Homework 5

Jing Leng

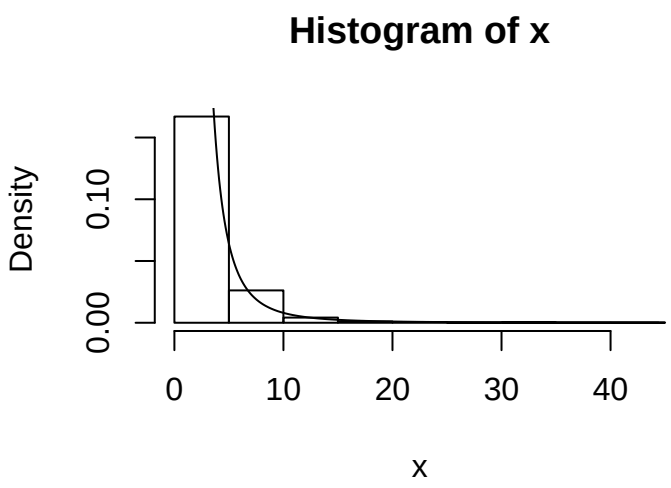
October 22, 2014

1

a

$F^{-1}(u) = \frac{b}{\sqrt[a]{1-u}}$

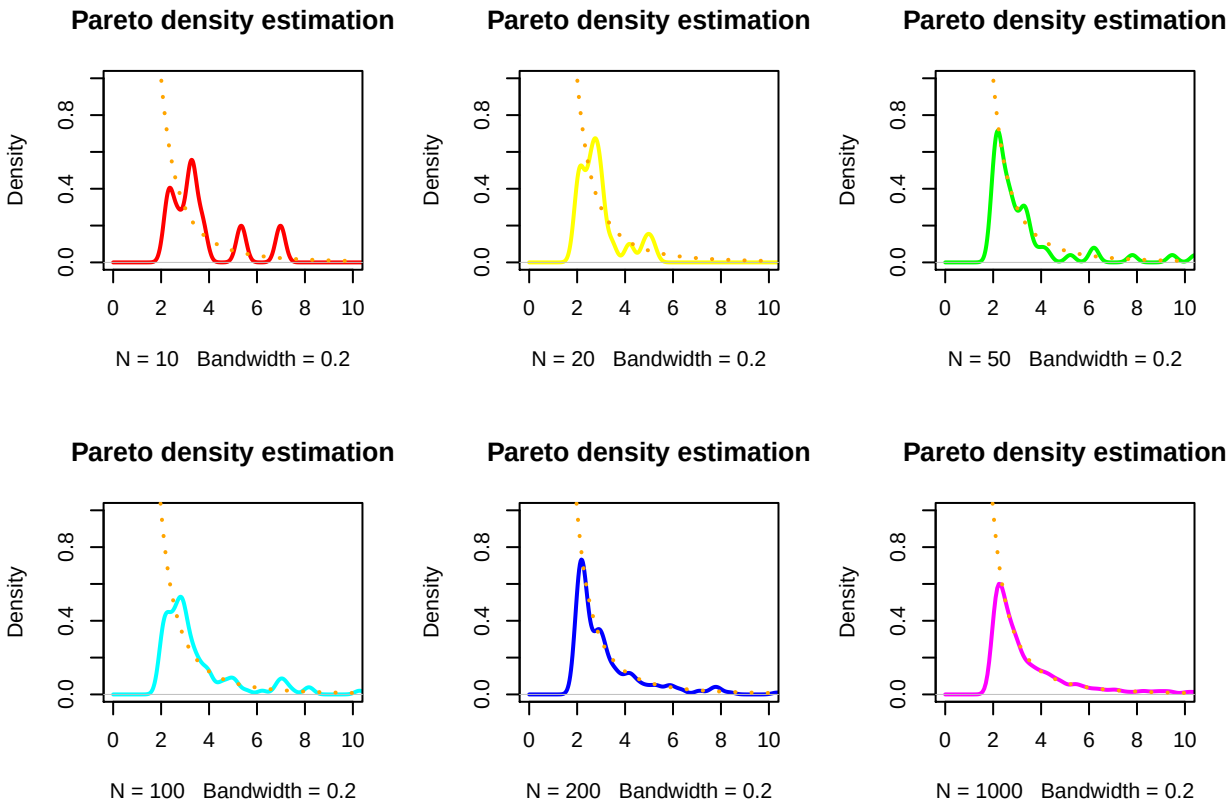
```
u = runif(1000)
x <- 2/sqrt(1-u)
hist(x, prob = T)
y <- seq(2, max(x), 0.1)
lines(y, 8*y^-3)
```



