STAT580 HOMEWORK2

Xingche Guo

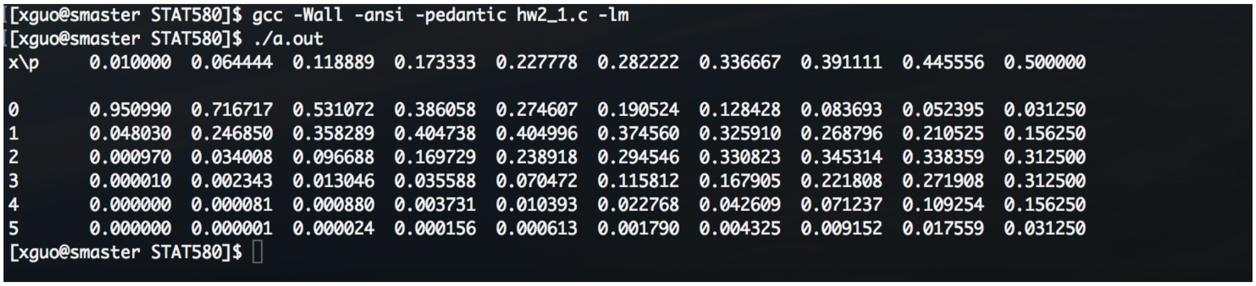
2/9/2017

Problem1

Code:

```
#include<stdio.h>
#include<math.h>
#define P0 0.01
#define P1 0.5
#define PLEN 10
#define N 5

int factorial(int x){
    int i, factorial;
    factorial=1;
    for (i=1; i<=x; i++){
        factorial = factorial * i;
    }
    return (factorial);
}

int choose(int n, int x){
    int c;
    c = factorial(n)/( factorial(x)*factorial(n-x) );
        return (c);
}

double bin_pmf(int n, int x, double p){
    double pmf;
    pmf = choose(n,x) * pow(p,x) * pow(1-p,n-x);
    return (pmf);
}


int main(){
    int x, i, j;
    double p, q, pmf;
    q = (P1 - P0)/(PLEN - 1);
    printf("x\\p\t");
    for (i=0; i<=9; i++){
        printf("%f  ", P0+q*i);
    }
    printf("\n\n");

    for (x=0; x<=N; x++ ){
        printf("%d\t",x);
        for (j=0; j<=9; j++){
            p = P0 + q*j;
            pmf = bin_pmf(N,x,p);
            printf("%f  ",pmf);
        }
        printf("\n");
    }
    return 0;
}
```

Output:



```
[xguo@smaster STAT580]$ gcc -Wall -ansi -pedantic hw2_1.c -lm
[xguo@smaster STAT580]$ ./a.out
x\p     0.010000  0.064444  0.118889  0.173333  0.227778  0.282222  0.336667  0.391111  0.445556  0.500000

0       0.950990  0.716717  0.531072  0.386058  0.274607  0.190524  0.128428  0.083693  0.052395  0.031250
1       0.048030  0.246850  0.358289  0.404738  0.404996  0.374560  0.325910  0.268796  0.210525  0.156250
2       0.000970  0.034008  0.096688  0.169729  0.238918  0.294546  0.330823  0.345314  0.338359  0.312500
3       0.000010  0.002343  0.013046  0.035588  0.070472  0.115812  0.167905  0.221808  0.271908  0.312500
4       0.000000  0.000081  0.000880  0.003731  0.010393  0.022768  0.042609  0.071237  0.109254  0.156250
5       0.000000  0.000001  0.000024  0.000156  0.000613  0.001790  0.004325  0.009152  0.017559  0.031250
[xguo@smaster STAT580]$ 
```

Figure 1: Output of (1)

