HW 9

Problem 1

```
x <- c(-.86, -.3, -.05, .73)
n <- rep(5, 4)
y <- c(0, 1, 3, 5)
```

Checking the gradient.

```
grad <- function(a, b) {
  c(sum(y/(1+exp(a+b*x)) - (n-y)/(1+exp(-a-b*x))),
    sum(x*y/(1+exp(a+b*x)) - x*(n-y)/(1+exp(-a-b*x))))
}
```

The analytic gradient evaluated at $(1, 10)$ is

```
grad(1, 10)
```

```
## [1]  0.2904300 -0.1125231
```

The finite-difference approximation of gradient at $(1, 10)$ is

```
logp <- function(a, b) {
  sum(-y*log(1+exp(-a-b*x)) - (n-y)*log(1+exp(a+b*x)))
}

(logp(1+1e-5, 10) - logp(1, 10))/1e-5
```

```
## [1] 0.2904215
```

```
(logp(1, 10+1e-5) - logp(1, 10))/1e-5
```

```
## [1] -0.1125233
```

They do match up to several decimal places.

Implementation

```
set.seed(0)

hmc <- function(iter, init) {
  Theta <- matrix(0, nrow=iter+1, ncol=2)
  Theta[1, ] <- init

  M <- matrix(c(5, 0, 0, 5), 2, 2)
  eps <- .12
  L <- 10
  Accept <- rep(NA, iter)
```

```r
  for (i in 1:iter) {
  # Leapfrog
  Theta0 <- Theta[i, ]
  phi <- rnorm(2, mean=0, sd=diag(M))
  phi0 <- phi
  phi <- phi + .5*eps*grad(Theta0[1], Theta0[2])
  Theta0 <- Theta0 + eps*t(phi)%*%solve(M)
  for (j in 1:(L-1)) {
    phi <- phi + eps*grad(Theta0[1], Theta0[2])
    Theta0 <- Theta0 + eps*t(phi)%*%solve(M)
  }
  phi <- phi + .5*eps*grad(Theta0[1], Theta0[2])

  #Accept-reject sampling
  log_r <- logp(Theta0[1], Theta0[2]) - logp(Theta[i, 1], Theta[i, 2]) +
    sum(dnorm(phi, sd=diag(M), log=T)) - sum(dnorm(phi0, sd=diag(M), log=T))
  Accept[i] <- (log(runif(1)) < log_r)
  Theta[i+1, ] <- ifelse(rep(Accept[i], 2),
                         Theta0,
                         Theta[i, ])
  }
  return(list(chain=Theta, accept=Accept))
}

mean(hmc(1e4, c(0, 10))$accept)
```

```
## [1] 0.6668
```

Effective sample size.

```r
burn_in <- function(chain) {
  tail(chain, -.3*length(chain))
}

 my_diff <- function(t, x) {
    mean(diff(x, lag = t)^2, na.rm = TRUE)
 }

chain1 <- hmc(3000, c(0, 10))$chain
chain2 <- hmc(3000, c(10, 20))$chain
chain3 <- hmc(3000, c(-10, 30))$chain
chain4 <- hmc(3000, c(-20, 50))$chain

chains <- cbind(chain1[, 1], chain2[, 1], chain3[, 1], chain4[, 1])
chains <- apply(chains, 2, burn_in)

n_eff <- function(chains) {
  require(purrr)
  m <- ncol(chains)
  n <- nrow(chains)
  # Within-chain var
  W <- 1/m*sum(apply(chains, 2, var))
  means <- colMeans(chains)
  B <- n * var(means)

  V_plus <- (n-1)/n * W + 1/n * B

  V <- sapply(seq_len(nrow(chains) - 1), my_diff, x=chains[, 1]) + sapply(seq_len(nrow(chains) - 1), my_

  V <- V/m

  rho <- head_while(1 - V / (2*V_plus), ~. >0)

  n_hat_eff <- m*n / (1 + 2*sum(rho))
  return(n_hat_eff)
}

n_eff(chains)
```

```
## Loading required package: purrr
```

```
## [1] 106.1946
```

Using my tuned hyper-parameters, it takes 3000 iterations to achieve 100 effective sample size.

Check with direct approach

```r
colMeans(chain1)
```

```
## [1]  1.448485 12.267518
```

Back in homework 3 the mean estimation using direct approach is $(0.9960628, 10.5482460)$. It is consistent with the direct approach. There is error because the sample size is not large enough and we don't expect the estimation to be accurately matched.

