# SMC: Exercises set II

*Niharika*

*June 17, 2018*

This document provides solution for the Exercises set I given at http://www.it.uu.se/research/systems_and_control/education/2017/smc/homework/SMC2017_exercises2.pdf.

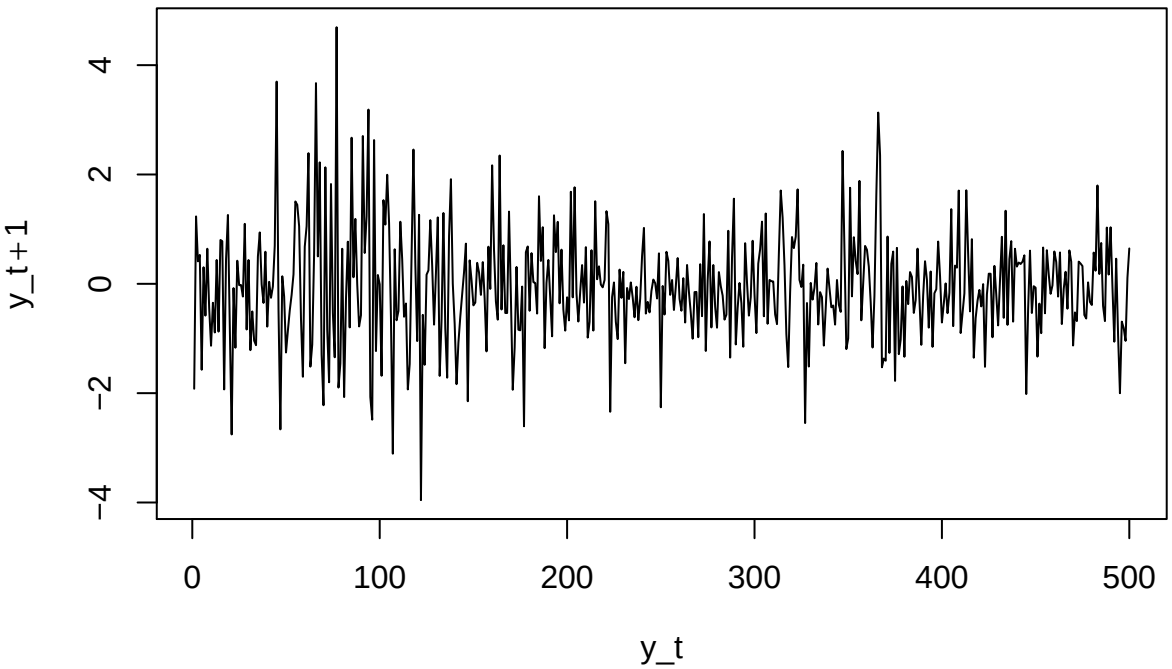
### II.1 Likelihood estimates for the stochastic volatility model

We consider the following stochastic volatility model:

$$
\begin{aligned}
x_t \mid x_{t-1} &\sim N(x_t; \phi x_t, \sigma^2) \\
y_t \mid x_t &\sim N(y_t; 0, \beta^2\ exp(x_t)) \\
\theta &= \{\phi, \sigma, \beta\}
\end{aligned}
$$