Problem2

(a)

$$\because \quad f(x) \propto \frac{1}{x},\ 1 < x < 10$$

$$\because \quad \int_1^{10} f(x)dx = 1$$

$$\therefore \quad f(x) = \frac{1}{xlog(10)}$$

$$\therefore \quad F(x) = \int_1^x \frac{1}{xlog(10)}dx = \frac{log(x)}{log(10)}$$

$$\therefore \quad F^{-1}(x) = \exp\{xlog10\}$$

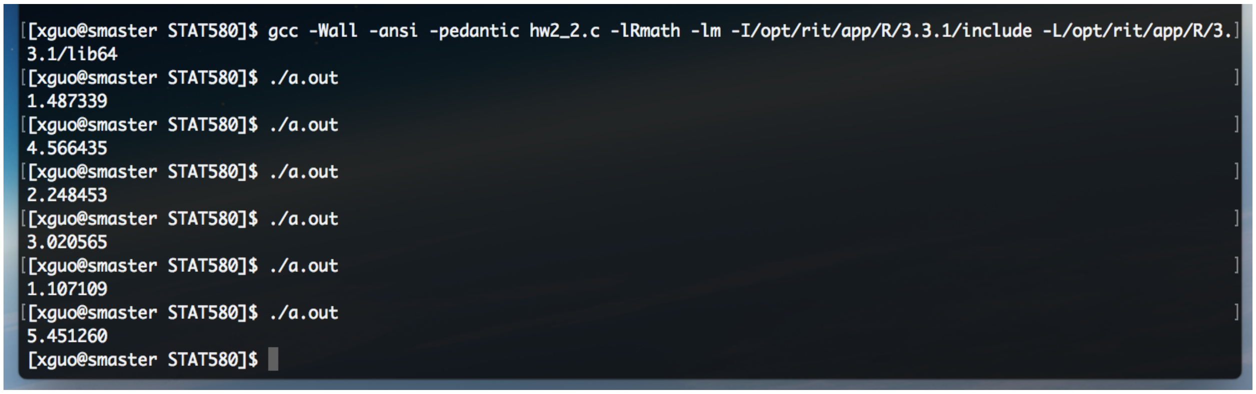
$$\therefore \quad X = F^{-1}(U) = \exp\{Ulog(10)\},\ where\ U \sim Unif(0,1)$$

(b)

Code:

```
#include <stdio.h>
#include <time.h>
#define MATHLIB_STANDALONE
#include <Rmath.h>

int main(){
    double U, X;
    set_seed(time(NULL),580580);
    U = unif_rand();
    X = exp( log(10)*U );
    printf("%f\n", X);
    return (0);
}
```

Output:

