Bayesian Statistics : week 2

와이빅타 사이언스팀 김주은

Lesson 4. Metropolis-Hastings

Algorithm

stationary distribution(target distribution) $p(\theta) \propto g(\theta)$

normalizing constant($g(\theta)$의 적분값)를 모르므로 그냥 $g(\theta)$를 이용해서 구한다

1. Select initial value θ_0
2. for $i = 1, \dots, m$ repeat:

   a) Draw candidate θ^* ~ $q(\theta^*|\theta_{i-1})$ (proposal distribution)

   b) $\alpha = \frac{g(\theta^*)/q(\theta^*|\theta_{i-1})}{g(\theta_{i-1})/q(\theta_{i-1}|\theta^*)} = \frac{g(\theta^*)q(\theta_{i-1}|\theta^*)}{g(\theta_{i-1})q(\theta^*|\theta_{i-1})}$

   c) if $\alpha \geq 1$, acccept θ^* and set $\theta_i \leftarrow \theta^*$

   if $0 < \alpha < 1$, acccept θ^* and set $\theta_i \leftarrow \theta^*$ with probability α

   reject θ^* and set $\theta_i \leftarrow \theta_{i-1}$ with probability $1 - \alpha$

   if $\alpha \leq 0$, reject θ^* and set $\theta_i \leftarrow \theta_{i-1}$

: Markov Chain (현재 체인의 상태에만 의존하므로)

Proposal distribution

- dependency를 허용하게 되는 경우: random walk Metropolis-Hastings
  - proposal distribution이 직전에 뽑은 샘플 θ_{i-1} 에 의존함
  - 정규분포의 경우 대칭적 - $q(\theta_{i-1}|\theta^*) = q(\theta^*|\theta_{i-1})$

    따라서 위 알고리즘에서의 α 가 $\frac{g(\theta^*)}{g(\theta_{i-1})}$ 로 간단하게 표현됨

Acceptance rate

- θ 샘플의 업데이트
  - small steps - high acceptance rate - requires long time to get full posterior distribution
  - large steps - low acceptance rate - waste of draws
  - 23~50%의 acceptance rate가 적절하다

Example

percentage change in total personnel for 10 companies

$p(\mu|y_1, \dots, y_n) \propto \frac{\exp(n(\bar{y}\mu - \mu^2/2))}{1+\mu^2}$: posterior distribution of non-conjugate model

Take log and get $\log(g(\mu)) = n(\bar{y}\mu - \mu^2/2) - \log(1 + \mu^2)$

```
lg = function(mu, n, ybar) {
  mu2 = mu^2
  n * (ybar * mu - mu2 / 2.0) - log(1 + mu2)
}
```

Random-Walk Metropolis-Hastings sampler function

```
mh = function(n, ybar, n_iter, mu_init, cand_sd) {
  ## Random-Walk Metropolis-Hastings algorithm

  ## step 1, initialize
  mu_out = numeric(n_iter)
  accpt = 0
  mu_now = mu_init
  lg_now = lg(mu=mu_now, n=n, ybar=ybar)

  ## step 2, iterate
  for (i in 1:n_iter) {
    ## step 2a
    mu_cand = rnorm(n=1, mean=mu_now, sd=cand_sd) # draw a candidate

    ## step 2b
    lg_cand = lg(mu=mu_cand, n=n, ybar=ybar) # evaluate log of g with the
candidate
    lalpha = lg_cand - lg_now # log of acceptance ratio
    alpha = exp(lalpha)

    ## step 2c
    u = runif(1) # draw a uniform variable which will be less than alpha with
probability min(1, alpha)
    if (u < alpha) { # then accept the candidate
      mu_now = mu_cand
      accpt = accpt + 1 # to keep track of acceptance
      lg_now = lg_cand
    }

    ## collect results
    mu_out[i] = mu_now # save this iteration's value of mu
  }

  ## return a list of output
  list(mu=mu_out, accpt=accpt/n_iter)
}
```