Problem 2

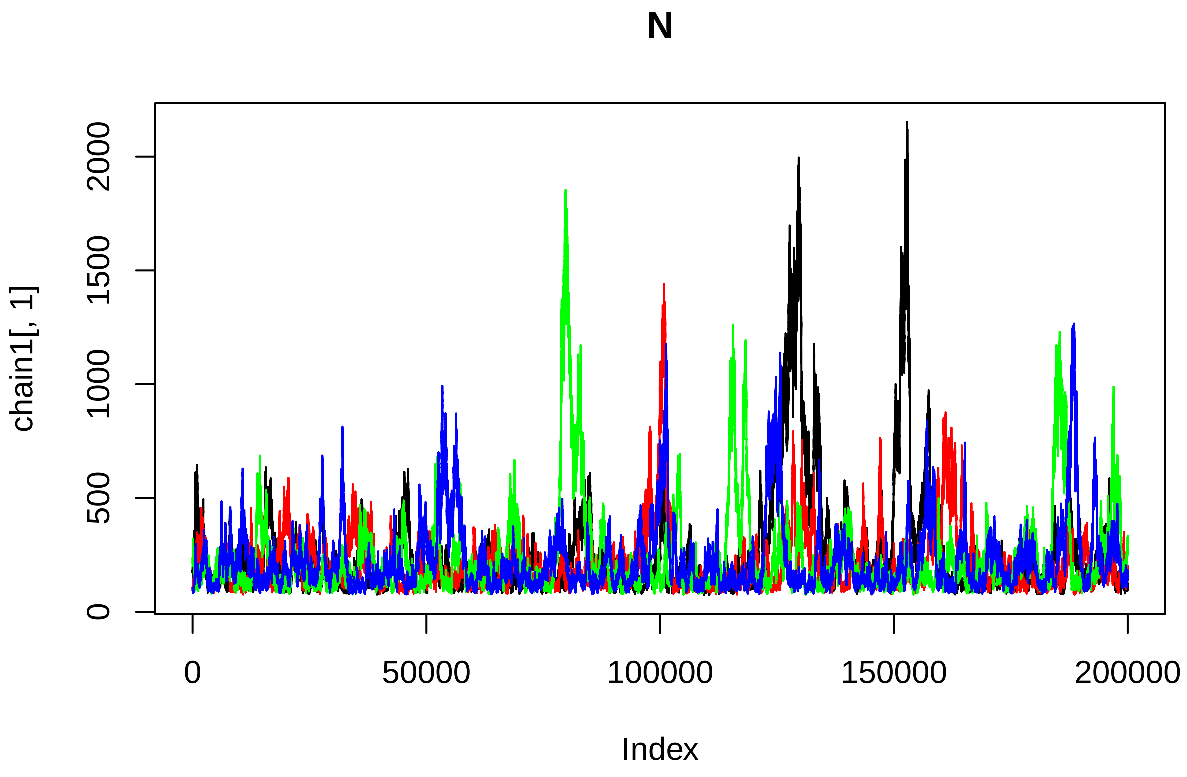
Sampling and diagnostic

```r
y <- c(53, 57, 66, 67, 72)
joint_density <- function(N, theta, y) {
  # y can be a vector
  prod(choose(N, y)*(theta^y)*(1-theta)^(N-y))
}

banana <- function(iter, init) {
  res <- matrix(nrow=iter+1, ncol=2)
  res[1, ] <- init #c(2*max(y), 1/2) init
  for (i in 1:iter) {
    temp_N <- rpois(1, lambda=res[i, 1]) # proposal
    log_ratio <- log(joint_density(temp_N, res[i, 2], y)) -
      log(joint_density(res[i, 1], res[i, 2], y)) +
      log(dpois(temp_N, lambda=res[i, 1])) -
      log(dpois(res[i, 1], lambda=temp_N))
    temp_log_U <- log(runif(1))
    if (temp_log_U <= log_ratio) {
      res[i+1, 1] <- temp_N
    } else {
      res[i+1, 1] <- res[i, 1]
    }

    temp_theta <- runif(1, min=max(0, res[i, 2]-.2), max=min(1, res[i,2]+.2)) # proposal
    log_ratio <- log(joint_density(res[i+1, 1], temp_theta, y)) -
      log(joint_density(res[i+1, 1], res[i, 2], y))
    # proposal ratio is always one so dropped
    temp_log_U <- log(runif(1))
    if (temp_log_U <= log_ratio) {
      res[i+1, 2] <- temp_theta
    } else {
      res[i+1, 2] <- res[i, 2]
    }
  }
  return(res)
}
```

