HW6

Yan Liu

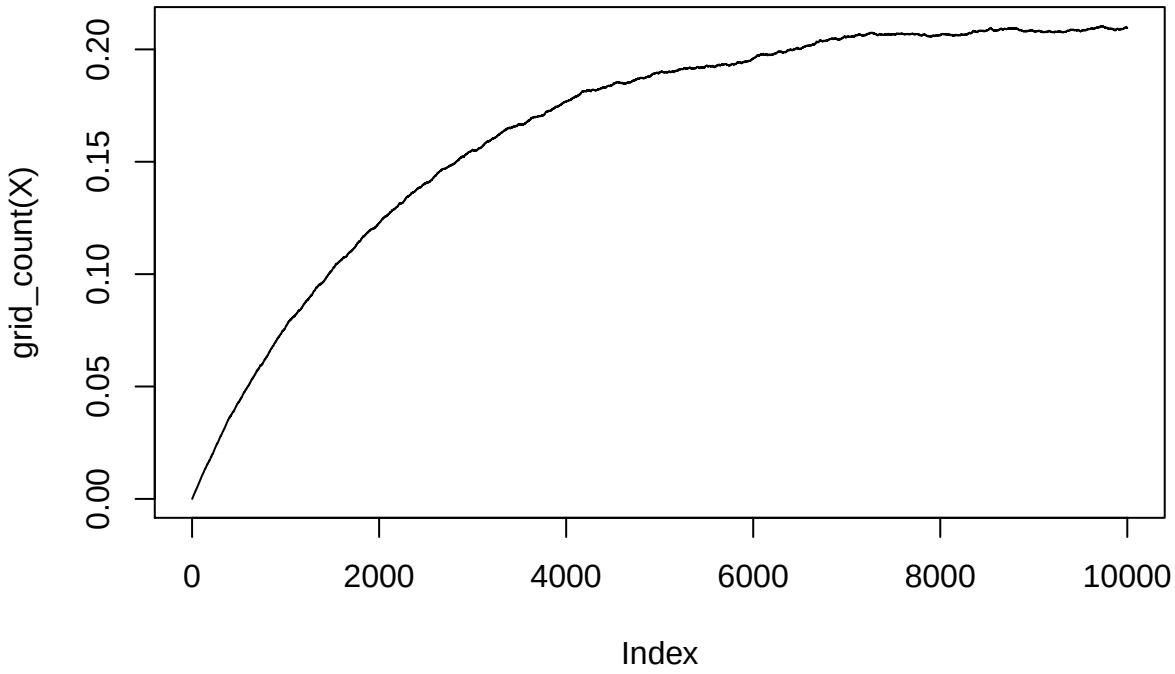
10/13/2019

1 Note for Homework 5: rerunning my code with different initializations was able to reach convergence for both hard EM and soft EM.

a)

```
MH_X <- function(Xt){
    i <- sample(100,1)
    j <- sample(100,1)
    w.prop <- Xt
    w.prop[i,j] <- !w.prop[i,j]
    pass <- hard_core_check(w.prop)
    if (runif(1) < min(1, ifelse(pass, 1, 0))){
      Xt.next <- w.prop
    }else{
      Xt.next <- Xt
    }
    return(Xt.next)
}

hard_core_check <- function(w.prop){
    w.pad <- matrix(data = rep(0, 102*102),
                    nrow = 102,
                    ncol = 102)
    w.pad[2:101, 2:101] <- w.prop
    for (i in 2:101){
      for (j in 2:101){
        if (any(w.pad[i,j] + w.pad[i-1,j] > 1,
                w.pad[i,j] + w.pad[i+1,j] > 1,
                w.pad[i,j] + w.pad[i,j-1] > 1,
                w.pad[i,j] + w.pad[i,j+1] > 1))
        {
          return (FALSE)
          break
        }
      }
    }
    return(TRUE)
}

Nsim <- 10000
X = list(Nsim)
w0 <- matrix(data = rep(0, 100*100),
              nrow = 100,
              ncol = 100)
X[[1]] <- w0
for (t in 2:Nsim){
  X[[t]] <- MH_X(X[[t-1]])
```

```
}

grid_count <- function(X){
  fx <- numeric()
  for (i in 1:length(X)){
    fx[i] <- sum(X[[i]])/100^2
  }
  return(fx)
}

plot(grid_count(X), type = "l")
```



