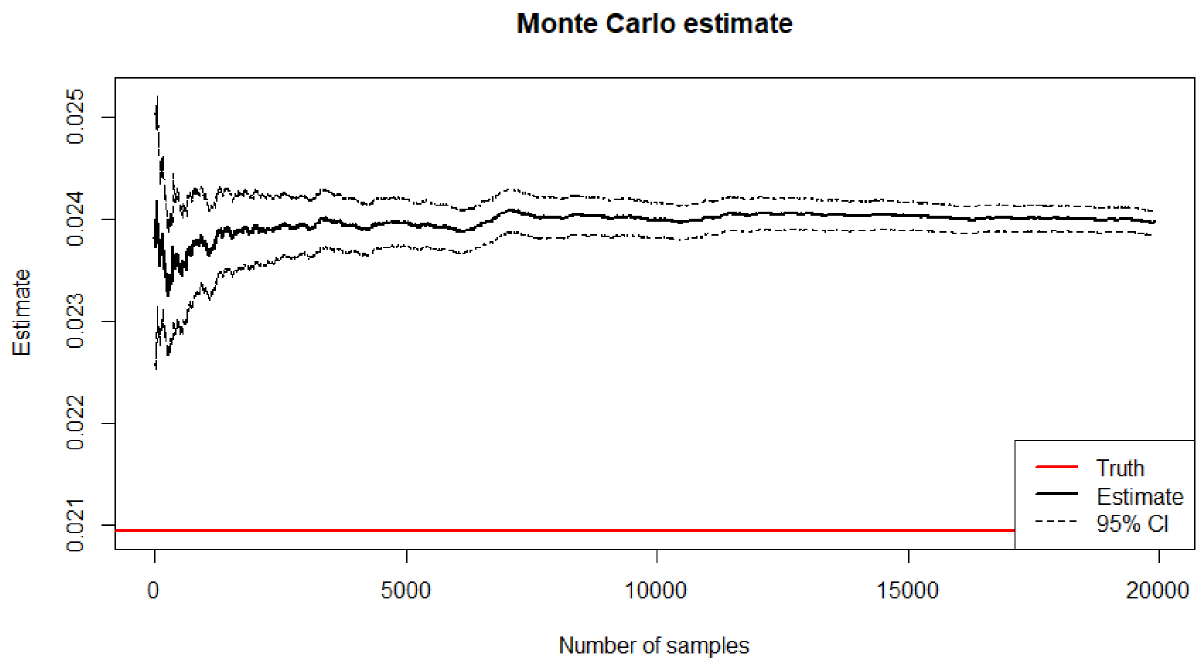


Fig. 24: MC estimates and 95% credible intervals of $E(h(X))$ by importance sampling using g_3

(3) $j = 4$:

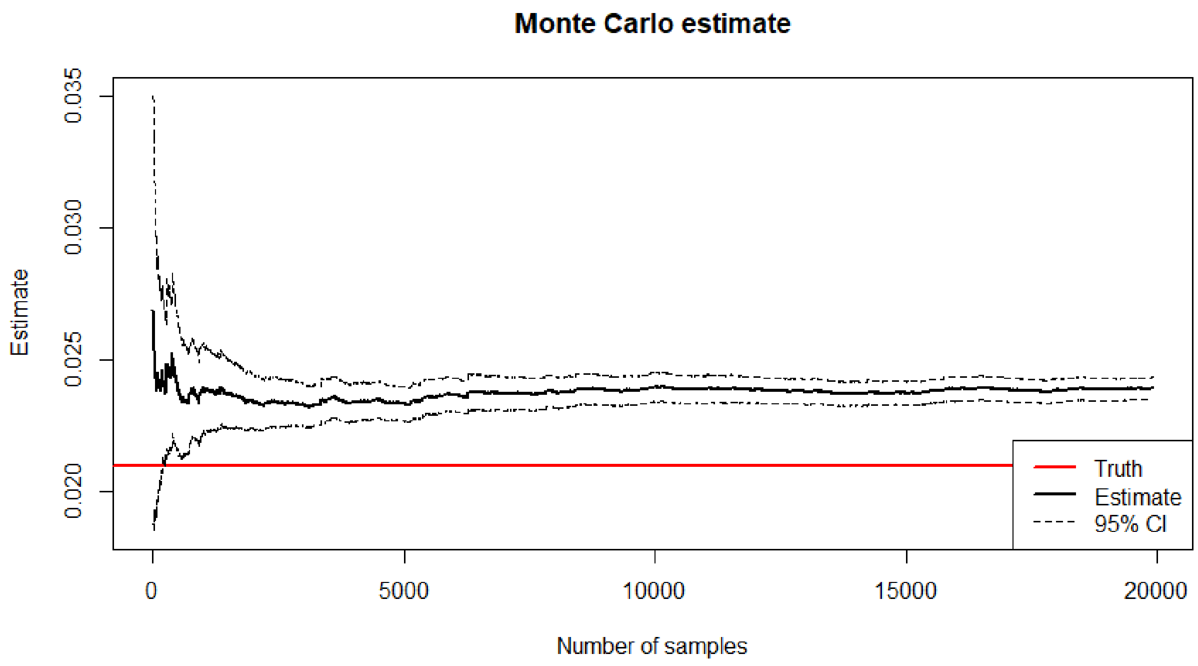


Fig. 24: MC estimates and 95% credible intervals of $E\left(h(X)\right)$ by importance sampling using g_4

Whether g_2, g_3 or g_4, the Monte Carlo estimates of $E\left(h(X)\right)$ based on importance sampling using the proposal g all eventually converge to 0.024 with an increase in the number of samples, which is very close to the analytical integration 0.021. In addition, these results follow the strong law of larger numbers (SLLN), reaching convergence rather than divergence. The length of 95% eqaul-tailed approximate credible intervals (CIs) is very small and densely concentrated in the MC estimates, which is significantly smaller than those derived from 4.4 (shown in Fig. 21), indicating that the importance sampling method is very efficient in approximating the expectation of $h(X)$ and estimating $E\left(h(X)\right)$.