HW1 - STAT 580 - Sp 2015

Yet T Nguyen

01/29/2015

1.

```
#include <stdio.h>
#include <math.h>
#define P0 0.01 /* lower limit of the probability (p)*/
#define P1 0.5 /* upper limit of the probability (p)*/
#define PLEN 10 /* number of columns*/
#define N 5 /* number of experiments (n)*/


int fact(int n); /*funtion to calculate factorial of n*/

int main(){
  int x = 0;
  float dist, p;
  p = P0;
  dist = (P1-P0)/(PLEN-1);
  printf("x\\p\t");
  for (p = P0; p <= P1; p+= dist){
    printf("%.4f ", p); /* print p values*/
  }
  printf("\n\v"); /* print vertical tab */
  for (x = 0; x <= N; x++){
    printf("%d  \t", x);
    for (p = P0; p <= P1; p+= dist){
      printf("%.4f ", fact(N)/(fact(x)*fact(N-x))*pow(p,x)*pow(1-p, N-x)); /*print probabilites*/
    }
    printf("\n");
  }
  return 0;
}

int fact(int n){
  int out = 1;
  if (n ==0) out = 1;
  while (n >= 1){
    out = out*n;
    n--;
  }
  return out;
}
```

2.

(a) We have

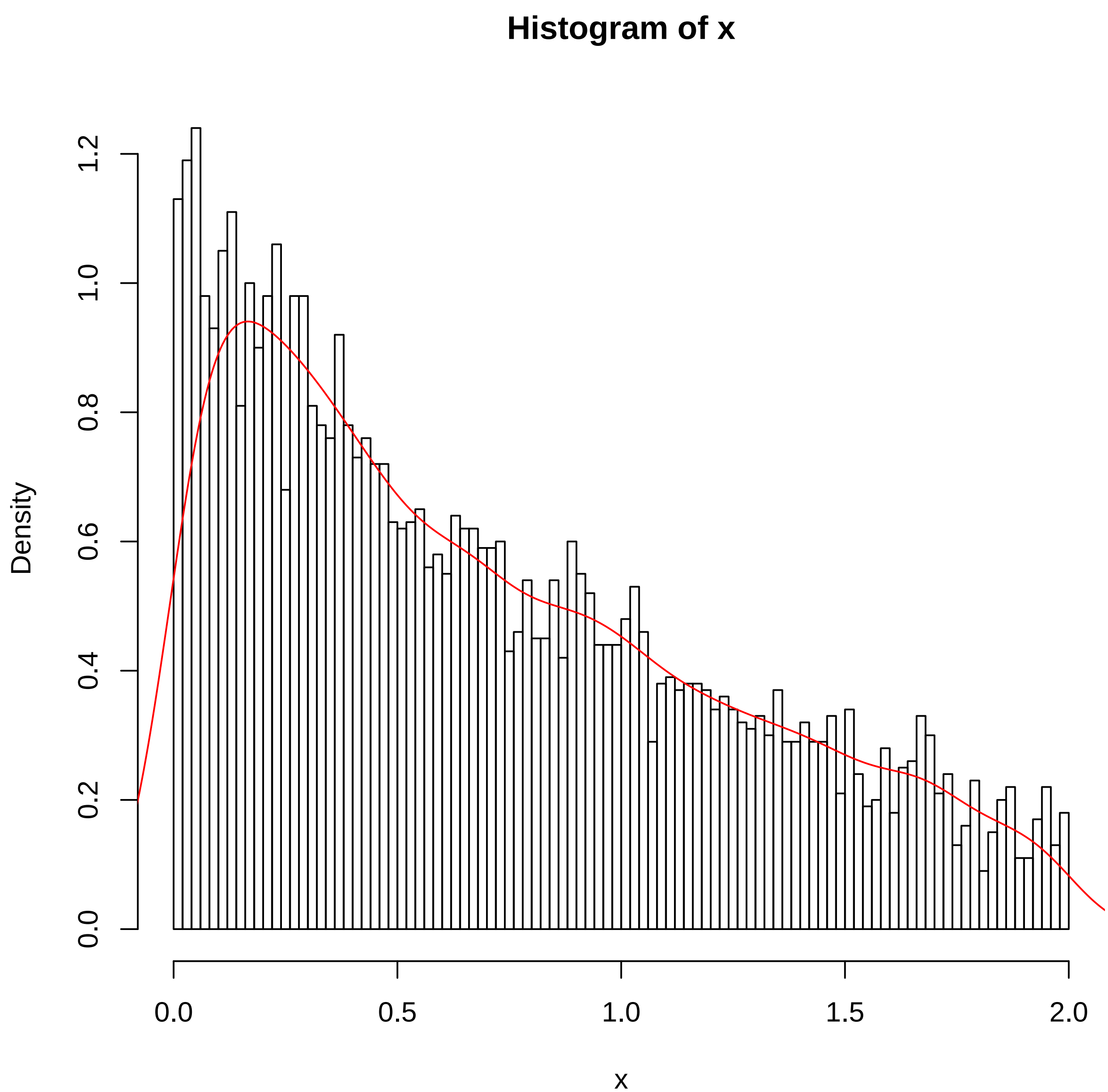
$$f(x) \propto \exp(-x) \quad 0 < x < 2$$
$$\Rightarrow F(x) = (1 - \exp(-x))/(1 - exp(-2)) \quad 0 < x < 2$$
$$\Rightarrow F^{-1}(u) = -\log(1 - (1 - \exp(-2))u) \quad 0 < u < 1.$$

The simulation algorithm is as below

- Generate a uniform random variable u.
- The variable $X \equiv F^{-1}(u)$ will has the requried distribution.