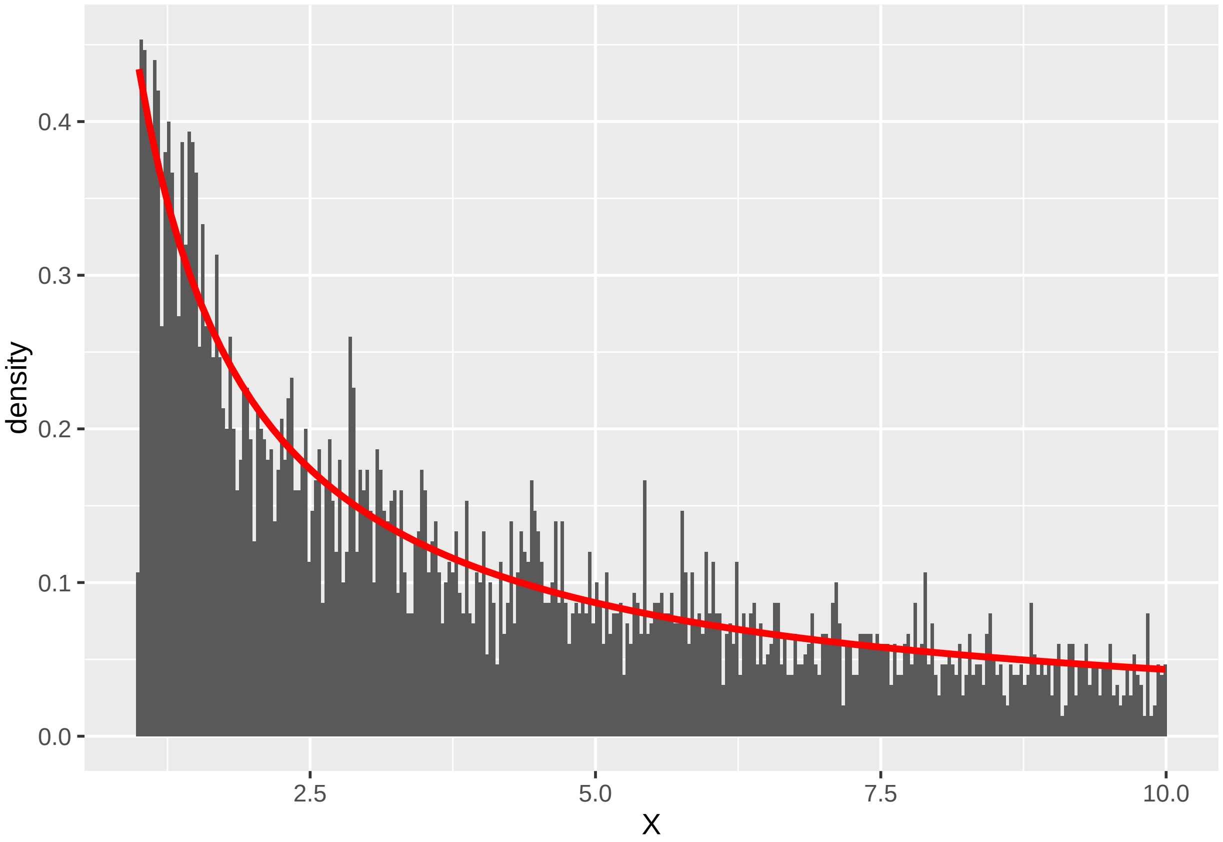


```
[xguo@smaster STAT580]$ gcc -Wall -ansi -pedantic hw2_2.c -lRmath -lm -I/opt/rit/app/R/3.3.1/include -L/opt/rit/app/R/3.
3.1/lib64
[xguo@smaster STAT580]$ ./a.out
1.487339
[xguo@smaster STAT580]$ ./a.out
4.566435
[xguo@smaster STAT580]$ ./a.out
2.248453
[xguo@smaster STAT580]$ ./a.out
3.020565
[xguo@smaster STAT580]$ ./a.out
1.107109
[xguo@smaster STAT580]$ ./a.out
5.451260
[xguo@smaster STAT580]$
```

Figure 2: Output of (1)

(c)

```
#######################
#
# F(x) = log(X)/log(10)
#
# X = exp( log(10) * U )
#
#######################

X <- exp(log(10)*runif(5000))
rX <-as.data.frame(X)

library(ggplot2)

f <- function(x) 1/(x*log(10))

ggplot(data=rX) +
  geom_histogram(aes(x = X, ..density..),binwidth = 0.03)+
  stat_function(fun = f, colour = "red", size=1.2)
```



Problem3

$$q(x) = \frac{e^{-1}}{1+x^2}; \quad g_1(x) = e^{-x}; \quad g_2(x) = \frac{2}{\pi(1+x^2)}$$

