HW3 Problem 2

(a) The code is shown below. The rejection sampling is used.

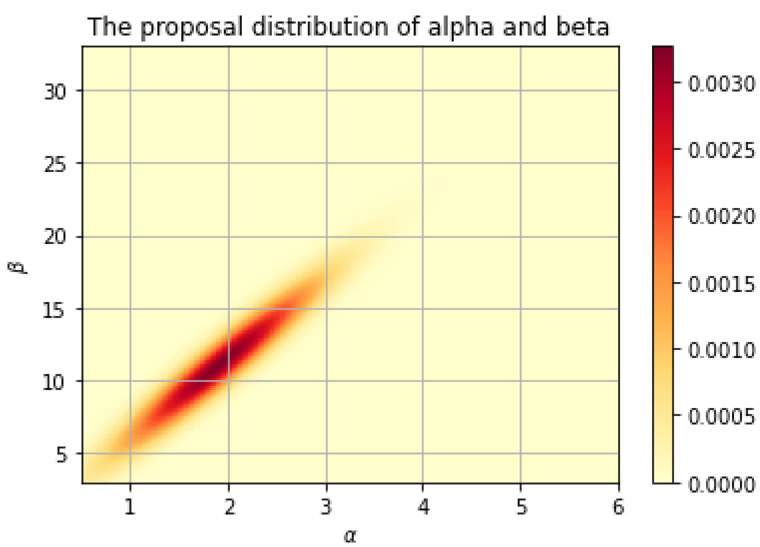
```
import numpy as np
from scipy import stats
# from scipy.stats import beta
from scipy.special import gammaln

%matplotlib inline
import matplotlib.pyplot as plt
```

```
# rat data (BDA3, p. 102)
y = np.array([
     0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  1,  1,  1,
     1,  1,  1,  1,  1,  2,  2,  2,  2,  2,  2,  2,  2,  2,  1,  5,  2,
     5,  3,  2,  7,  7,  3,  3,  2,  9, 10,  4,  4,  4,  4,  4,  4,  4,
    10,  4,  4,  4,  5, 11, 12,  5,  5,  6,  5,  6,  6,  6,  6, 16, 15,
    15,  9,  4
])
n = np.array([
    20, 20, 20, 20, 20, 20, 20, 19, 19, 19, 19, 18, 18, 17, 20, 20, 20,
    20, 19, 19, 18, 18, 25, 24, 23, 20, 20, 20, 20, 20, 20, 10, 49, 19,
    46, 27, 17, 49, 47, 20, 20, 13, 48, 50, 20, 20, 20, 20, 20, 20, 20,
    48, 19, 19, 19, 22, 46, 49, 20, 20, 23, 19, 22, 20, 20, 20, 52, 46,
    47, 24, 14
])
```

```
# compute the marginal posterior of alpha and beta in the hierarchical model in a grid
A = np.linspace(0.5, 6, 100)
B = np.linspace(3, 33, 100)
# calculated in logarithms for numerical accuracy
lp = (
  - 5/2 * np.log(A + B[:,None])
  + np.sum(
        gammaln(A + B[:,None])
      - gammaln(A)
      - gammaln(B[:,None])
      + gammaln(A + y[:,None,None])
      + gammaln(B[:,None] + (n - y)[:,None,None])
      - gammaln(A + B[:,None] + n[:,None,None]),
        axis=0
    )
)
# subtract the maximum value to avoid over/underflow in exponentation
lp -= lp.max()
p = np.exp(lp)
# normalize
p = p/p.sum()

# use rejection sampling
points_x = []
```

```python
points_y = []
g_mu = [11, 1.9]
g_cov = [[16, 2.7], [2.7, 0.5]]
g = np.zeros([B.shape[0], A.shape[0]])
for j in range(B.shape[0]):
    for i in range(A.shape[0]):
        temp = [B[j], A[i]]
        points_x.append(B[j])
        points_y.append(A[i])
        g[j, i] = stats.multivariate_normal.pdf(temp, g_mu, g_cov)

g = g/g.sum()
```