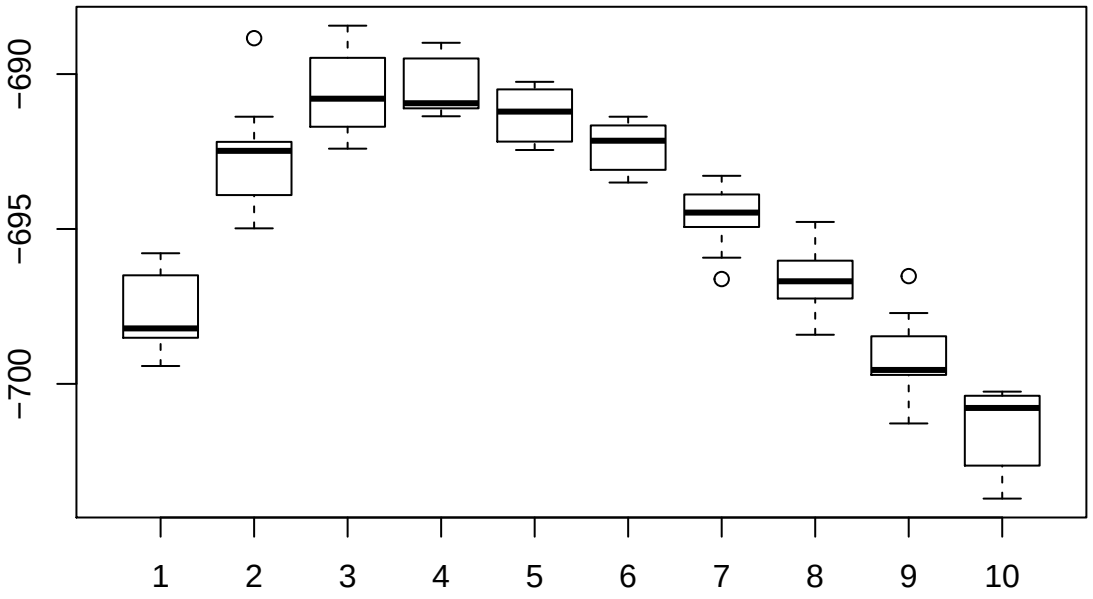
Given the observations $y$, $\theta = \{0.98, 0.16\}$, and $\beta \in (0, 2)$, estimate the likelihood using the bootstrap particle filter with $N = 500$ particles.

```
T <- 500
N <- 500
#Read the observations and plot them
y <- read.csv("seOMXlogreturns2012to2014.csv", header = FALSE)
y <- y[,1]
plot(c(1:T), y, type = "l", xlab = expression(y_t), ylab = expression(y_t+1))
```




```
########## Boot strap particle filter##############
# given parameters
phi <- 0.98
sigma <- 0.16
beta_seq <- seq(0.5, 2, length = 10)

BPF <- function(phi = 0.98, sigma = 0.16, beta = 0.7)
{
```