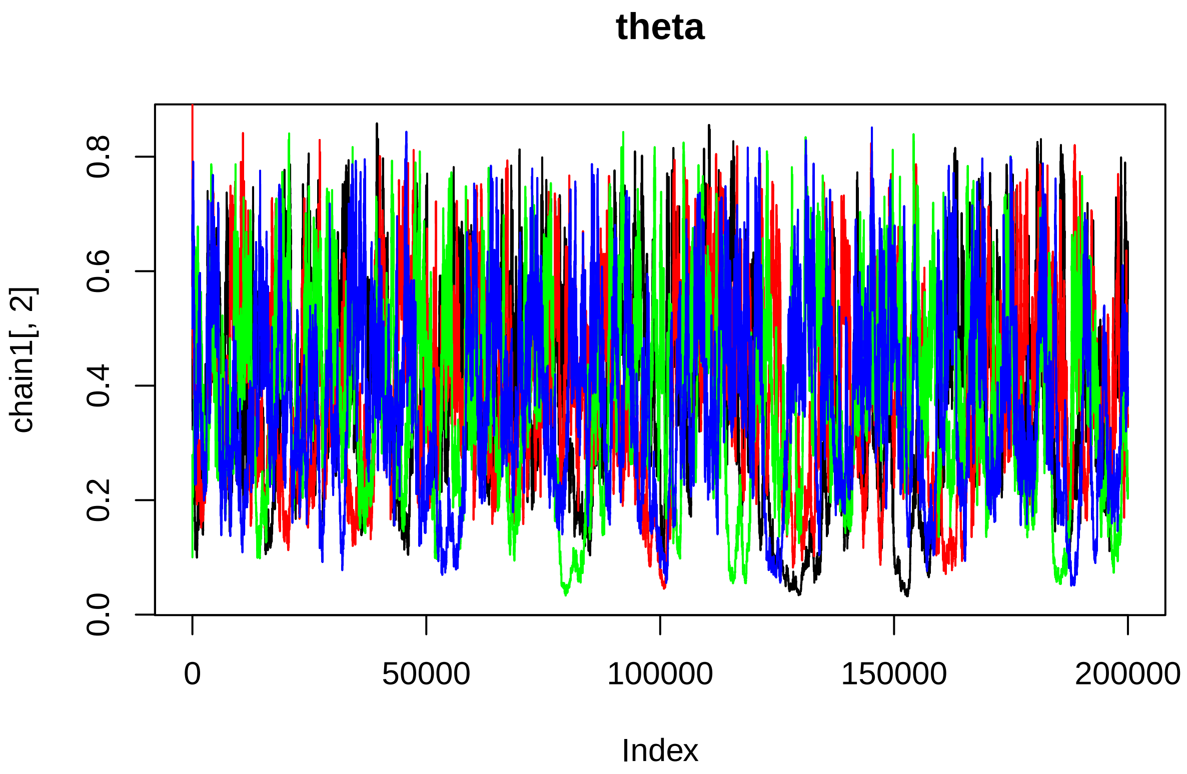
```
chain1 <- banana(2e5, c(200, .5))
chain2 <- banana(2e5, c(100, .9))
chain3 <- banana(2e5, c(200, .1))
chain4 <- banana(2e5, c(100, .5))
```

It takes too long to compute the $r\hat{h}o_t$ so I will do diagnostics only monitoring the mixing.

```
plot(chain1[, 1], type='l', main = 'N')
lines(chain2[, 1], col='red')
lines(chain3[, 1], col='green')
lines(chain4[, 1], col='blue')
```



```r
plot(chain1[, 2], type='l', main = 'theta')
lines(chain2[, 2], col='red')
lines(chain3[, 2], col='green')
lines(chain4[, 2], col='blue')
```