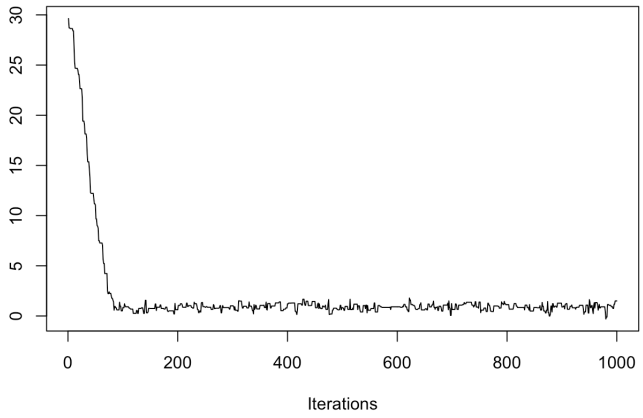
Set up the problem

```
y = c(1.2, 1.4, -0.5, 0.3, 0.9, 2.3, 1.0, 0.1, 1.3, 1.9)
ybar = mean(y)
n = length(y)

set.seed(43) # set the random seed for reproducibility
post = mh(n=n, ybar=ybar, n_iter=1e3, mu_init=0.0, cand_sd=0.9)
post$accpt
traceplot(as.mcmc(post$mu))
```

proposal step size too large (acceptance rate below 23%) -> cand_sd too large -> lower sd

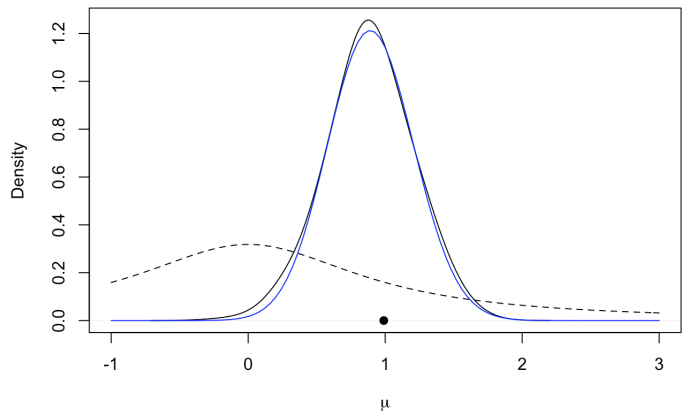
acceptance rate is too high (above 50%) -> cand_sd too small -> higher sd



plot the posterior density against the prior

```
post$mu_keep = post$mu[-c(1:100)] # discard the first 200 samples
plot(density(post$mu_keep, adjust=2.0), main="", xlim=c(-1.0, 3.0),
xlab=expression(mu)) # plot density estimate of the posterior
curve(dt(x=x, df=1), lty=2, add=TRUE) # prior for mu
points(ybar, 0, pch=19) # sample mean

curve(0.017*exp(lg(mu=x, n=n, ybar=ybar)), from=-1.0, to=3.0, add=TRUE,
col="blue") # approximation to the true posterior in blue
```



Lesson 5. Gibbs Sampling

posterior distribution of multiple parameters?

방법 1) Metropolis-Hastrings algorithm으로 매 번 모든 파라미터를 샘플링하고 accept/reject

방법 2) 파라미터를 1개씩 샘플링: 하나의 파라미터 샘플링 -> 업데이트 -> 업데이트 된 파라미터 이용해서 다른 파라미터 샘플링 -> 업데이트 -> ...

Full conditional distributions

Chain Rule : $p(\theta, \phi|y) = p(\theta|\phi, y)p(\phi|y)$

$p(\theta|\phi, y) \propto p(\theta, \phi|y) \propto g(\theta, \phi)$, where $p(\theta|\phi, y)$ is full conditional distribution of θ

full conditional distribution: 다른 모든 파라미터를 안다는 가정 아래 각 파라미터의 posterior distribution

Gibbs sampler

방법 2: 파라미터가 여러 개 주어졌을 때, 한 번에 1개의 파라미터만 샘플링하여 업데이트하고 모든 파라미터에 대해서 똑같이 샘플링하는 사이클을 반복한다. 각 파라미터를 샘플링할 때, 다른 파라미터들의 업데이트된 값을 반영한다.