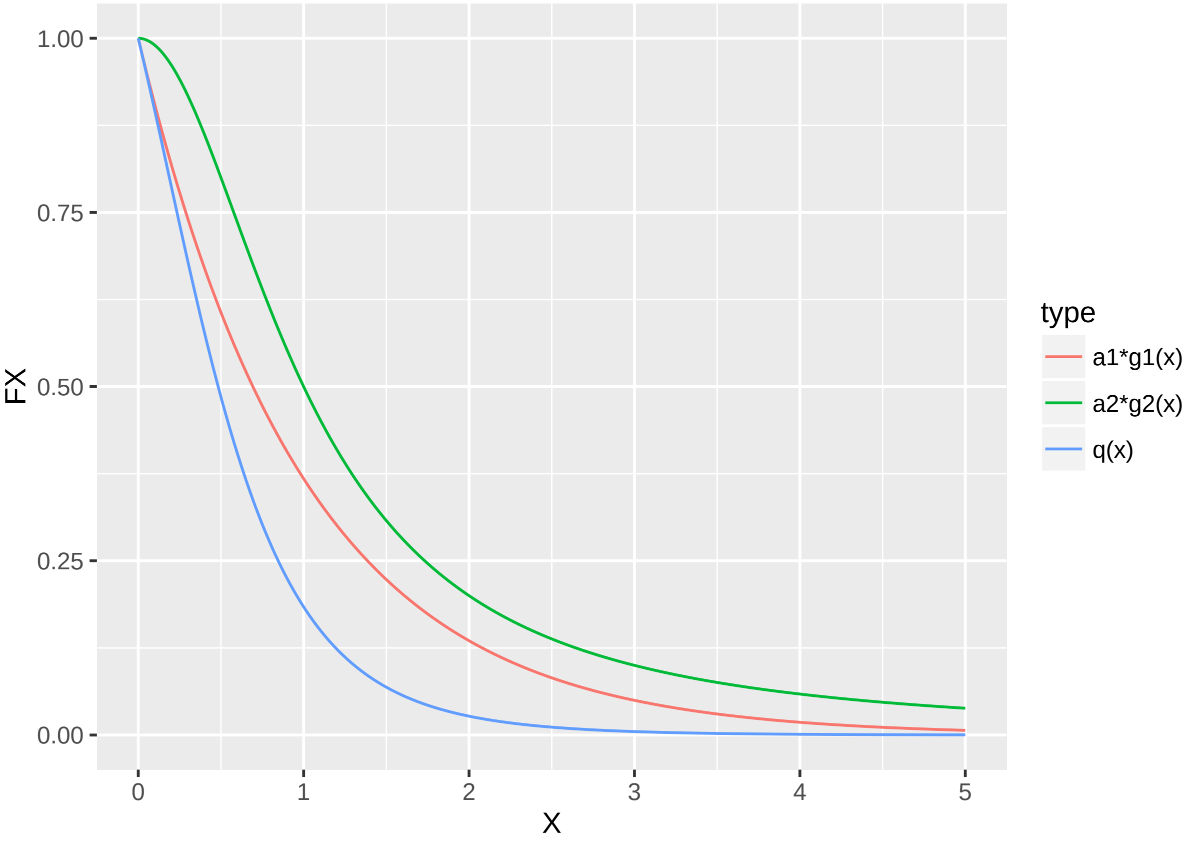
It's easy to find that:

$$X_1 \sim \exp(1), \quad X_2 = |Y|, \; where \;\; Y \sim Cauchy(0,1)$$

(a)

```
Q <- function(x){
  out <- exp(-x)/(1 + x^2)
  return(out)
}

### g1~exp(1); g2~abs(cauchy(0,1))
a1 <- 1
a2 <- pi/2
x <- seq(0, 5, length.out=500)
X <- rep(x,times = 3)
q <- Q(x)
g1 <- a1*dexp(x, 1)
g2 <- a2*2*dcauchy(x,0,1)
FX <- c(q,g1,g2)
type <- factor(rep(c("q(x)","a1*g1(x)","a2*g2(x)"), each=500))
```

```
df <- data.frame(X,FX,type)

ggplot(data = df,aes(x = X, y = FX)) +
  geom_line(aes(group = type, colour = type))
```