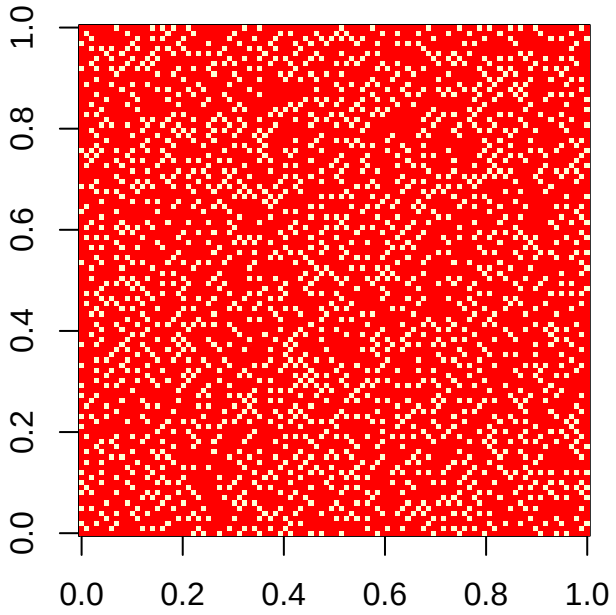
We start to oberve noisy zig zag from 5000 steps.

```
hard_core_sim_X <- function(Nsim){
  X = list(Nsim)
  w0 <- matrix(data = rep(0, 100*100),
               nrow = 100,
               ncol = 100)
  X[[1]] <- w0
  for (t in 2:Nsim){
    X[[t]] <- MH_X(X[[t-1]])
  }
  return(X[[Nsim]])
}
```

```
Nsim <- 5000
Nsample <- 100
h_X = list(Nsample)
for (k in 1:Nsample){
  h_X[[k]] <- hard_core_sim_X(Nsim)
}
```

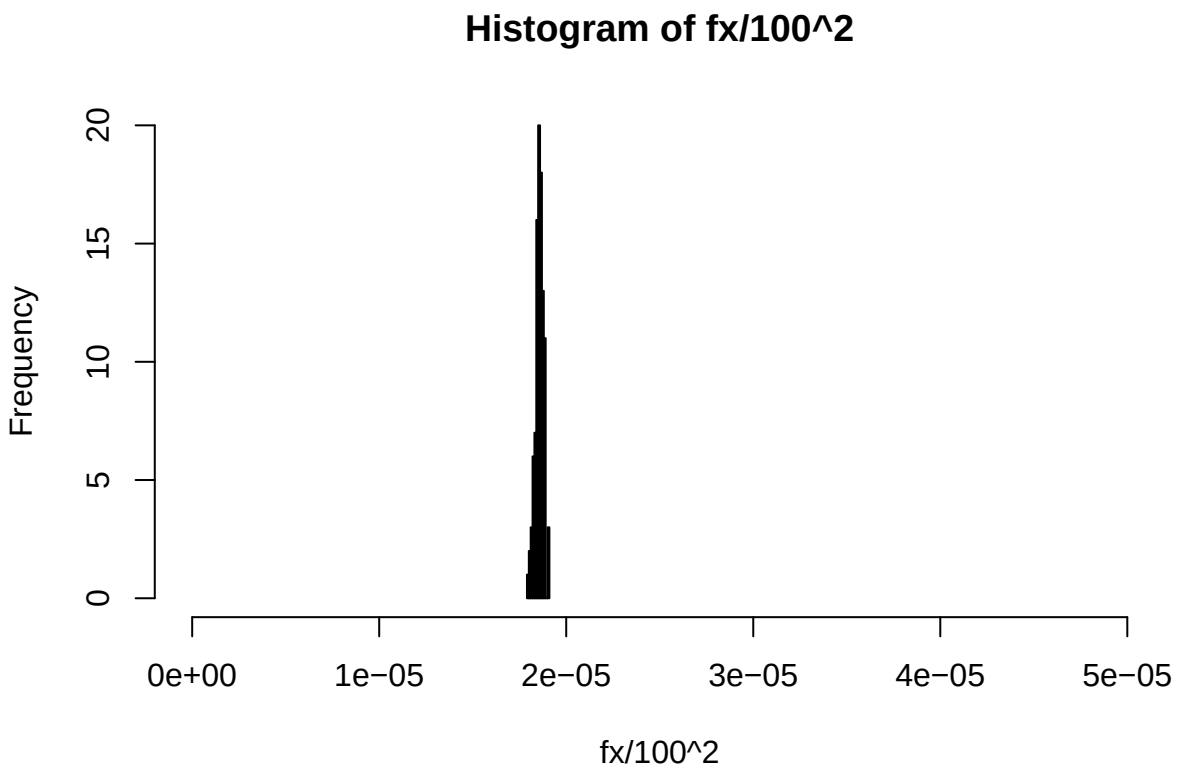
Take a look at a sample configuration

```
par(pty="s")
image(h_X[[2]])
```