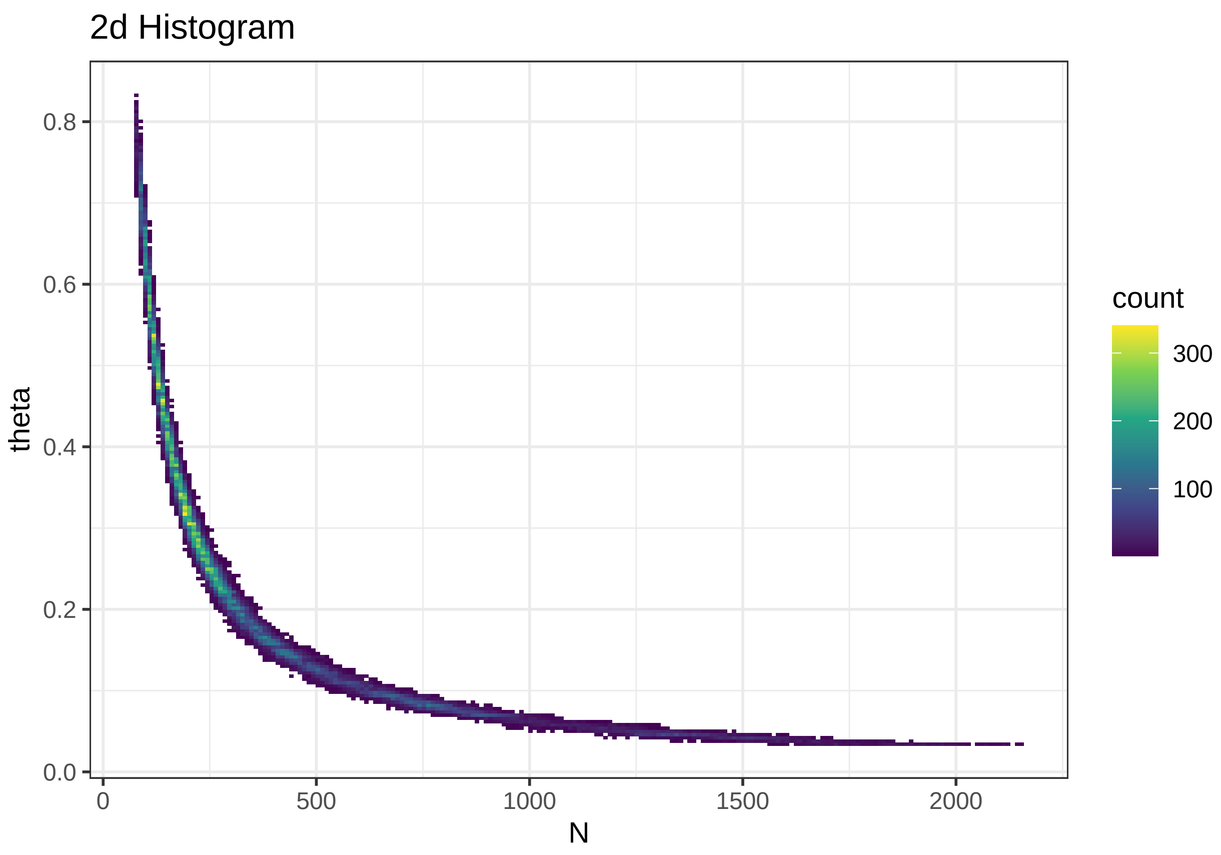
It looks like for θ the chains are indistinguishable but for N there are some disturbance. I think it is good enough.

Posterior density visualization

2d histogram

Let's first take a look at a "nice" plot. It looks better comparing to the scatterplot and contour plot.