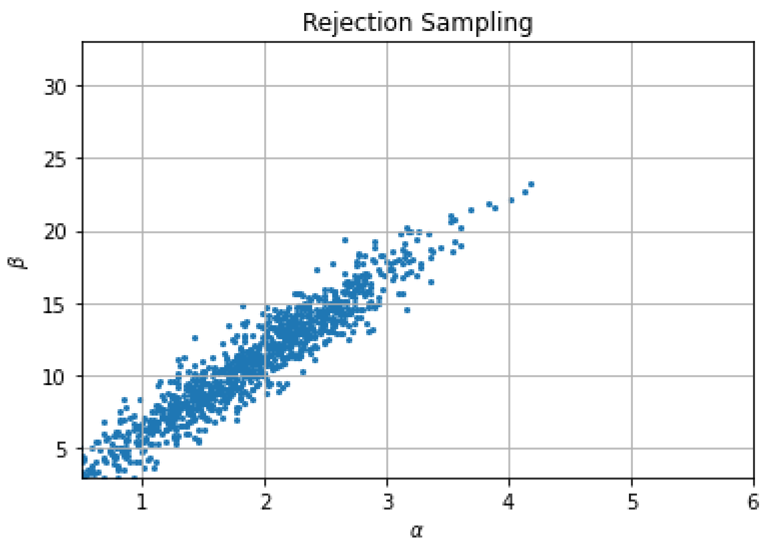
```python
plt.imshow(
    g,
    origin='lower',
    aspect='auto',
    extent=(A[0], A[-1], B[0], B[-1]),
    cmap = 'YlOrRd'
)
plt.xlabel(r'$\alpha$')
plt.ylabel(r'$\beta$')
plt.title('The proposal distribution of alpha and beta')
plt.grid('off')
plt.colorbar()
plt.show()
```




```python
M = np.max(p/g)
g *= M
# randomly generate points
nsamp = 1000
sam_x, sam_y = np.random.multivariate_normal(size=nsamp, mean=g_mu, cov=g_cov).T

points_x = np.asarray(points_x)
points_y = np.asarray(points_y)
# the index of the nearest point
```

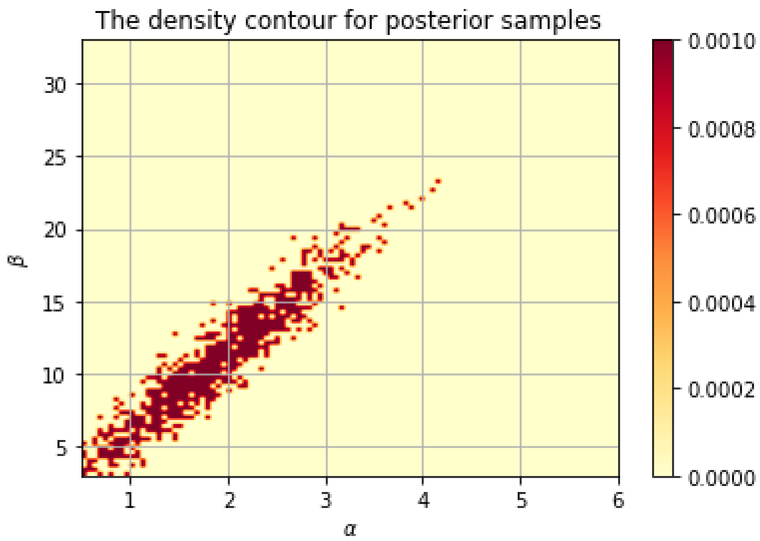
```
idx = np.argmin((points_x[:, None] - sam_x)**2 + (points_y[:, None] - sam_y)**2, axis=0)
pos_B, pos_A = divmod(idx, A.shape[0])

acc = np.random.rand(nsamp) * g[pos_B, pos_A]
acc = acc < p[pos_B, pos_A]
```

```
samp_A = A[pos_A]
samp_B = B[pos_B]
# add random jitter
samp_A += (np.random.rand(nsamp) - 0.5) * (A[1]-A[0])
samp_B += (np.random.rand(nsamp) - 0.5) * (B[1]-B[0])

plt.scatter(samp_A, samp_B, 10, linewidth=0)
plt.xlim([A[0], A[-1]])
plt.ylim([B[0], B[-1]])
plt.xlabel(r'$\alpha$')
plt.ylabel(r'$\beta$')
plt.title('Rejection Sampling')
plt.grid('on')
plt.show()
```

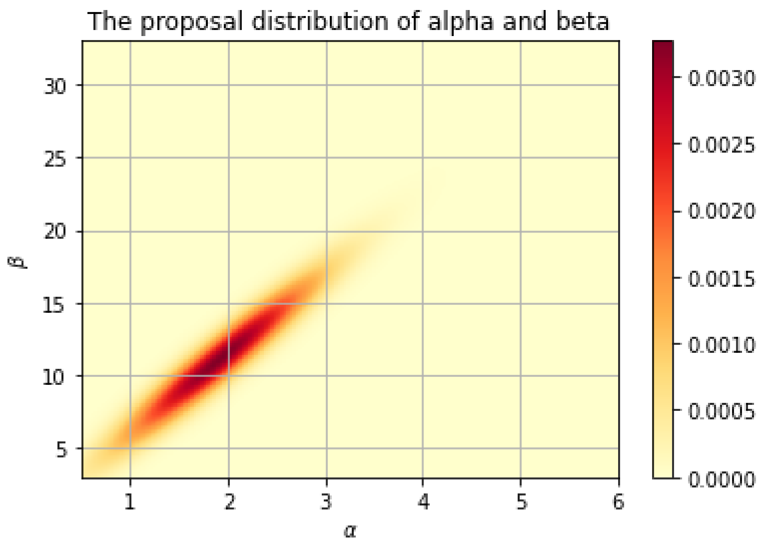



```
new_mat = np.zeros([B.shape[0], A.shape[0]])
samp_A = A[pos_A]
samp_B = B[pos_B]
new_mat[pos_B, pos_A] = 1
new_mat /= len(pos_A)

plt.imshow(
    new_mat,
    origin='lower',
    aspect='auto',
    extent=(A[0], A[-1], B[0], B[-1]),
    cmap = 'YlOrRd'
)
plt.xlabel(r'$\alpha$')
plt.ylabel(r'$\beta$')
plt.title('The density contour for posterior samples')
```

