Homework 4

Chuan Lu
13300180056

September 21, 2016

Problem 1.
Simulate the random variable X and Y, and estimate E(X) and E(XY).

Proof.

1 Answer

1. Randomly choose x_0 and y_0, which obey U(0, B);
2. For i in 1:n,
2.1 Sample x_{i+1} from $f(x|y=y_i) = C(y_i)e^{-y_i x}$;
2.2 Sample y_{i+1} from $f(y|x=x_{i+1}) = C(x_{i+1})e^{-x_{i+1}y}$;
3. Choose the last $\frac{n}{2}$ samples as a simulation of X and Y;
4. $E(X) = \frac{2}{n}\Sigma x_i$, for x in the samples mentioned above; $E(XY) = \frac{4}{n^2}\Sigma_i\Sigma_j x_i y_j$, for x and y in the samples in step 3; □

Problem 2.
Estimate (1)$E(X_1+2X_2+3X_3|X_1+2X_2+3X_3>15)$, (2)$E(X_1+2X_2+3X_3|X_1+2X_2+3X_3<1)$ with simulation methods.

Proof.

2 Answer

2.1 subproblem1

□

Problem 3.
Estimate X, Y, Z and E(XYZ) with the distribution given.

Proof.

3 Answer

3.1 subproblem1

1. We have $f(x|y,z) = \frac{f(x,y,z)}{\int_0^\infty f(x,y,z)dx} = (ay+bz+1)e^{-(x+axy+bxz)}$.
Equally, $f(y|x,z) = (ax+cz+1)e^{-(y+axy+cyz)}$, $f(z|x,y) = (bx+cy+1)e^{-(z+bxz+cyz)}$.
2. Randomly select the initial data x_0, y_0, z_0, which are all larger than 0.
3. For i in 0:n,
3.1 Sample x_{i+1} from $f(x|y_i,z_i) = (ay_i+bz_i+1)e^{-(1+ay_i+bz_i)x}$;
3.2 Sample y_{i+1} from $f(y|x_{i+1},z_i) = (ax_{i+1}+cz_i+1)e^{-(1+ax_{i+1}+cz_i)y}$;
3.3 Sample z_{i+1} from $f(z|x_{i+1},y_{i+1}) = (bx_{i+1}+cy_{i+1}+1)e^{-(1+bx_{i+1}+cy_{i+1})z}$;
4. Choose the last half as an estimation of X, Y and Z.

3.2 subproblem2

1. Sample X, Y, Z with the process above, in which a, b, c replaced by 1;
2. Estimate $E(XYZ) = \frac{8}{n^3} \sum \sum \sum xyz$, in which x, y, z are the samples chosen.

3.3 the code of subproblem2 is as follows.

```
######### Gibbs Sampling for problem 3 ###########
sample3 = function(param1, param2) {
  lambda = 1/(param1 + param2 + 1);
  ##### Using inverse transform algorithm to generate the distribution:
  ##### f(x|lambda) = lambda*exp(-lambda*x);
  u = runif(1);
  x = -lambda*log(lambda*u);
  return(x);
}

gibbs_sampling3 = function(n, init_param) {
  ##### Initialize parameters;
  x = rep(0, n);
  y = rep(0, n);
  z = rep(0, n);
  x[1] = init_param[1];
  y[1] = init_param[2];
  z[1] = init_param[3];

  ##### Iterative Sampling;
  for(i in 2:n) {
    x[i] = sample3(y[i-1], z[i-1]);
    y[i] = sample3(x[i], z[i-1]);
    z[i] = sample3(x[i], y[i]);
  }
  rlist = list(x, y, z);
  return(rlist);
}

estimate3 = function(n) {
  res = gibbs_sampling3(2*n, c(1, 1, 1));
  X = res[[1]];
  Y = res[[2]];
  Z = res[[3]];
  X_sample = X[(n+1):(2*n)];
  Y_sample = Y[(n+1):(2*n)];
  Z_sample = Z[(n+1):(2*n)];

  sum = sum(X_sample)*sum(Y_sample)*sum(Z_sample);
  return(sum/(n^3));
}

estimate3(100000)
```

3.4 the result of subproblem2 is as follows

```
> estimate3(100000)
[1] 0.4532435
```

$\square$

Problem 4.
Estimate E(X), E(Y) and E(N) with the distribution given.

Proof.

4 Answer

4.1 the algorithm

First of all, one can see that the r.v. Y should be between 0 and 1.
Consequently,

$$f(X|Y,N) \propto C_N^x Y^x (1-Y)^{nx},$$
$$f(Y|X,N) \propto Beta(X+\alpha, NX+\beta),$$
$$f(N|X,Y) \propto C_N^X (1-Y)^{NX} \frac{\lambda^N}{N!}$$

We can use Gibbs Sampling to generate these random sequences.