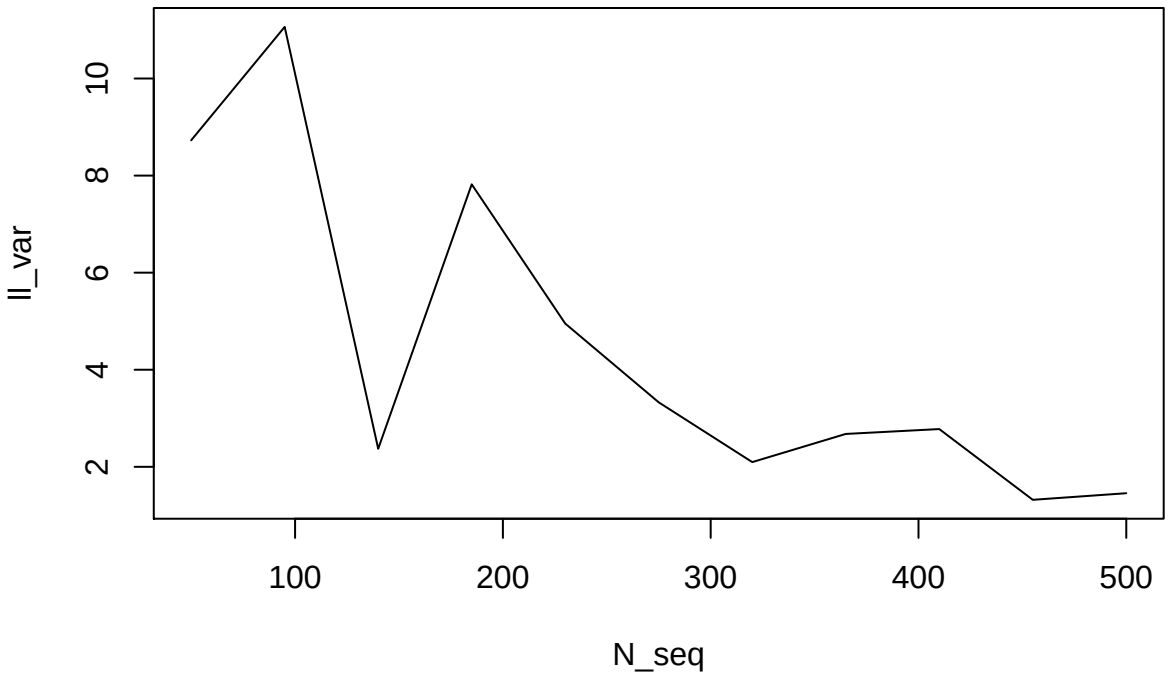
```r
  loglikelihood <- 0
  #state at t=0
  x0 <- rnorm(N, 0, .8)
  x <- matrix(rep(0, T*N), nrow = T)
  # Step 1. Initialize
  x[1, ] <- x0
  m <- rep(0, T)
  m[1] <- 0
  s <- rep(1:N)

  for (t in 2:T) {
    # Step 2. Importance sampling step
    x[t, ] <- 0.98*x[t-1, s] + rnorm(N)*0.16

    #Likelihood
    w.tilde <- dnorm(y[t], mean = 0, sd = beta*exp(x[t,]/2),
                     log = TRUE)

    #Normalize
    max_weight <- max(w.tilde)
    imp_weight <- exp(w.tilde - max_weight)
    w <- imp_weight / sum(imp_weight)

    # accumulate the log-likelihood
    loglikelihood = loglikelihood + max_weight +
      log(sum(imp_weight)) - log(N)

    m[t-1] <- sum(w*x[t,])

    # 3. Resampling step
    s <- sample(1:N, size=N, replace=TRUE, prob=w)

    # Note: resample WHOLE path
    x <- x[, s]
  }
  #return(list(x = x, m = m))
  return(loglikelihood)
}

#result <- BPF()
#plot(c(1:T), result$x[,1], type = "l")
#lines(result$m, col="red")
ll <- matrix(0, 10, length(beta_seq))
for (i in 1:length(beta_seq)) {
  for (k in 1:10) {
    result <- BPF(beta = beta_seq[i])
    ll[k,i] <- result
  }
}

boxplot(ll)
```




Variance decreases exponentially with increasing $N$.

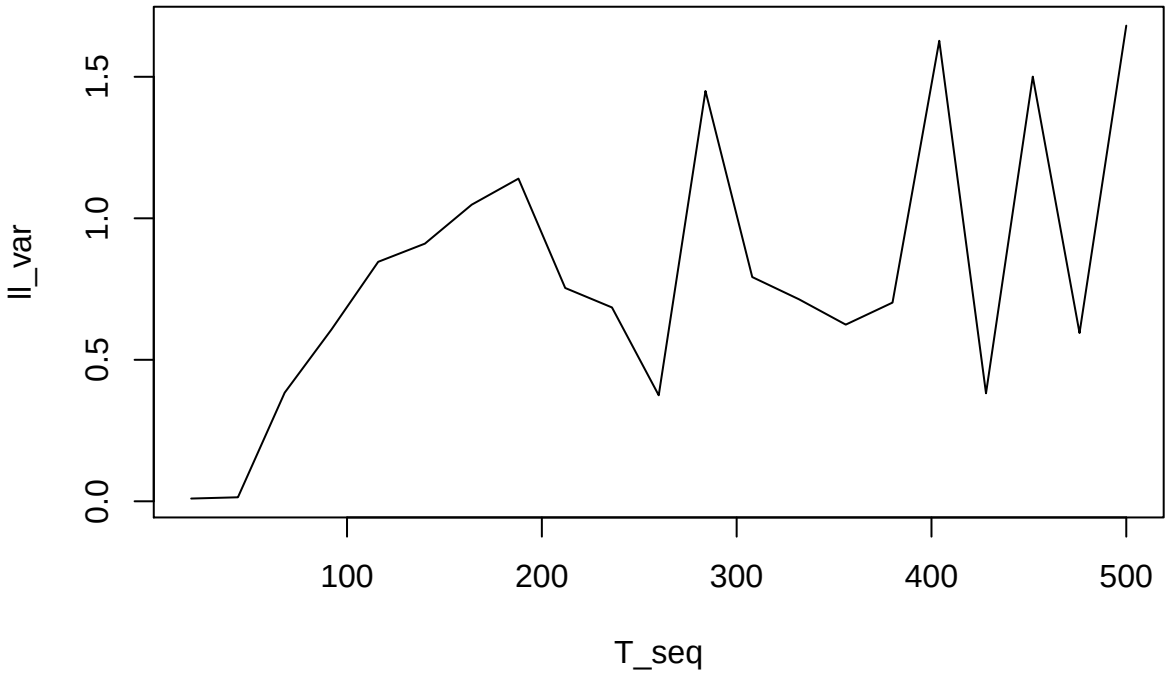
**(b) how $N$ affects the variance of the log-likelihood estimate**

```
N_seq <- seq(50, 500, length = 11)
ll_var <- rep(0, length(N_seq))
ll <- matrix(0, 10, length(N_seq))
for (i in 1:length(N_seq)) {
  N <- N_seq[i]
  for (k in 1:10) {
    result <- BPF(beta = 0.7)
    ll[k,i] <- result
  }
  ll_var[i] <- var(ll[,i])
}

plot(N_seq, ll_var, type = "l")
```