θ, ϕ 의 joint distribution을 구하기 위해 full conditional distribution을 이용한다. 즉, $p(\theta, \phi|y)$ 를 구하기 위해 $p(\theta|\phi, y), p(\phi|\theta, y)$ 로부터 샘플링:

1. Initialize θ_0, ϕ_0
2. for $i = 1, \ldots, m$ repeat:

   a) Using ϕ_{i-1}, draw θ_i ~ $p(\theta|\phi_{i-1}, y)$

   b) Using θ_i from a), draw ϕ_i ~ $p(\phi|\theta_i, y)$

   where a), b) forms a cycle with pair of (θ_i, ϕ_i)

full conditional for μ, given σ^2 : $N(\mu|\frac{n\bar{y}/\sigma^2+\mu_0/\sigma_0^2}{n/\sigma^2+1/\sigma_0^2}, \frac{1}{n/\sigma^2+1/\sigma_0^2})$

```
update_mu = function(n, ybar, sig2, mu_0, sig2_0) {
  sig2_1 = 1.0 / (n / sig2 + 1.0 / sig2_0)
  mu_1 = sig2_1 * (n * ybar / sig2 + mu_0 / sig2_0)
  rnorm(n=1, mean=mu_1, sd=sqrt(sig2_1))
}
```

full conditional for σ^2 given μ: $IG(\sigma^2|\nu_0 + \frac{n}{2}, \beta_0 + \frac{\Sigma_{i=1}^n (y_i-\mu)^2}{2})$

```
update_sig2 = function(n, y, mu, nu_0, beta_0) {
  nu_1 = nu_0 + n / 2.0
  sumsq = sum( (y - mu)^2 ) # vectorized
  beta_1 = beta_0 + sumsq / 2.0
  out_gamma = rgamma(n=1, shape=nu_1, rate=beta_1) # rate for gamma is shape
for inv-gamma
  1.0 / out_gamma # reciprocal of a gamma random variable is distributed inv-
gamma
}
```

Gibbs sampling function: update sigma -> mu in each cycle

```
gibbs = function(y, n_iter, init, prior) {
  ybar = mean(y)
  n = length(y)

  ## initialize
  mu_out = numeric(n_iter)
  sig2_out = numeric(n_iter)

  mu_now = init$mu

  ## Gibbs sampler
  for (i in 1:n_iter) {
    sig2_now = update_sig2(n=n, y=y, mu=mu_now, nu_0=prior$nu_0,
beta_0=prior$beta_0)
    mu_now = update_mu(n=n, ybar=ybar, sig2=sig2_now, mu_0=prior$mu_0,
sig2_0=prior$sig2_0)

    sig2_out[i] = sig2_now
    mu_out[i] = mu_now
  }

  cbind(mu=mu_out, sig2=sig2_out)
}
```

Set up the problem

```
y = c(1.2, 1.4, -0.5, 0.3, 0.9, 2.3, 1.0, 0.1, 1.3, 1.9)
ybar = mean(y)
n = length(y)

## prior
prior = list()
prior$mu_0 = 0.0
prior$sig2_0 = 1.0
prior$n_0 = 2.0 # prior effective sample size for sig2
prior$s2_0 = 1.0 # prior point estimate for sig2
```

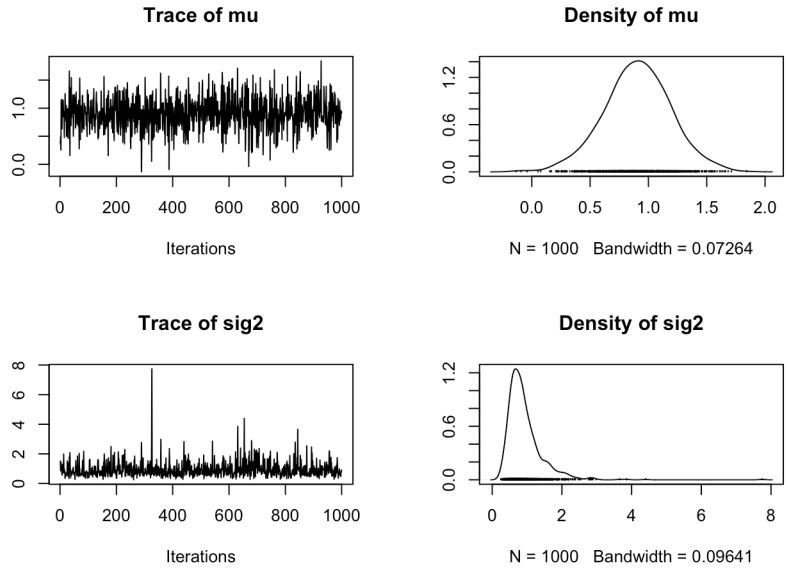
```r
prior$nu_0 = prior$n_0 / 2.0 # prior parameter for inverse-gamma
prior$beta_0 = prior$n_0 * prior$s2_0 / 2.0 # prior parameter for inverse-gamma

set.seed(53)

init = list()
init$mu = 0.0

post = gibbs(y=y, n_iter=1e3, init=init, prior=prior)
```

```r
library("coda")
plot(as.mcmc(post))
summary(as.mcmc(post))
```




```
## Iterations = 1:1000
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 1000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##         Mean     SD Naive SE Time-series SE
## mu   0.9051 0.2868  0.00907        0.00907
## sig2 0.9282 0.5177  0.01637        0.01810
##
## 2. Quantiles for each variable:
##
##         2.5%    25%    50%   75% 97.5%
## mu   0.3024 0.7244 0.9089 1.090 1.481
## sig2 0.3577 0.6084 0.8188 1.094 2.141
```