4.2 the code is as follows

```r
##### Gibbs Sampling for Problem 4 #####
sample4_x = function(y, n) {
  probs = rep(0, n+1);
  c = 1; # C_{n}^{x};
  ycon = 1; # y^x * (1-y)^(nx);
  for(i in 1:(n+1)) {
    probs[i] = c * ycon;
    c = c*(n-i)/(i+1);
    ycon = ycon * y * (1-y)^n;
  }
  normalized_probs = rep(0, n+1);
  sums = 0;
  total_sum = sum(probs);
  u = runif(1);

  for(i in 1:(n+1)) {
    sums = sums + probs[i];
    normalized_probs[i] = sums/total_sum;
    if(u < normalized_probs[i]) {
      x = i-1;
      break;
    }
  }
  return(x);
}

sample4_y = function(x, n, alpha, beta) {
  y = rbeta(1, x+alpha, n*x+beta);
  return(y);
}


sample4_n = function(x, y, lambda) {
  ### Use Acceptance-Rejection Method to generate N ###
```

```
  if(x == 0){
    n_max = 0;
  }
  else {
    n_max = floor(x/(1-(1-y)^x));
  }
  c = choose(n_max, x)*(1-y)^(n*x);
  # print(c(n_max, x, y, n));
  while(1) {
    pois = -1;
    while(pois < x) {
      pois = rpois(1, lambda);
    }
    u = runif(1);
    if(u*c < choose(pois, x)*(1-y)^(pois*x)) {
      return(pois);
    }
  }
}

gibbs_sampling4 = function(n, alpha, beta, lambda, init_param) {
  x = rep(0, n);
  y = rep(0, n);
  n = rep(0, n);
  x[1] = init_param[1];
  y[1] = init_param[2];
  n[1] = init_param[3];

  for(i in 2:n) {
    print(i)
    x[i] = sample4_x(y[i-1], n[i-1]);
    y[i] = sample4_y(x[i], n[i-1], alpha, beta);
    n[i] = sample4_n(x[i], y[i], lambda);
  }
  rlist = list(x, y, n);
  return(rlist);
}

estimate4 = function(n, alpha, beta, lambda) {
  rl = gibbs_sampling4(4*n, alpha, beta, lambda, c(2, 0.5, 3));
  x = rl[[1]];
  y = rl[[2]];
  n = rl[[3]];

  ex = mean(x[(3*n):(4*n)]);
  ey = mean(y[(3*n):(4*n)]);
  en = mean(n[(3*n):(4*n)]);
  return(c(ex, ey, en));
}

n = 10000
alpha = 2
beta = 3
lambda = 4
```

```
estimate4(n, alpha, beta, lambda)
```

Well, there must be some bugs in this program¿..... □

Problem 5.
Generate a mixed normal distribution with the means and covariances given.

Proof.

5 Answer

5.1 the algorithm

1. Randomly select the initial vector X and Y.
2. For each step:
2.1 Generate X_{i+1}, Y_{i+1} with X_i, Y_i using Gibbs Sampling.
2.2 Generate u $U(0,1)$, if $u > 0.5$, set Z = X; else Z = Y.
3. In detail, in order to generate a 2D normal distribution, we can regard the joint distribution as

$$f(x_1, x_2) = \frac{1}{2\pi\sqrt{\det\Sigma}} e^{\frac{1}{2}(a_{11}(x_1-\mu_1)^2+(a_{21}+a_{12})(x_1-\mu_1)(x_2-\mu_2)+a_{22}(x_2-\mu_2)^2)},$$

in which a_{ij} are the elements in Σ, and μ_i are the elements in μ.
3.1 $f(x_2|x_1 = \hat{x_1}) = N(\mu_2 + a_{21}a_{11}^{-1}(\hat{x_1} - \mu_1), \quad a_{22} - a_{21}a_{11}^{-1}a_{12})$,
$f(x_1|x_2 = \hat{x_2}) = N(\mu_1 + a_{12}a_{22}^{-1}(\hat{x_2} - \mu_2), \quad a_{11} - a_{21}a_{22}^{-1}a_{12})$.
3.2 Use Gibbs sampling to generate iteratively X and Y.

5.2 the code is as follows

```
##### Gibbs Sampling for Problem 5 ###
sample5 = function(mu, sigma, x) {
  mean = mu[2]+sigma[2,1]*(1/sigma[1,1])*(x-mu[1]);
  sd = sigma[2,2]-sigma[2,1]*(1/sigma[1,1])*sigma[1,2];
  return(rnorm(1, mean, sd));
}

gibbs_sampling5 = function(n, mu, sigma, init_param) {
  x1 = rep(0, n);
  x2 = rep(0, n);
  x1[1] = init_param[1];
  x2[1] = init_param[2];

  mu2 = c(mu[2], mu[1]);
  sigma2 = matrix(c(sigma[2,2], sigma[2,1], sigma[1,2], sigma[1,1]), ncol = 2);

  for(i in 2:n) {
    x2[i] = sample5(mu, sigma, x1[i-1]);
    x1[i] = sample5(mu2, sigma2, x2[i]);
  }
  rlist = list(x1, x2);
  return(rlist);
}

simulate5 = function(n) {
  mu1 = c(1, 4);
  mu2 = c(-2, -1);
  sigma1 = matrix(c(1, 0.3, 0.3, 2), ncol = 2);
```

```
  sigma2 = matrix(c(3, 0.4, 0.4, 1), ncol = 2);

  Z = matrix(rep(0, 2*n), nrow = 2);
  ##### Each col in Z is a sample point #####

  X = gibbs_sampling5(2*n, mu1, sigma1, c(1, 1));
  Y = gibbs_sampling5(2*n, mu2, sigma2, c(-3, -3));

  x1 = X[[1]];
  x2 = X[[2]];
  y1 = Y[[1]];
  y2 = Y[[2]];

  ##### Take the last half of samples as a simulation ###
  for(i in 1:n) {
    u = runif(1);
    if(u > 0.5) {
      Z[1, i] = x1[i + n];
      Z[2, i] = x2[i + n];
    }
    else {
      Z[1, i] = y1[i + n];
      Z[2, i] = y2[i + n];
    }
  }
  return(Z);
}

n = 10000
z = simulate5(n)
plot(z[1, ], z[2, ])
```