Histogram of X

```
fx <- grid_count(h_X)
hist(fx/100^2,
     xlim = c(0,0.5/100^2))
```



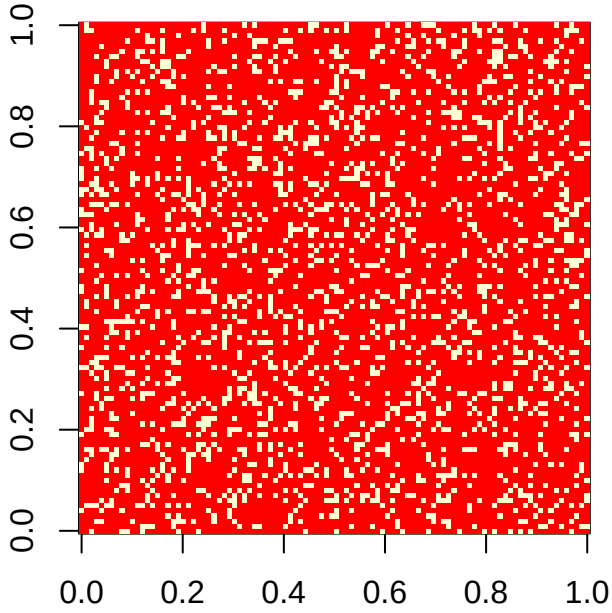
b)

```
MH_Y <- function(Yt){
    i <- sample(100,1)
    j <- sample(100,1)
    w.prop <- Yt
    w.prop[i,j] <- !w.prop[i,j]
    if (runif(1) < sum(w.prop)^2 / (sum(Yt)^2 + sum(w.prop)^2)){
      Yt.next <- w.prop
    }else{
      Yt.next <- Yt
    }
    return(Yt.next)
}

hard_core_sim_Y <- function(Nsim){
  Y = list(Nsim)
  w0 <- matrix(data = rep(0, 100*100),
               nrow = 100,
               ncol = 100)
  Y[[1]] <- w0
  for (t in 2:Nsim){
    Y[[t]] <- MH_Y(Y[[t-1]])
  }
  return(Y[[Nsim]])
}

Nsim <- 5000
```