Lesson 6. Convergence diagnostics

Autocorrelation

autocorrelation: MCMC 체인의 현재 값과 과거 값(lags) 간의 선형 의존성(linear dependency)을 측정하는 정도

$$COV(X,Y) = E[(X - E(X))(Y - E(Y))]$$
$$\rho_{XY} = COR(X,Y) = \frac{COV(X,Y)}{\sqrt{Var(X)Var(Y)}} = \frac{\sigma_{XY}}{\sqrt{\sigma_X^2 \sigma_Y^2}}$$

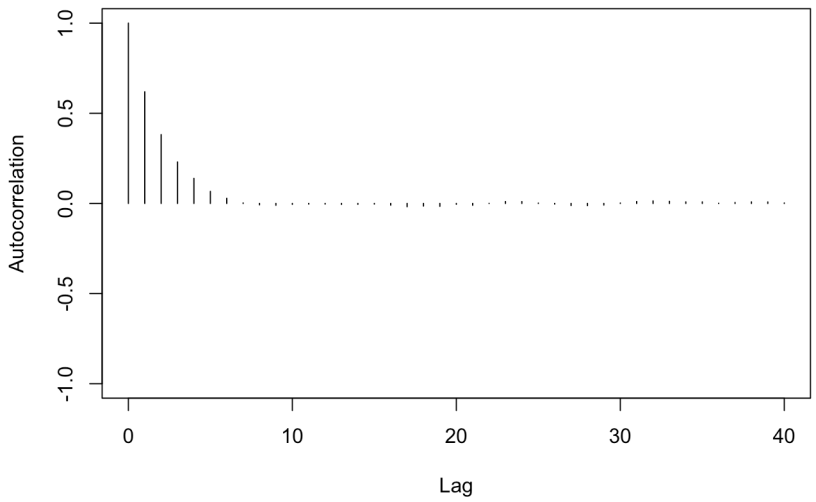
ACF (autocorrelation): correlation of X_t and X_{t-l}

$$ACF(X_t, X_{t-l}) = \frac{COV(X_t, X_{t-l})}{\sqrt{Var(X_t)Var(X_{t-l})}}$$

Stationary distribution: autocorrelation 값이 t 가 아닌 두 값 사이의 차이 (X_t 와 X_{t-l} 이라면 l)에 의존한다.

$$ACF(X_t, X_{t-l}) = ACF(X_s, X_{s+l})$$

```
set.seed(61)
post0 = mh(n=n, ybar=ybar, n_iter=10e3, mu_init=0.0, cand_sd=0.9)
coda::autocorr.plot(as.mcmc(post0$mu))
```