**how $T$ affects the variance of the log-likelihood estimate**

```
T_seq <- seq(20, 500, length = 21)
ll_var <- rep(0, length(T_seq))
ll <- matrix(0, 10, length(T_seq))
for (i in 1:length(T_seq)) {
  T <- T_seq[i]
  for (k in 1:10) {
    result <- BPF(beta = 0.7)
    ll[k,i] <- result
  }
  ll_var[i] <- var(ll[,i])
}

plot(T_seq, ll_var, type = "l")
```

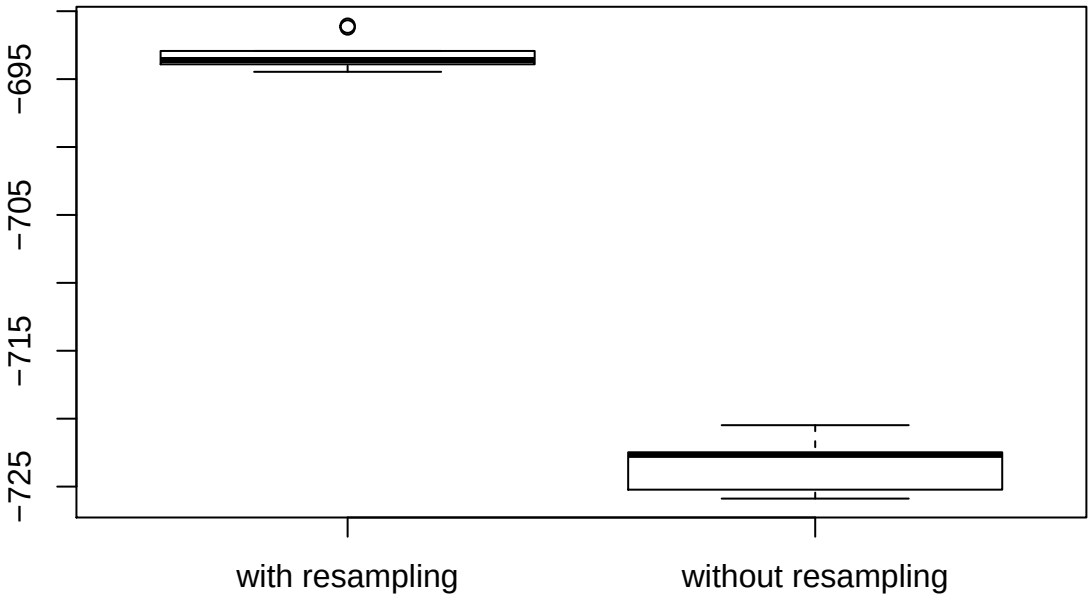



Variance increases almost linearly with increasing $T$.

