Homework # 6

1. Consider the hard core model on a 100×100 grid. Let Ω be the set of all configurations and H the set of configurations that do not violate the hard-core restriction (no neighboring 1's). For $w \in \Omega$ let $f(w)$ be the number of positions with a 1 in the grid.

   (a) Let X be the r.v. on Ω,

   $$P(X = w) = \begin{cases} \frac{1}{|H|} & \text{if } w \in H \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

   Using the Metropolis-Hasting algorithm, write a sampler for X. Show a sample configurations using the **image** function in R (or equivalent in Python). Using your sampler, generate a histogram for $f(X)/100^2$, the fraction of sites with a 1 under the uniform distribution X. To decide how long to run the MH-algorithm before sampling, plot $f(X)$ as a function of the time step of your chain. If plotted on a long enough time scale, the plot should look noisy. Once you decide how long to run the chain, run the chain many times to produce a histogram. (Each time you sample from the Metropolis-Hastings algorithm you have to rerun the chain.)

   (b) Let Y be the r.v. on Ω defined by $P(Y = w) = \alpha(f(w))^2$, where α is a normalizing constant that makes the probabilities sum to 1. Use your sampler to generate a histogram for $f(Y)/100^2$. Compare to part a).

2. Suppose we would like to sample the r.v. X, where $X \in \{1, 2, 3\}$ and $P(X = i) = v_i$ for $i = 1, 2, 3$ and where $v = (v_1, v_2, v_3) = (1/2, 1/3, 1/6)$.

   (a) Write a function to do this using a single draw from a uniform r.v. You may not use your languages discrete random sampler (e.g. sample or sample.int in R). We discussed how to do this at the very beginning of the semester.

   (b) Now use the Metropolis-Hastings algorithm to write a sampler for X. Given that we are in state i, let the proposal be 1 with probability .99, 2 with probability .009 and 3 with probability .001. (Notice, in this case the proposal doesn't depend on the current state, but this is not always the case.) Write a function in R/Python to implement the Metropolis-Hastings algorithm. Your function simulates a Markov chain, write down the transition probability matrix of the markov chain. (Typically, we don't write this matrix down, but in this simple setting it is a good exercise.). How long do you have to run the chain to be close to the stationary distribution?