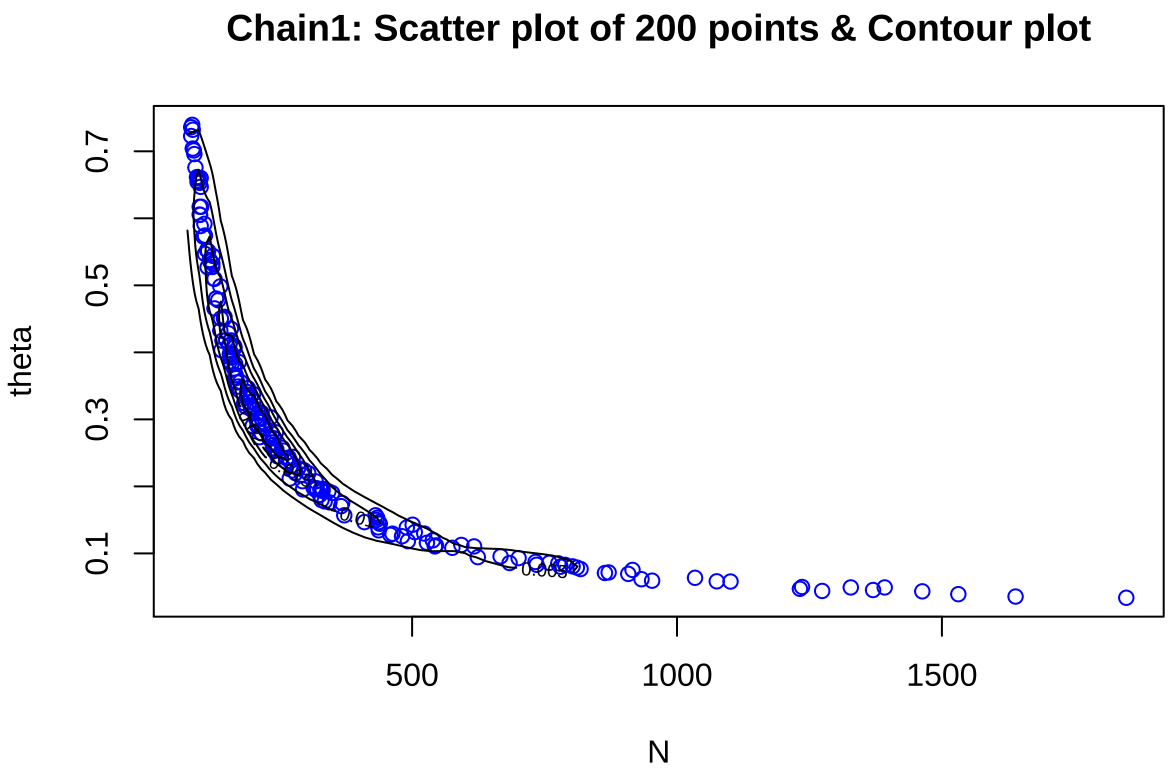
```r
library(ggplot2)
res <- burn_in(chain1) # remove burn-in
plt_data <- as.data.frame(res)
colnames(plt_data) <- c('N', 'theta')
ggplot(plt_data, aes(x=N, y=theta) ) +
  geom_bin2d(bins = 200) +
  scale_fill_continuous(type = "viridis") +
  theme_bw() + ggtitle('2d Histogram')
```