**(c) Study the influence of resampling on the variance of the estimator**

```
T <- 500
N <- 500
#Read the observations and plot them
y <- read.csv("seOMXlogreturns2012to2014.csv", header = FALSE)
y <- y[,1]
#plot(c(1:T), y, type = "l", xlab = expression(y_t), ylab = expression(y_t+1))

########## Boot strap particle filter#############
# given parameters
phi <- 0.98
sigma <- 0.16
beta_seq <- seq(0.5, 2, length = 10)

BPF <- function(phi = 0.98, sigma = 0.16, beta = 0.7,
                resampling = TRUE)
{
  loglikelihood <- 0
  #state at t=0
  x0 <- rnorm(N, 0, .8)
```

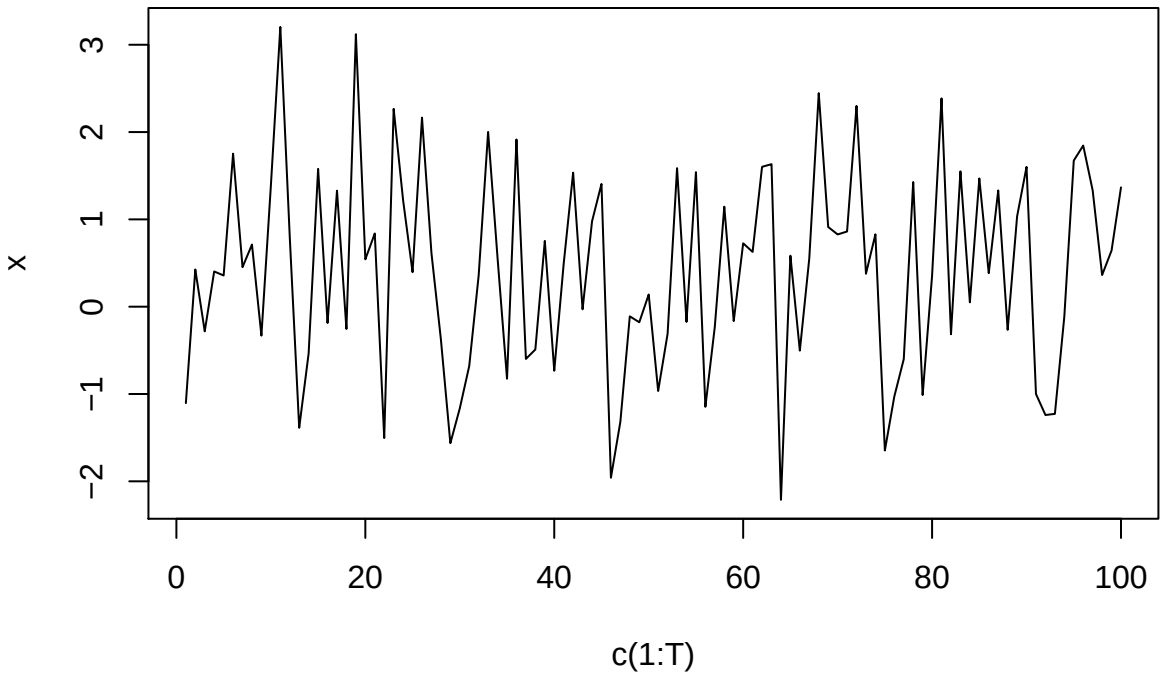
```r
  x <- matrix(rep(0, T*N), nrow = T)
  # Step 1. Initialize
  x[1, ] <- x0
  m <- rep(0, T)
  m[1] <- 0
  s <- rep(1:N)

  for (t in 2:T) {
    # Step 2. Importance sampling step
    x[t, ] <- 0.98*x[t-1, s] + rnorm(N)*0.16

    #Likelihood
    w.tilde <- dnorm(y[t], mean = 0, sd = beta*exp(x[t,]/2),
                     log = TRUE)

    #Normalize
    max_weight <- max(w.tilde)
    imp_weight <- exp(w.tilde - max_weight)
    w <- imp_weight / sum(imp_weight)

    # accumulate the log-likelihood
    loglikelihood = loglikelihood + max_weight +
      log(sum(imp_weight)) - log(N)

    m[t-1] <- sum(w*x[t,])

    # 3. Resampling step
    if(resampling)
      s <- sample(1:N, size=N, replace=TRUE, prob=w)
    else
      s <- sample(1:N, size=N)
    # Note: resample WHOLE path
    x <- x[, s]
  }
  #return(list(x = x, m = m))
  return(loglikelihood)
}

ll <- rep(0, 10)
for (k in 1:10) {
    result <- BPF(beta = 0.7, resampling = TRUE)
    ll[k] <- result
  }

ll1 <- rep(0, 10)
for (k in 1:10) {
  result <- BPF(beta = 0.7, resampling = FALSE)
  ll1[k] <- result
}


ll_df <- cbind(ll, ll1)
colnames(ll_df) <- c("with resampling", "without resampling")
boxplot(ll_df)
```