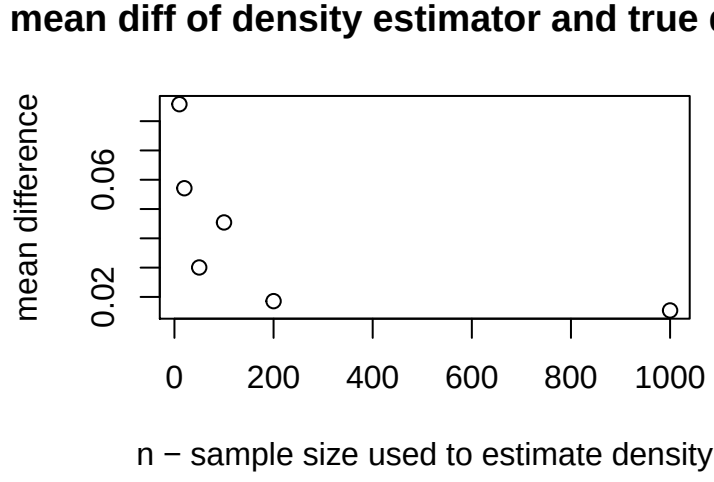
b

```
n = c(10, 20, 50, 100, 200, 1000)
pareto = lapply(n, function(i) 2/sqrt(1-runif(i)) )

grid = 1000
y = seq(0,10,length = grid)
par(mfrow=c(2,3))
for(i in 1:6){
  di = density(pareto[[i]],from=0,to=max(pareto[[i]]), n = grid, bw=0.2)
  plot(di, xlim=c(0,10), ylim=c(0,1), col=rainbow(6)[i], lwd = 2, main="Pareto density estimation")
  lines(y, 8*y^-3, col = 'orange', lty = 3, lwd = 2)
}
```




```
MeanDensDiff <- function(pareto_i){
  grid = 1000
  z = seq(0,10,length = grid)
  d <- density(pareto_i,from=2 ,to=10, n=grid, bw=.2)
  return(mean(abs( d$y - 8*d$x^-3 )))
}
meandiff = lapply(pareto, MeanDensDiff)
par(mfrow = c(1,1))
plot(n, meandiff, xlab = 'n - sample size used to estimate density',
     ylab = 'mean difference',
     main = 'mean diff of density estimator and true density')
```



2

```
k = 10
n <- c(10, 50, 100, 500, 1000)
p <- c(rep(1/k^2, k), 1-1/k)

gen <- function(m) {
  random <- integer(m)
  for (j in 1:m) {
  u <- runif(1)

  i = 1
  s = p[i]
  while (s < u) {
    i = i+1
    s = s + p[i]
  }
  random[j] = i
  }
  return (random)
}
sapply(n, function(i) system.time(gen(i)))
```