(b)
```
#include <stdio.h>
#include <time.h>
#define MATHLIB_STANDALONE
#include <Rmath.h>
int main() {
  double u, x;
  set_seed(time(NULL), 580580); /* set seed */
    u = unif_rand(); /* uniform random variable */
    x = -log(1- (1-exp(-2))*u);
    printf("%f\n ", x);
    return 0;
    }
```

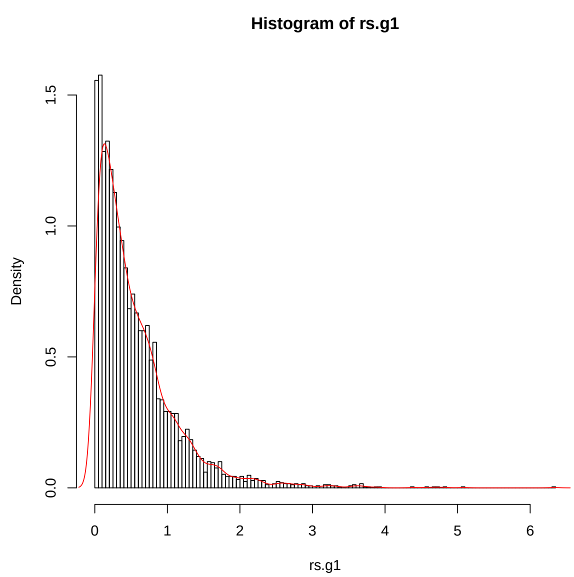
(c)
```r
u <- runif(5000, 0, 1)

x <- -log(1-(1-exp(-2))*u)
hist(x, prob = T, nclass = 100)
lines(density(x), col = "red")
```

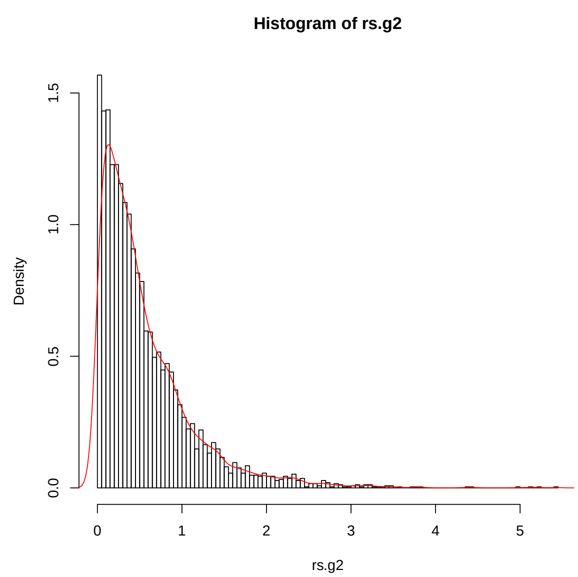


3.

(a)
```
set.seed(1)
rs <- function(g){
  if (g=="g1"){
    repeat{
      x <- rexp(1,1)
      u <- runif(1,0,1)
      r <- 1/(1+x^2)
      if (u <= r){
        res <- x
        break
      }
  }
  }
```

```r
  if (g=="g2"){
    repeat{
      x <- abs(rcauchy(1,0,1))
      u <- runif(1,0,1)
      r <- exp(-x)
      if (u <= r){
        res <- x
        break
      }
    }
  }
  res
}

library(plyr)
n <- 5000
pm1 <- proc.time()
rs.g1 <- laply(1:n, function(i)rs("g1"))
proc.time()-pm1

##    user  system elapsed
##   0.119   0.000   0.124

hist(rs.g1, nclass = 100, prob = T)
lines(density(rs.g1), col = "red")
pm2 <- proc.time()
rs.g2 <- laply(1:n, function(i)rs("g2"))
proc.time()-pm2

##    user  system elapsed
##   0.204   0.004   0.209

hist(rs.g2, nclass = 100, prob = T)
lines(density(rs.g2), col = "red")
```





(b) From the output, the algorithm using envelop density g_1 is faster than the one using envelop density g_2. The sampling results of those densities are similar.

4.

We have

$$\begin{aligned} & f(x,y) \propto x^\alpha y \\ & \Rightarrow f(x) \propto \int_{0<y<\sqrt{1-x^2}} x^\alpha y dy \propto x^\alpha (1-x^2) \quad \text{for } 0<x<1 & (1) \\ & \text{and } f(y|x) \propto \frac{x^\alpha y}{x^\alpha (1-x^2)} \propto y \quad \text{for } y \le \sqrt{1-x^2}. & (2) \end{aligned}$$

(1) implies that

$$x^\alpha (1-x^2) \le x^\alpha \tag{3}$$

and (2) implies that

$$F(y|x) = \frac{y^2}{1-x^2} \quad 0 < y \le 1-x^2, x>0 \tag{4}$$

From (3) and (4), a rejection sampling algorithm to sample (x, y) has distribution f is as below

- First simulate x with density $f_x \propto x^\alpha (1-x^2)$ by
  - Simulate z having the density $\propto z^\alpha$ $(0 < z < 1)$ by using the inverse transform method.
  - Using z above as a proposal to simulate x according to rejection sampling method.
- Next, simulate $y|x$ having CDF as in (4) by using inverse transform method.