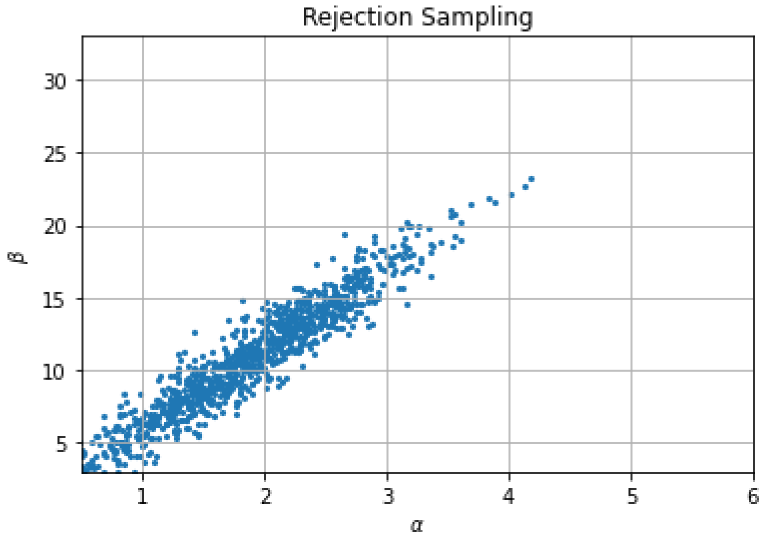
```
plt.grid('off')
plt.colorbar()
plt.show()
```



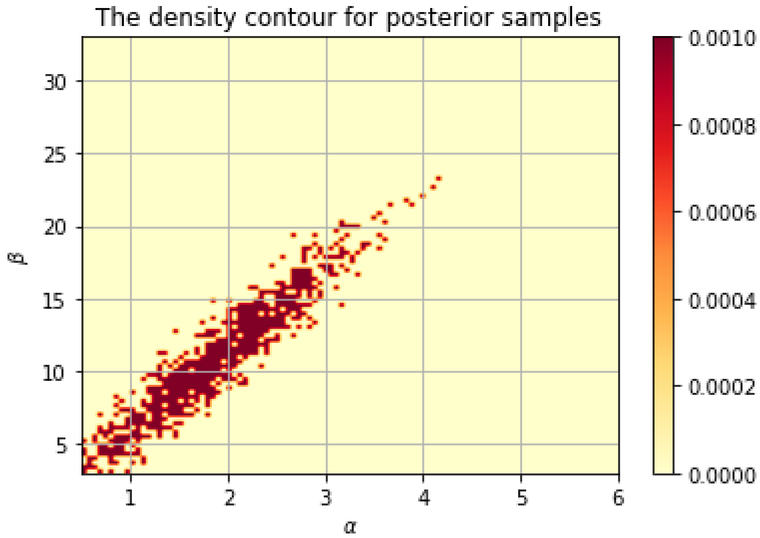
(b) The plot below is the proposal distribution.



The figures below are the posterior sample by using rejection sampling and its corresponding density contour.



Rejection Sampling
30
25
20
15
10
5
1
2
3
4
5
6
α
β



The density contour for posterior samples
30
25
20
15
10
5
1
2
3
4
5
6
α
β
0.0010
0.0008
0.0006
0.0004
0.0002
0.0000

Problem 3

Both Gibbs sampling and Metropolis-Hastings algorithm updates their parameter θ ~~at~~. ~~Suppose~~ But their acception rates are different.

Suppose the parameter at time t-1 is θ^{t-1}, which can be a scalar or a vector. The parameter at time t is θ^{t}. The proposal distribution is $J(\theta^{t}|\theta^{t-1})$.