Effective sample size

1) number of samples needed to get the same information

2) length of chain after removal of autocorrelation

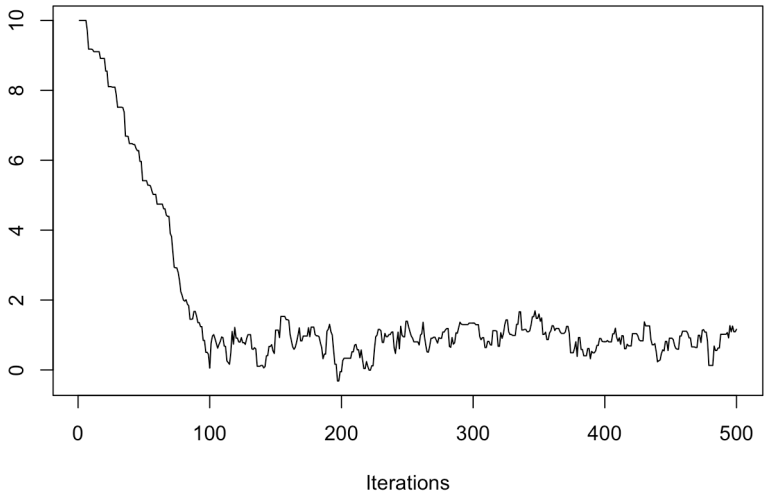
High autocorrelation -> low effective sample size (compare to total number of samples)

Low auto correlation -> large effective sample size

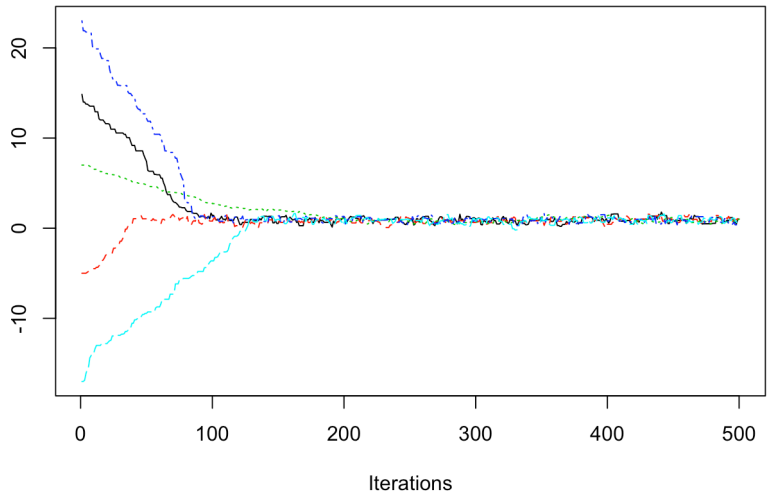
```
coda::effectiveSize(as.mcmc(post0$mu))
```

Burn-in period

early iterations of simulation : not from stationary distribution -> discard



Multiple chains

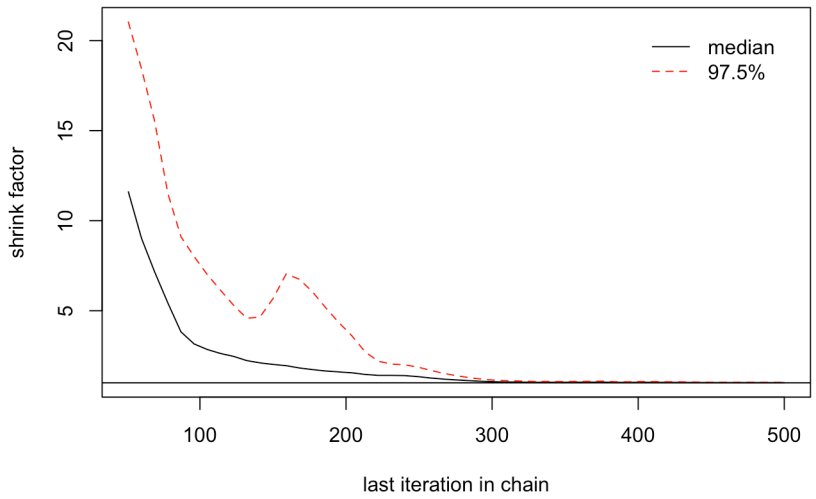


Gelman-Rubin diagnostic: variability between / within multiple chains

특징: 초기값이 모두 다르더라도 결국 같은 stationary distribution을 갖는다면 수렴한당!

초기 burn-in period에는 shrink factor > 1이지만 수렴한 이후에는 shrink factor = 1

```
coda::gelman.plot(pmc)
```


median
97.5%
shrink factor
5
10
15
20
100
200
300
400
500
last iteration in chain