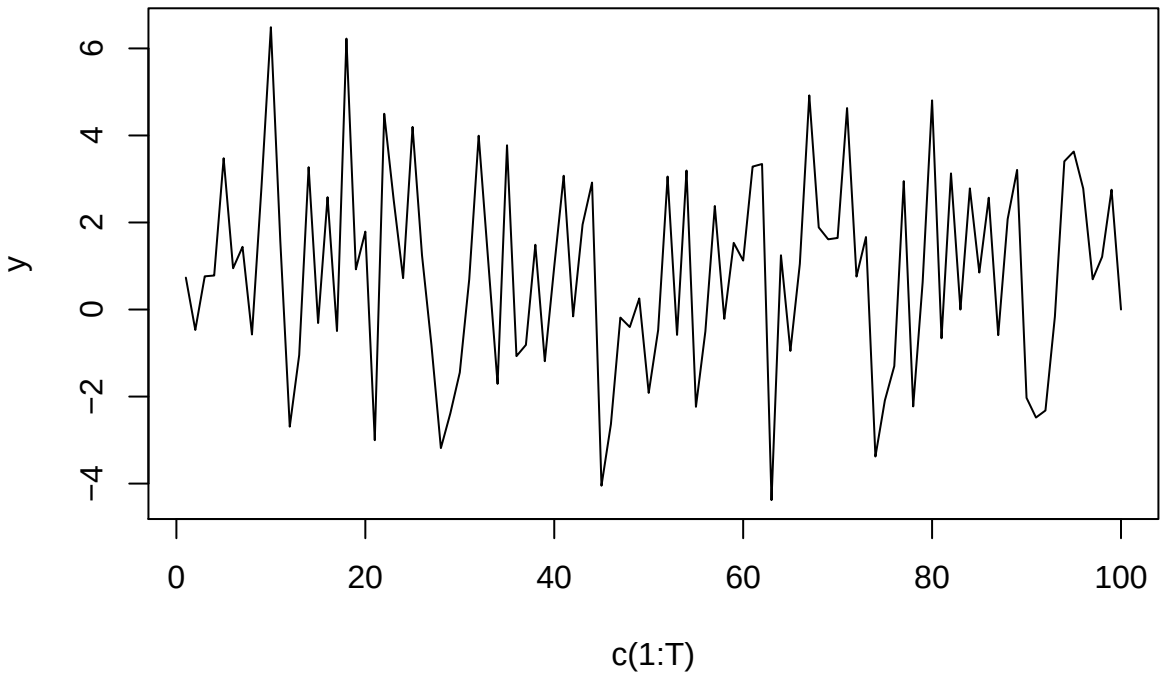

Without resampling the variance is slightly more and log-likelihood is much lower.