Then, according to Metropolis-Hastings algorithm, we will accept θ^{t} with ~~probabilit~~ probability $\min(r,1)$

$$\text{where } r = \frac{P(\theta^{t}|y)\cdot J(\theta^{t-1}|\theta^{t})}{P(\theta^{t-1}|y)\cdot J(\theta^{t}|\theta^{t-1})}$$

However, in Gibbs sampling, we will always accept θ^{t} $(r=1)$. We assume that the proposal distribution is the same as the conditional distribution, which means that $P(\theta^{t}|y)=J(\theta^{t}|\theta^{t-1})$ for all θ^{t-1} and θ^{t}. In addition, Gibbs sampling will only update one component in θ even if θ is a vector, while it's not the case for Metropolis-Hastings.

So Gibbs Sampling is a special case of Metropolis-Hastings algorithm.

Problem 4

(a) $Y=0$ represents the white ball and $Y=1$ represents the black ball. $P(Y_1=0,Y_2=1)=\frac{1}{2}\times\frac{1}{2}=\frac{1}{4}=P(Y_1=1,Y_2=0)$

So observations y_1 and y_2 are exchangable.

The balls is put back, $P(Y_2|Y_1)=P(Y_2)$, they are independent.

We can act as if the two observations are independent.

(b) Suppose there are k black balls.

$$P(Y_1=0,Y_2=1)=\frac{n-k}{n}\cdot\frac{k}{n-1}=\frac{k(n-k)}{n(n-1)}$$

$$P(Y_1=1,Y_2=0)=\frac{k}{n}\cdot\frac{n-k}{n-1}=\frac{k(n-k)}{n(n-1)}$$

$$P(Y_1=0,Y_2=1)=P(Y_1=1,Y_2=0)$$

~~It's exchan~~ The observations are exchangable.

$$P(Y_2=1|Y_1=0)=\frac{k}{n-1}\quad,\quad P(Y_2=1)=\frac{k}{n}$$

$P(Y_2=1|Y_1=0)\neq P(Y_2=1)$. The observations are dependent.

If n is not so large, $P(Y_2|Y_1)$ differs a lot from $P(Y_2)$. We can't act as if they are independent.

(c) Similar to (b), $P(Y_1=0,Y_2=1)=P(Y_1=1,Y_2=0)=\frac{k(n-k)}{n(n-1)}$. The observations are exchangable.

We have $P(Y_2=0)=\frac{n-k}{n}$, $P(Y_2=1)=\frac{k}{n}$ $P(Y_2=0|Y_1=0)=\frac{n-k-1}{n-1}$, $P(Y_2=0|Y_1=1)=\frac{k}{n-1}$

$P(Y_2=1|Y_1=0)=\frac{k}{n-1}$, $P(Y_2=1|Y_1=1)=\frac{k-1}{n-1}$

The observations are dependent. as $P(Y_2=0)\neq P(Y_2=0|Y_1=0)$.

Since $n, k, n-k$ are all large, $P(Y_2|Y_1)\approx P(Y_2)$ in all cases. We can act as if they are exchangable.

Problem 5

~~Suppose X, Y are two and two independent and identically of~~

$\forall i,j \in [1, J]$, $i,j \in \mathbb{Z}$. θ_i, θ_j are independent and identically distributed random variables.

$$Cov(\theta_i, \theta_j) = E(\theta_i\theta_j) - E(\theta_i)E(\theta_j)$$

$$= E[E(\theta_i\theta_j|\phi)] - E[E(\theta_i|\phi)] \cdot E[E(\theta_j|\phi)]$$

$$= E[E(\theta_i\theta_j|\phi)] - E[E(\theta_i|\phi) \cdot E(\theta_j|\phi)] + E[E(\theta_i|\phi) \cdot E(\theta_j|\phi)] - E[E(\theta_i|\phi)] \cdot E[E(\theta_j|\phi)]$$

$$= E[E(\theta_i\theta_j|\phi) - E(\theta_i|\phi)E(\theta_j|\phi)] + Cov[E(\theta_i|\phi), E(\theta_j|\phi)]$$

$$= E[cov(\theta_i\theta_j|\phi)] + Cov[E(\theta_i|\phi), E(\theta_j|\phi)]$$

Since θ_i, θ_j are independent and identically distributed random variables,

$Cov(\theta_i\theta_j|\phi) = Cov(\theta_i^2|\phi) = Var(\theta_i|\phi)$ and $E(\theta_i|\phi) = E(\theta_j|\phi)$

So $Cov(\theta_i, \theta_j) = E[Var(\theta_i|\phi)] + Var[E(\theta_i|\phi)]$

As $Var(\theta_i|\phi) \geq 0 \ \forall \phi$, $E[Var(\theta_i|\phi)] \geq 0$. And $Var[E(\theta_i|\phi)] \geq 0$.

Thus $Cov(\theta_i, \theta_j) \geq 0$