Project 2

# -\*- coding: utf-8 -\*-  
"""  
Created on Wed Nov 23 21:08:43 2016  
  
@author: Kanak  
"""  
  
  
import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
import matplotlib.pyplot as pltbar  
from mpl\_toolkits.mplot3d import Axes3D  
from math import exp  
from scipy.stats import chi2  
from scipy.stats import binom  
from scipy.special import gamma  
  
  
########################################################################  
#  
# Problem 1  
#  
########################################################################  
  
burn = 500  
a = 5  
b = 10  
n = 25  
  
def ncr(n, r):  
 r = min(r, n-r)  
 if r == 0: return 1  
 numer = 1  
 for i in range(n, n-r, -1):  
 numer \*= i  
 denom = 1  
 for i in range(1, r+1):  
 denom \*= i  
 return numer//denom  
   
   
def prob\_func(data, a, b, n):  
 length = len(x)  
 for i in range(0, length):  
 data[i].append(ncr(n, data[i][0])\*(data[i][1]\*\*(data[i][0]+a-1)  
 )\*((1-data[i][1])\*\*(n-data[i][0]+b-1)))  
   
 return(data)   
   
def gib\_sample(a, b, n, sample\_size = 1, burn = 500):  
 chain = sample\_size + burn  
 x = np.random.binomial(n, 0.5, size=None)  
 y = np.random.beta(x + a, n - x +b, size=None)  
 sam = []  
 sam.append([x, y])  
   
 for \_ in range(1, chain):  
 x = np.random.binomial(n, y, size=None)  
 y = np.random.beta(x+a, n-x+b, size=None)  
 sam.append([x, y])  
 return(sam[burn:chain])  
   
   
def marginalpdf(x, n, a, b):  
 fx = ncr(n, x)\*gamma(a+b)\*gamma(x+a)\*gamma(n-x+b)/(gamma(a)\*gamma(b)\*gamma(a+b+n))  
 return fx   
  
x = gib\_sample(sample\_size = 10000, a = a, b = b, n = n)  
   
xp = pd.DataFrame(prob\_func(x, a, b, n))  
xp.columns = ["x", "y", "p"]  
  
fig = plt.figure(figsize=(12,10))  
ax = fig.add\_subplot(111, projection='3d')  
  
z = xp["p"]  
x = xp["x"]  
y = xp["y"]  
  
ax.scatter(x, y, z, c='r', marker='.')  
  
ax.set\_xlabel('X')  
ax.set\_ylabel('Y')  
ax.set\_zlabel('Probability')  
  
plt.show()  
plt.clf()  
plt.cla()  
plt.close()  
  
bp = []  
for i in range(0, n):  
 bp.append(np.mean(binom.pmf(i, n, xp["y"])))  
  
fig, axs=pltbar.subplots(1,2, figsize=(10, 6), sharex='col', sharey='row')  
width = 1  
axs[0].bar(range(0, n), bp, width, color="blue")  
  
bpm = []  
for i in range(0, n):  
 bpm.append(marginalpdf(i, n, a, b))  
  
axs[1].bar(range(0, n), bpm, width, color="blue")  
pltbar.show()  
pltbar.clf()  
pltbar.cla()  
pltbar.close()