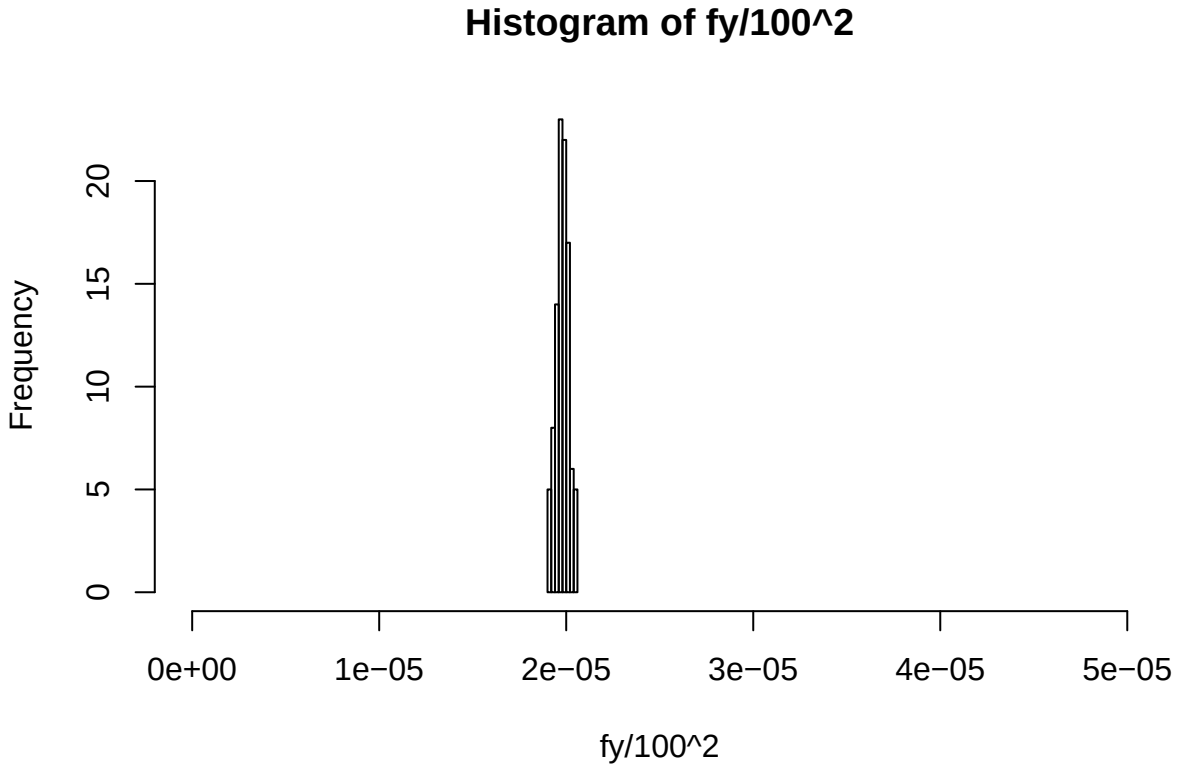
```
Nsample <- 100
h_Y = list(Nsample)
for (k in 1:Nsample){
  h_Y[[k]] <- hard_core_sim_Y(Nsim)
}
```

Take a look at a sample configuration

```
par(pty="s")
image(h_Y[[2]])
```




```
fy <- grid_count(h_Y)
hist(fy/100^2,
     xlim = c(0,0.5/100^2))
```



The distribution of Y seems to be more spread out than the distribution of X.

2.

a)

```
library(dplyr, quietly = TRUE)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
draw <- function(X, v){
  dice <- runif(1)
  return(
          case_when(dice < v[1] ~ X[1],
                    ((dice >= v[1]) & (dice < v[1]+v[2])) ~ X[2],
                    TRUE ~ X[3])
         )
}
```

```
X <- c(1,2,3)
v <- c(1/2, 1/3, 1/6)
draw(X,v)
```

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

Transition Matrix

```
p.prop <- c(.99, .009,.001)
M <- matrix(rep(p.prop,3), nrow =3 ,byrow = T) # transition matrix
M
```