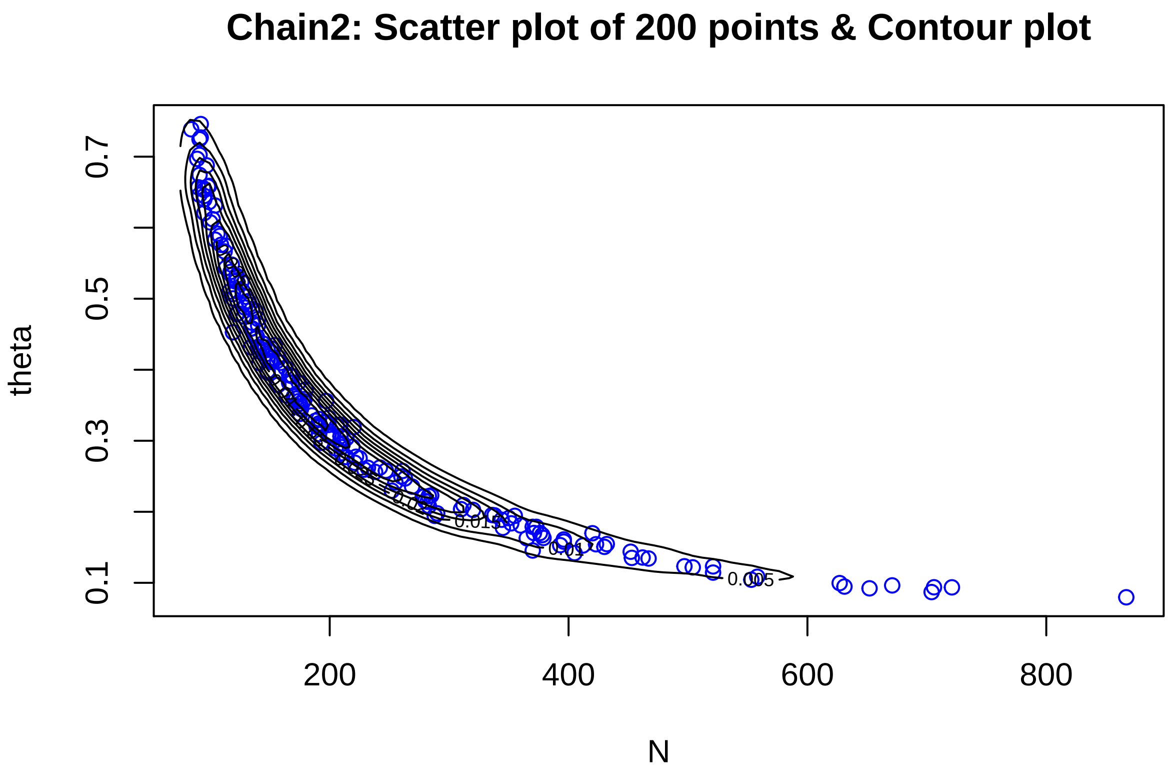
Scatterplot and contour plot

It is not reasonable to do a 200000 size scatterplot so I just randomly picked 200 points from the chain to draw the scatterplot. The coutour plot is using the entire chain.

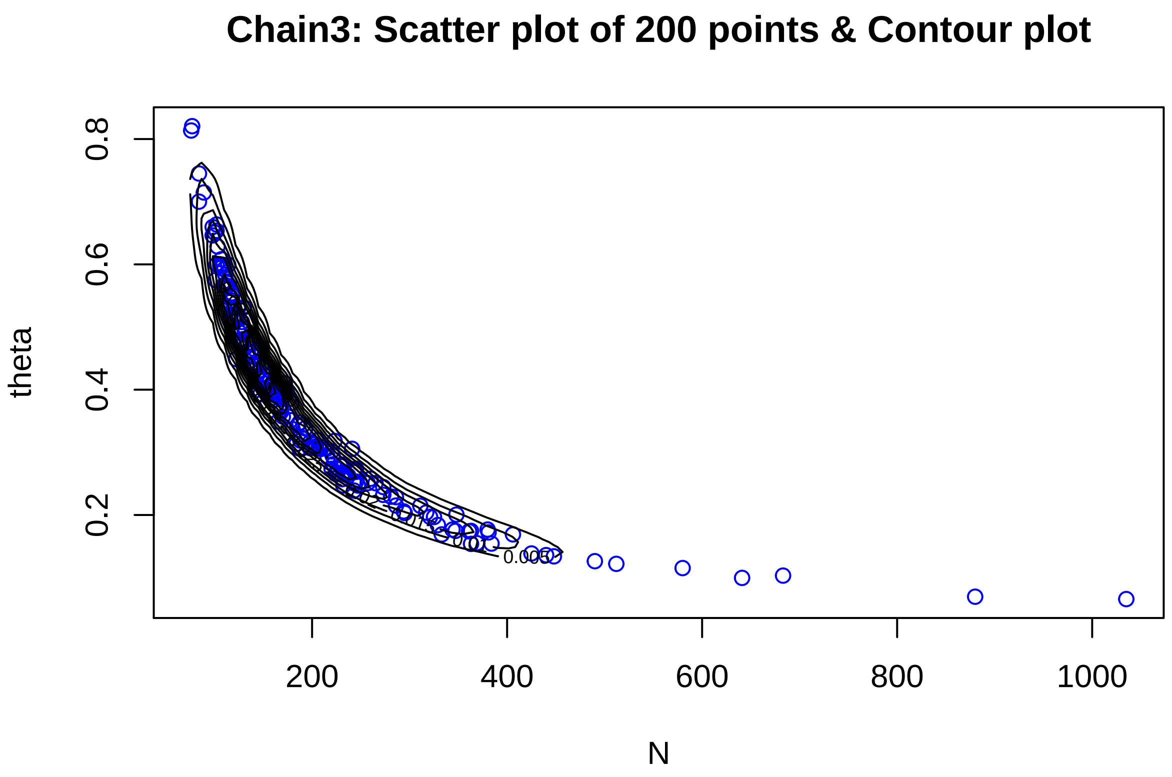
```r
library(MASS)
res <- burn_in(chain1)
ind <- sample(nrow(res), 200)
plot(res[ind, 1], res[ind, 2], col='blue',
     xlab='N', ylab='theta', main='Chain1: Scatter plot of 200 points & Contour plot')
contour(kde2d(res[, 1], res[, 2], n=100), add=TRUE)
```