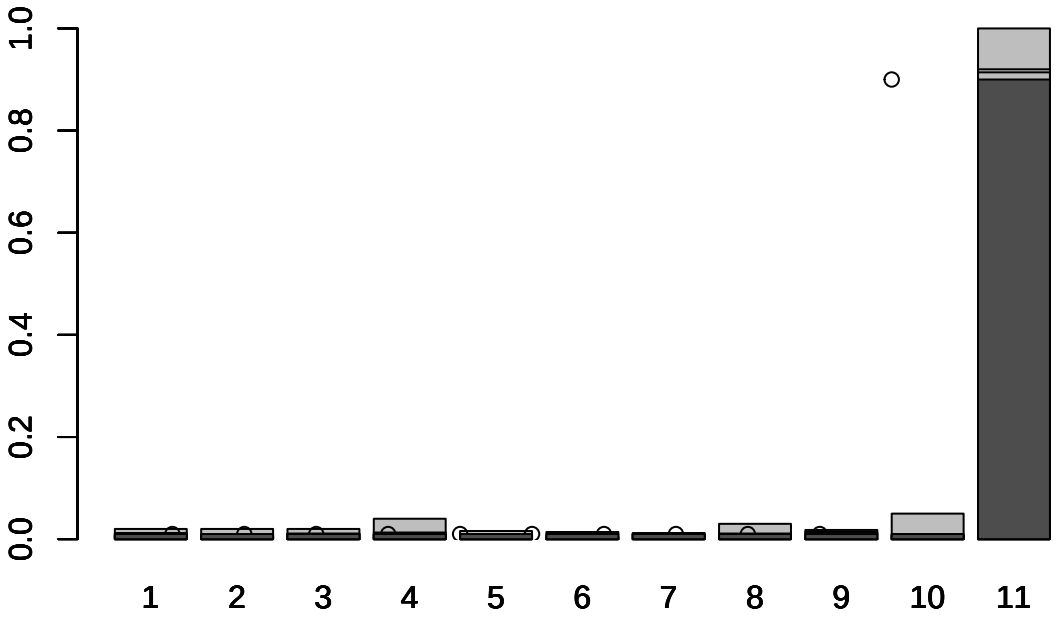
```
##             [,1]  [,2]  [,3]  [,4]  [,5]
## user.self  0.001 0.003 0.009 0.039 0.081
## sys.self   0.000 0.000 0.000 0.000 0.004
## elapsed    0.001 0.003 0.012 0.040 0.090
## user.child 0.000 0.000 0.000 0.000 0.000
## sys.child  0.000 0.000 0.000 0.000 0.000
```

To compare the distribution, plot the frequency.

```
sp <- lapply(n, gen)
sp <- lapply(sp, function(i) i = factor(i, levels(i) <- c(1:(k+1))))

frame()
a <- lapply(sp, function(i) {tb <- prop.table(table(i)); par(new = T); barplot(tb, ylim = c(0,1)) })
points( p)
par(new = T)

true <- matrix(p, nrow = 1)
colnames(true) <- 1:(k+1)
barplot(true, ylim = c(0,1))
```



b

It is slower because it take a lot of time looping and adding up the P. We could precalculate it so that we don't add it from start (and to end in most cases) every time.

```
k <- 1000
p <- c(rep(1/k^2, k), 1-1/k)
n <- c(100, 1000, 10000)
sapply(n, function(i) system.time(gen(i)))
```

```
##             [,1]  [,2]   [,3]
## user.self  0.425 4.122 31.936
## sys.self   0.003 0.029  0.194
## elapsed    0.434 4.422 33.358
## user.child 0.000 0.000  0.000
## sys.child  0.000 0.000  0.000
```

```
newgen <- function(m) {
  random <- integer(m)
  cp <- sapply(1:(k+1), function(a) sum(p[1:a]))


  for (j in 1:m) {
    u <- runif(1)
    ind = 0
    for (i in 1:(k+1)) {
      if (u >= cp[i]) {
        ind = i
        break
      }
    }
    random[j] = ind
```

```
  }
  return (random)
}

sapply(n, function(i) system.time(newgen(i)))
```

```
##             [,1]  [,2]  [,3]
## user.self  0.047 0.093 0.388
## sys.self   0.005 0.004 0.007
## elapsed    0.052 0.123 0.429
## user.child 0.000 0.000 0.000
## sys.child  0.000 0.000 0.000
```

It reduced time by more than 90%.