5.2.1 The result is shown as follows

□

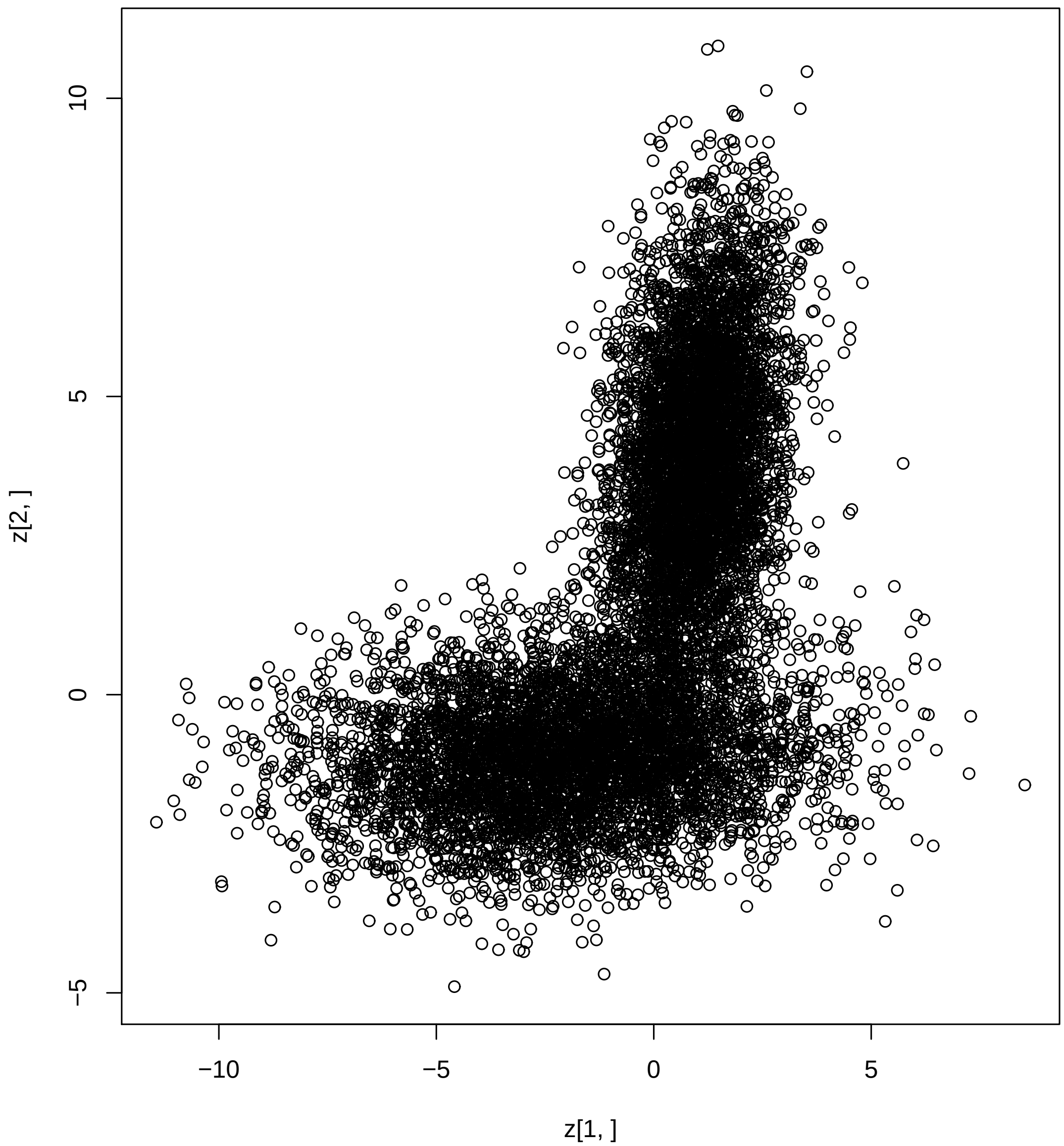

Figure 1: The simulation of the mixed distribution