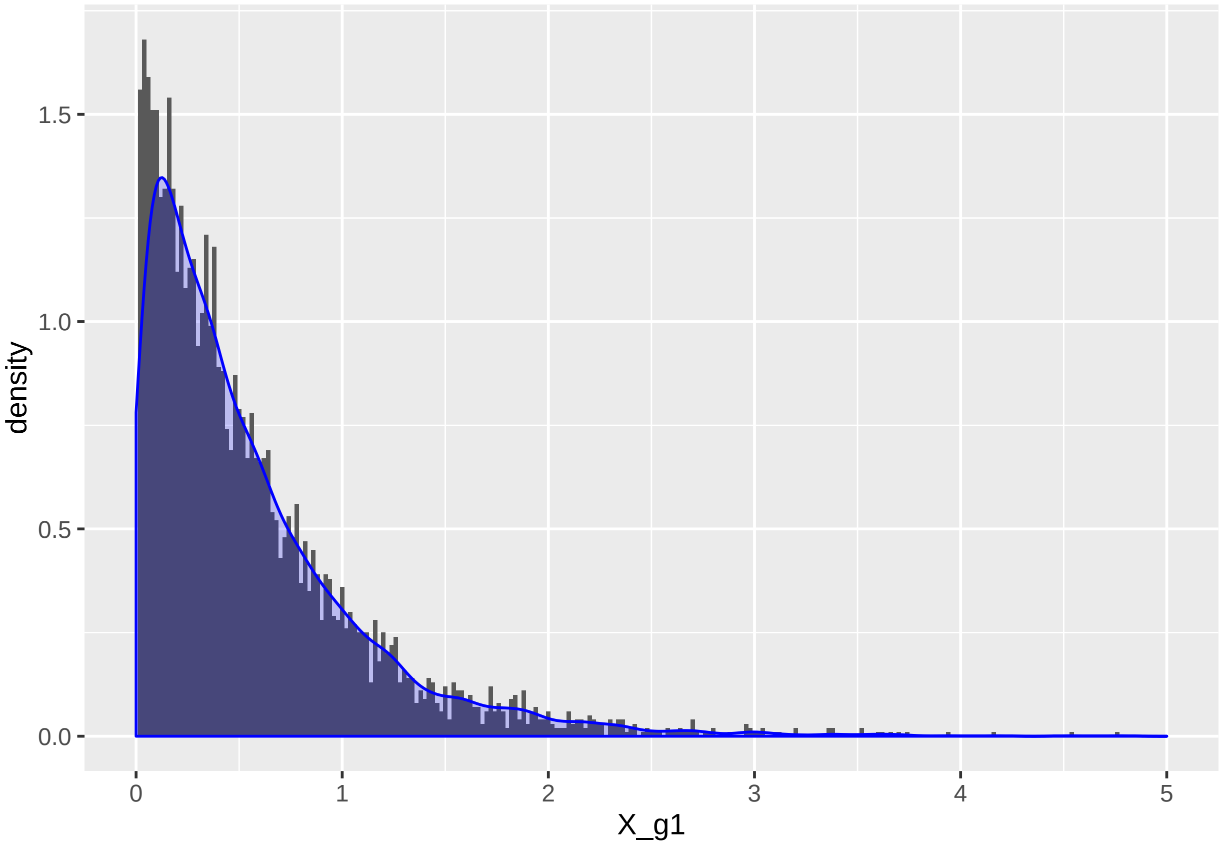
```
### for g1
U <- runif(30000)
X1 <- rexp(30000,1)
not_rej <- (    U <= Q(X1)/(a1*dexp(X1,1))    )
g1_accept_ratio <- sum(not_rej)/length(X1)
g1_accept_ratio
```

```
## [1] 0.6182
```

```
X_g1<-X1[not_rej][1:5000]
ggplot(data = as.data.frame(X_g1), aes(x = X_g1, ..density..)) +
  geom_histogram(binwidth = 0.02) +
  geom_density(colour = "blue", fill="blue", alpha=0.2)+
  xlim(0,5)
```

```
## Warning: Removed 1 rows containing non-finite values (stat_bin).
```

```
## Warning: Removed 1 rows containing non-finite values (stat_density).
```