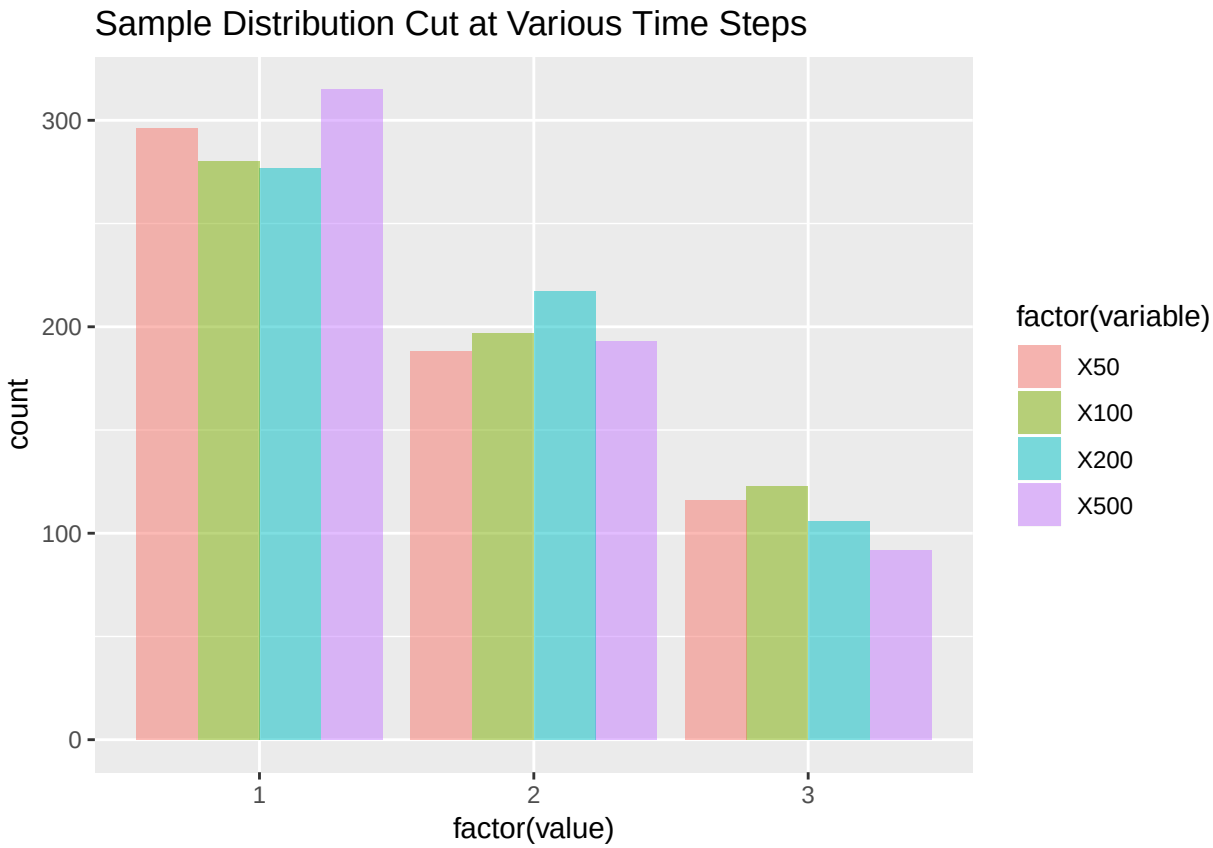
```
##      [,1]  [,2]  [,3]
## [1,] 0.99 0.009 0.001
## [2,] 0.99 0.009 0.001
## [3,] 0.99 0.009 0.001
```

```
MH_small <- function(x, v, p.prop){
    x.prop <- draw(X, p.prop)
    if (runif(1) < min(1, v[x.prop]*M[x.prop, x]/(v[x]*M[x,x.prop]))){
      x.next <- x.prop
    }else{
      x.next <- x
    }
    return(x.next)
}

small_sim <- function(v,Nsim){
  Xs = numeric(Nsim)
  Xs[1]<- draw(X,v)
  for (t in 2:Nsim){
    Xs[t] <- MH_small(Xs[t-1], v, p.prop)
  }
  return(Xs)
}

Nsim <- 500
Nsample <- 600
Xsample <- list(Nsample)
for (i in 1:Nsample){
  Xsample[[i]] <- small_sim(v,Nsim)
}
```

```
library(ggplot2)
library(reshape2)
sample.df <- data.frame(matrix(unlist(Xsample),
                               nrow = Nsample,
                               byrow = TRUE))
sample.m <- melt(sample.df[,c(50, 100, 200,500)])
```

```
## No id variables; using all as measure variables
```

```
ggplot(sample.m, aes(x=factor(value),
                     group = factor(variable))) +
  geom_bar(aes( fill = factor(variable)),
                alpha=0.5,
                position = position_dodge(width = 0.9)) +
  ggtitle("Sample Distribution Cut at Various Time Steps")
```



For 600 samples, the plot above shows the number of 1s, 2s and 3s if we cut the markov chain at 50th step, 100th step, 200th step and 500th step. The desired counts are 300, 200, 100 for 1, 2, 3. We can tell from the plot that at 500 steps the convergence is fairly good.