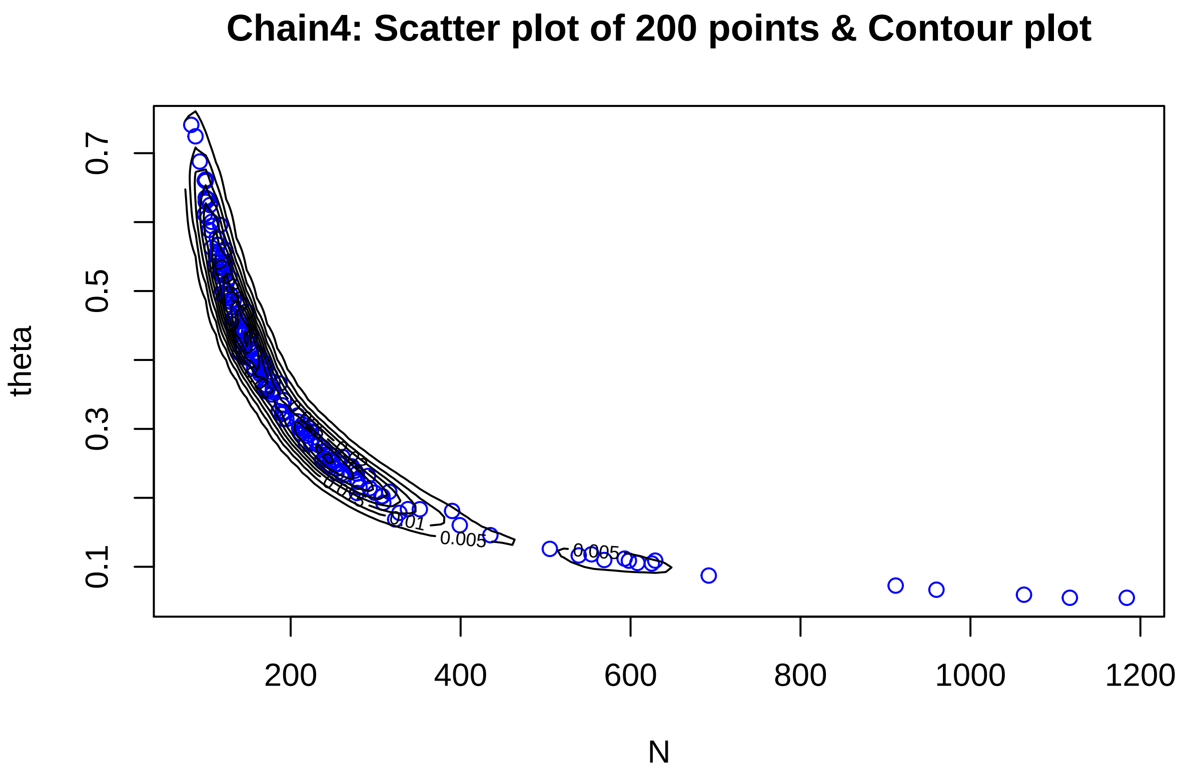
```r
res <- burn_in(chain2)
ind <- sample(nrow(res), 200)
plot(res[ind, 1], res[ind, 2], col='blue',
     xlab='N', ylab='theta', main='Chain2: Scatter plot of 200 points & Contour plot')
contour(kde2d(res[, 1], res[, 2], n=100), add=TRUE)
```



```
res <- burn_in(chain3)
ind <- sample(nrow(res), 200)
plot(res[ind, 1], res[ind, 2], col='blue',
     xlab='N', ylab='theta', main='Chain3: Scatter plot of 200 points & Contour plot')
contour(kde2d(res[, 1], res[, 2], n=100), add=TRUE)
```



```
res <- burn_in(chain4)
ind <- sample(nrow(res), 200)
plot(res[ind, 1], res[ind, 2], col='blue',
     xlab='N', ylab='theta', main='Chain4: Scatter plot of 200 points & Contour plot')
contour(kde2d(res[, 1], res[, 2], n=100), add=TRUE)
```



Posterior prob. estimation for event N > 100.

```
mean(res[, 1] > 100)
```

```
## [1] 0.9357883
```