### II.2 Fully adapted particle filter

```
T <- 100
N <- 100

x0 <- rnorm(1, 0, 1)
x <- rep(0, T)
y <- rep(0, T)
x[1] <- cos(x0)^2 + rnorm(1)
y[1] <- 2*x[1] + rnorm(1, 0, .1)
for (t in 2:T) {
  x[t] <- cos(x[t-1])^2 + rnorm(1)
  y[t-1] <- 2*x[t] + rnorm(1, 0, .1)
}

plot(c(1:T), x, type = "l")
```

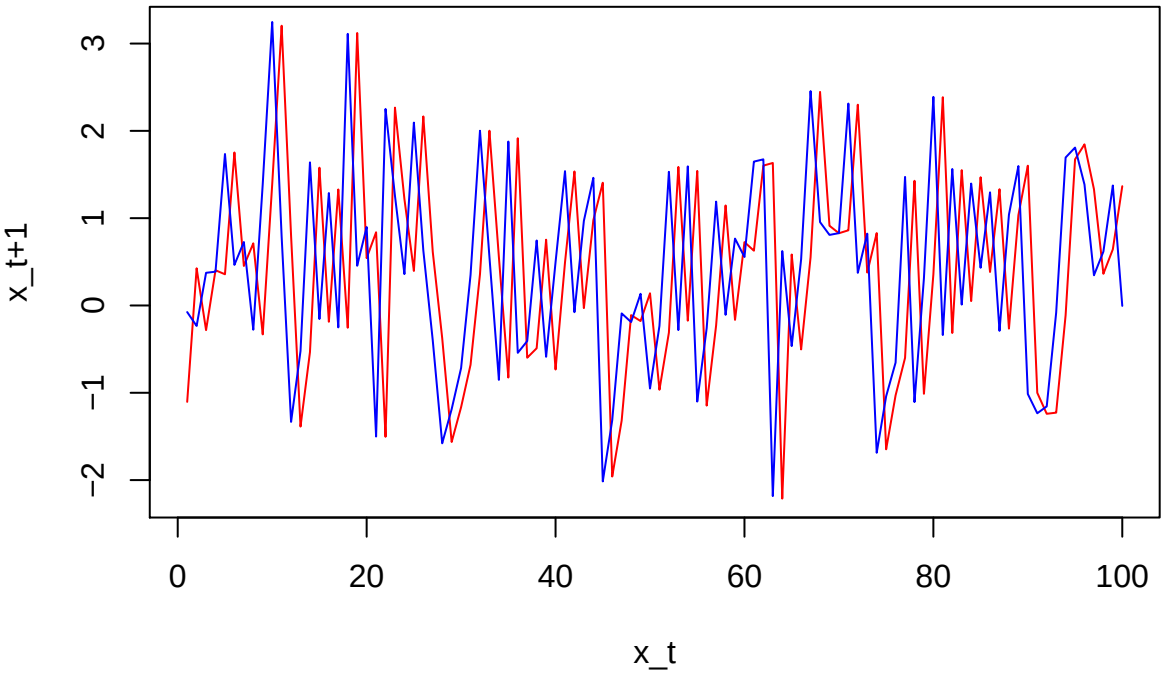


```r
plot(c(1:T), y, type = "l")
```




```r
########## Auxiliary particle filter##############

APF <- function()
{
  loglikelihood <- 0
  #state at t=0
  x0 <- rnorm(N, 0, 1)
  x <- matrix(rep(0, T*N), nrow = T)
  # Step 1. Initialize
  x[1, ] <- x0
  m <- rep(0, T)
  m[1] <- 0
  s <- rep(1:N)

  for (t in 2:T) {
    # compute resampling weights
    resamp_weights = dnorm(y[t], mean = 2*cos(x[t-1, ])^ 2,
                           sd = sqrt(4.01))

    # Normalize the weights
    resamp_weights = resamp_weights / sum(resamp_weights)



    #m[t-1] <- sum(w*x[t,])

    # Resampling step
    s <- sample(1:N, size = N, replace = TRUE, prob = resamp_weights)

    #propogation step
    x[t, ] <- 0.1/sqrt(4.01)*rnorm(N, 0, 1) + (2/4.01) * y[t]  + .01/4.01 * cos(s)^2

  }
  return(x)
}
```

```
simulated_x <- APF()

plot(c(1:T), x, type = "l", col = "red",
     xlab = "x_t", ylab = "x_t+1")
lines(c(1:T), rowMeans(simulated_x), type = "l", col="blue")
```




**II.4 Forgetting**