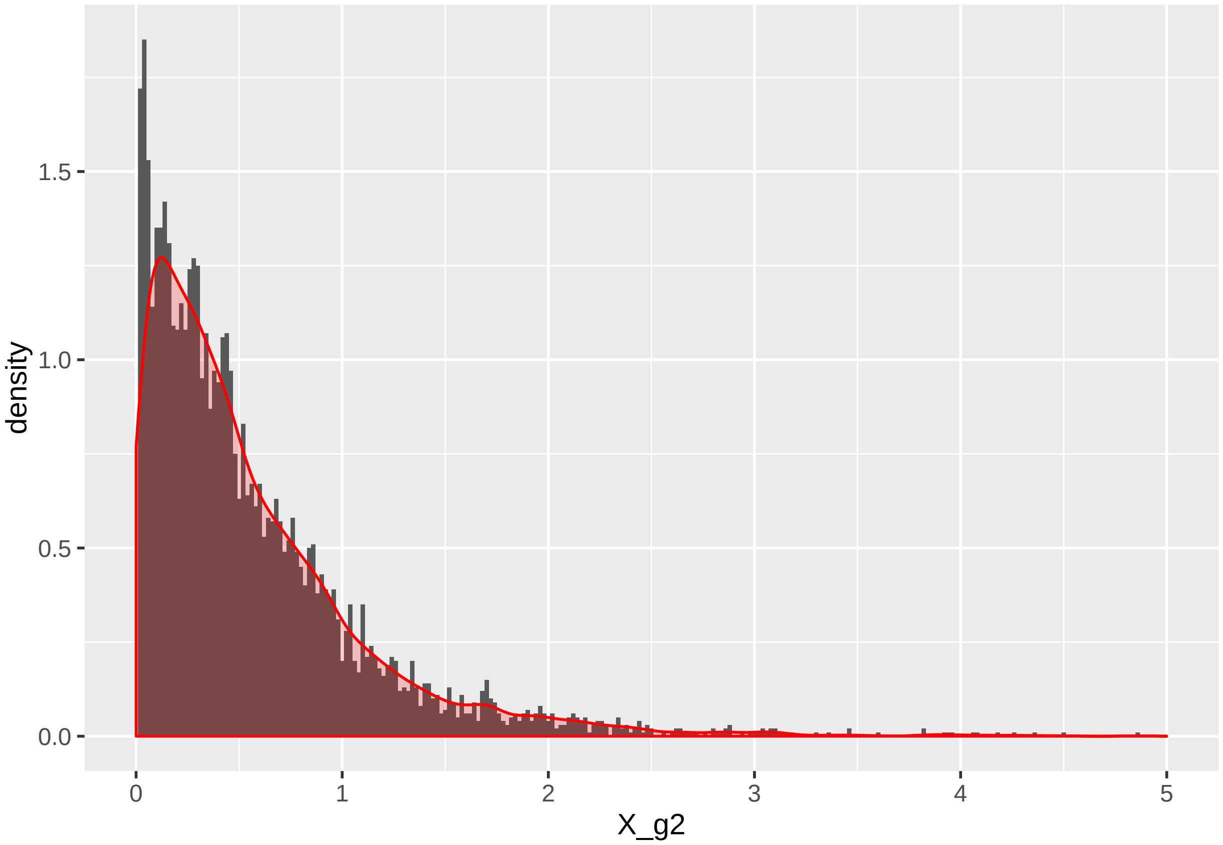


```
### for g2
U <- runif(30000)
X2 <- abs(rcauchy(30000,0,1))
not_rej <- (    U <= Q(X2)/(a2*2*dcauchy(X2,0,1))    )
g2_accept_ratio <- sum(not_rej)/length(X2)
g2_accept_ratio
```

```
## [1] 0.3981667
```

```
X_g2<-X2[not_rej][1:5000]
ggplot(data = as.data.frame(X_g2), aes(x = X_g2, ..density..)) +
  geom_histogram(binwidth = 0.02) +
  geom_density(colour = "red", fill="red", alpha=0.2)+
  xlim(0,5)
```

```
## Warning: Removed 1 rows containing non-finite values (stat_bin).

## Warning: Removed 1 rows containing non-finite values (stat_density).
```



(b)

```
g1_accept_ratio > g2_accept_ratio
```

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

We can compare the speeds of the two methods through the accept ratio. The result shows that using g1 is faster than using g2. We can also find this result by checking the plot of the 3 functions. g1 is closer to q than g2.

Problem4

Algorithm:

In this problem, I will use the rejection sampling to build the model. We know that:

$$f(x,y) \propto x^\alpha y, \quad x>0, \quad y>0, \quad x^2+y^2 \le 1$$

Therefore, we can find a $g(x,y)$, such that:

$$g(x,y) \propto x^\alpha y, \quad 0<x \le 1, \quad 0<y \le 1$$

It's easy to see that X and Y are independent. So:

$$g(x,y) = u(x)v(y), \quad where: \ u(x) \propto x^\alpha, \quad v(y) \propto y$$

Because of the property of pdf, it's not hard to find that:

$$u(x) = (\alpha + 1)x^a, \quad v(y) = 2y^2$$

$$\therefore \quad U(x) = x^{a+1}, \quad V(x) = y^2$$

Therefore, we can generate random varibles X and Y by:

$$X = U_1^{\frac{1}{\alpha+1}}, \quad Y = U_2^{\frac{1}{2}}$$

where:

$$U_1, U_2 \sim Unif(0,1), \quad U_1, U_2 \ are \ independent.$$

So, the algorithm is:

Step1:

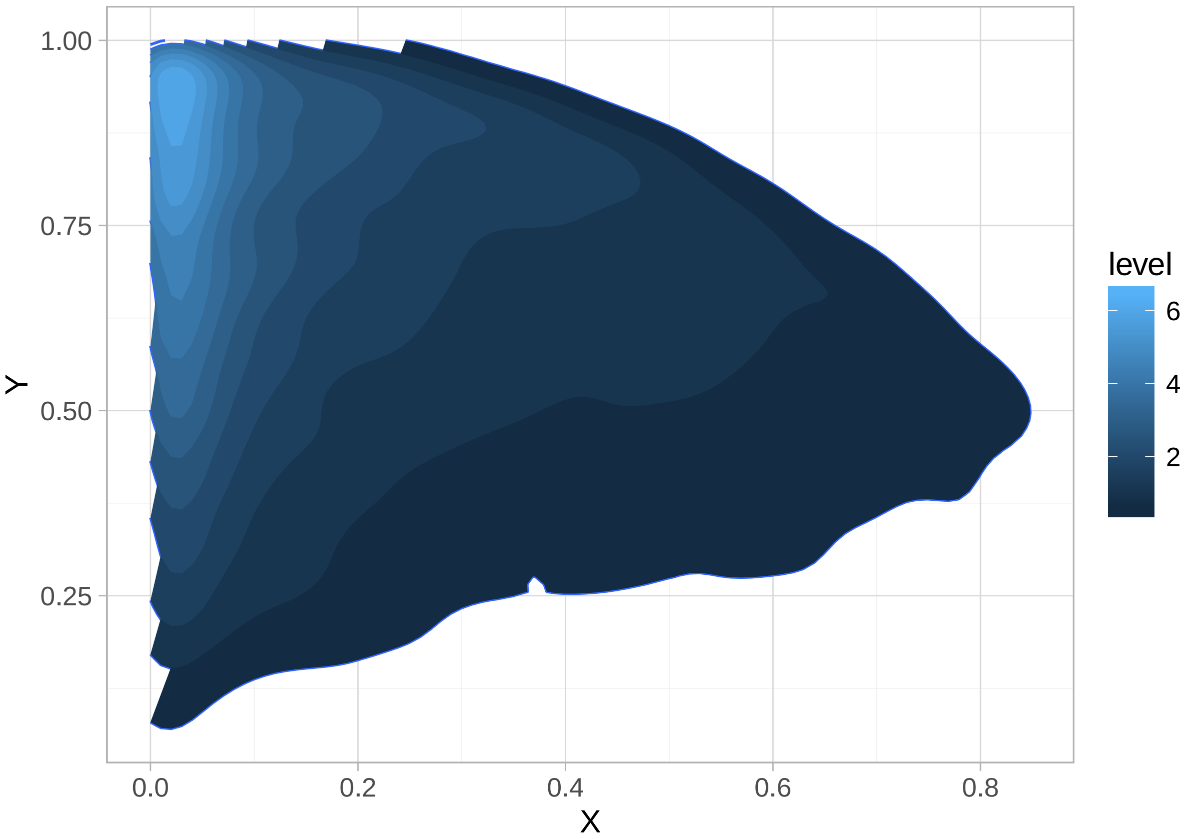
Generate X,Y.

Step2:

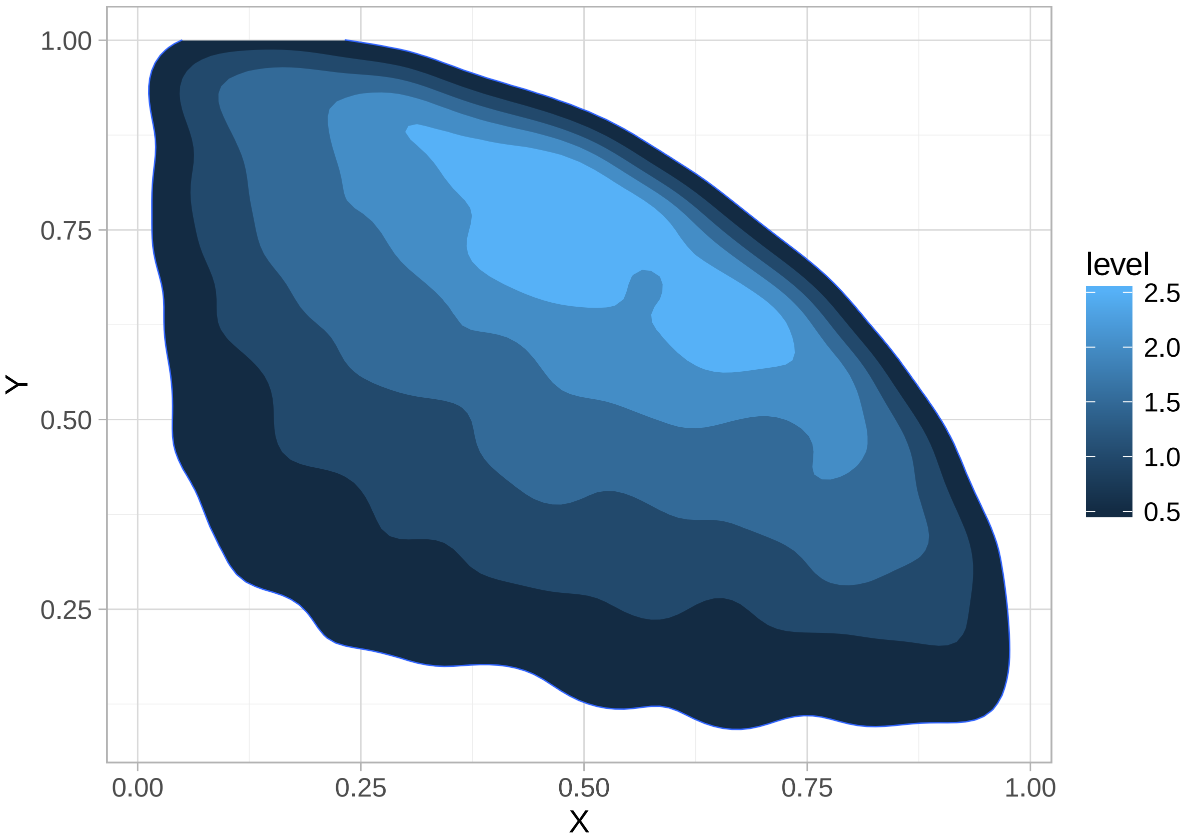
If $X^2 + Y^2 > 1$, then go to step 1, otherwise return (X,Y). The returned value is a random variable from $f(x,y)$.

Code:

```
## Set f(y) = 2*y  (0 <= y <= 1)
## Set f(x) = (a+1)*x^a (0 <= x <= 1)
## Set x,y independent.
## Thus, f(x,y)=2*(a+1)*y*x^a (0 <= x,y <= 1)

## Hence F(y) = y^2   ---->  Fy^(-1) = sqrt(y)
## Hence F(x) = x^(a+1) ----> Fx^(-1) = x^( 1/(a+1)  )

a = -1/2
U1 <- runif(30000)
U2 <- runif(30000)
Y0 <- sqrt(U1)
X0 <- U2^( 1/(a+1) )
not_rej <- (X0^2+Y0^2<=1)
X <- X0[not_rej]
Y <- Y0[not_rej]
d <- data.frame(X,Y)

library(ggplot2)
ggplot(data = d, aes(x = X, y = Y))+
  geom_density_2d()+
  stat_density_2d(aes(fill=..level..),geom="polygon")+
  theme_light()
```



```
a=1/2
U1 <- runif(30000)
U2 <- runif(30000)
Y0 <- sqrt(U1)
X0 <- U2^( 1/(a+1) )
not_rej <- (X0^2+Y0^2<=1)
X <- X0[not_rej]
Y <- Y0[not_rej]
d <- data.frame(X,Y)

library(ggplot2)
ggplot(data = d, aes(x = X, y = Y))+
  geom_density_2d()+
  stat_density_2d(aes(fill=..level..),geom="polygon")+
  theme_light()
```


1.00
0.75
0.50
0.25
Y
0.00
0.25
0.50
0.75
1.00
X
level
2.5
2.0
1.5
1.0
0.5