Assignment3.revision.R

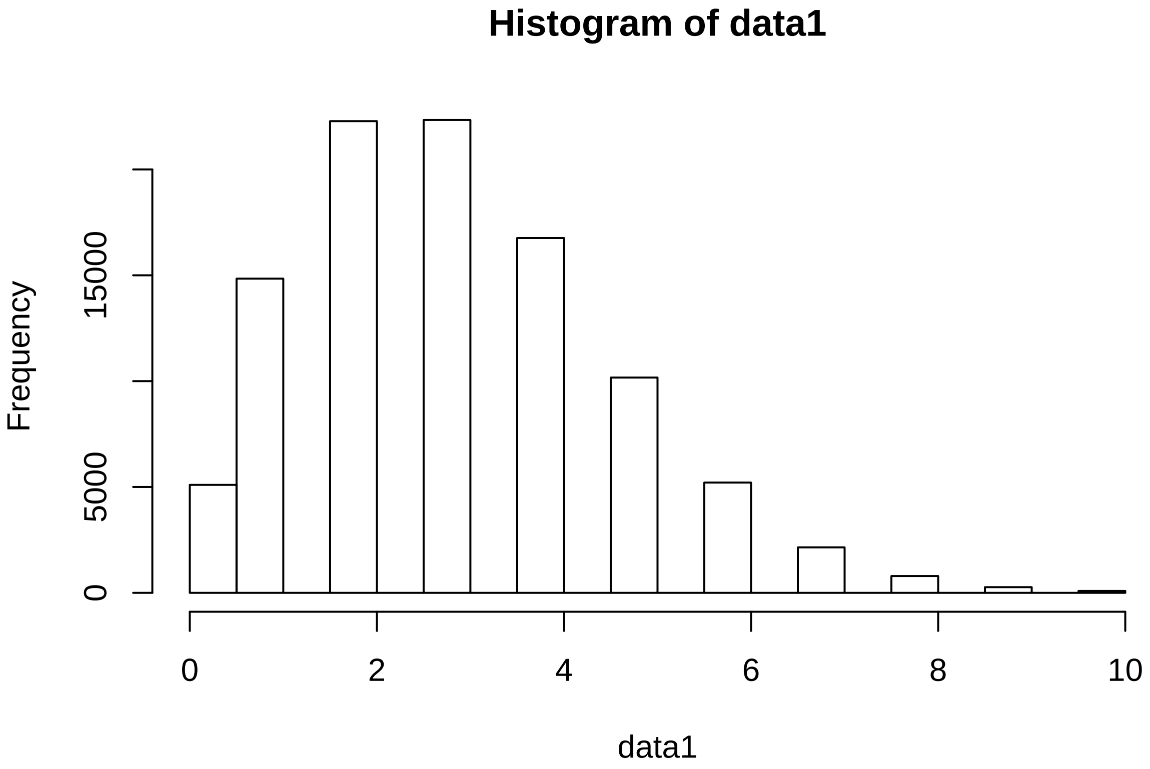
EvergreenFu

Wed Jun 22 00:21:43 2016

```r
###################################################
## dependencies
require(stats)
require("matlab")
```

```
## Loading required package: matlab

##
## Attaching package: 'matlab'

## The following object is masked from 'package:stats':
##
##     reshape

## The following objects are masked from 'package:utils':
##
##     find, fix

## The following object is masked from 'package:base':
##
##     sum
```

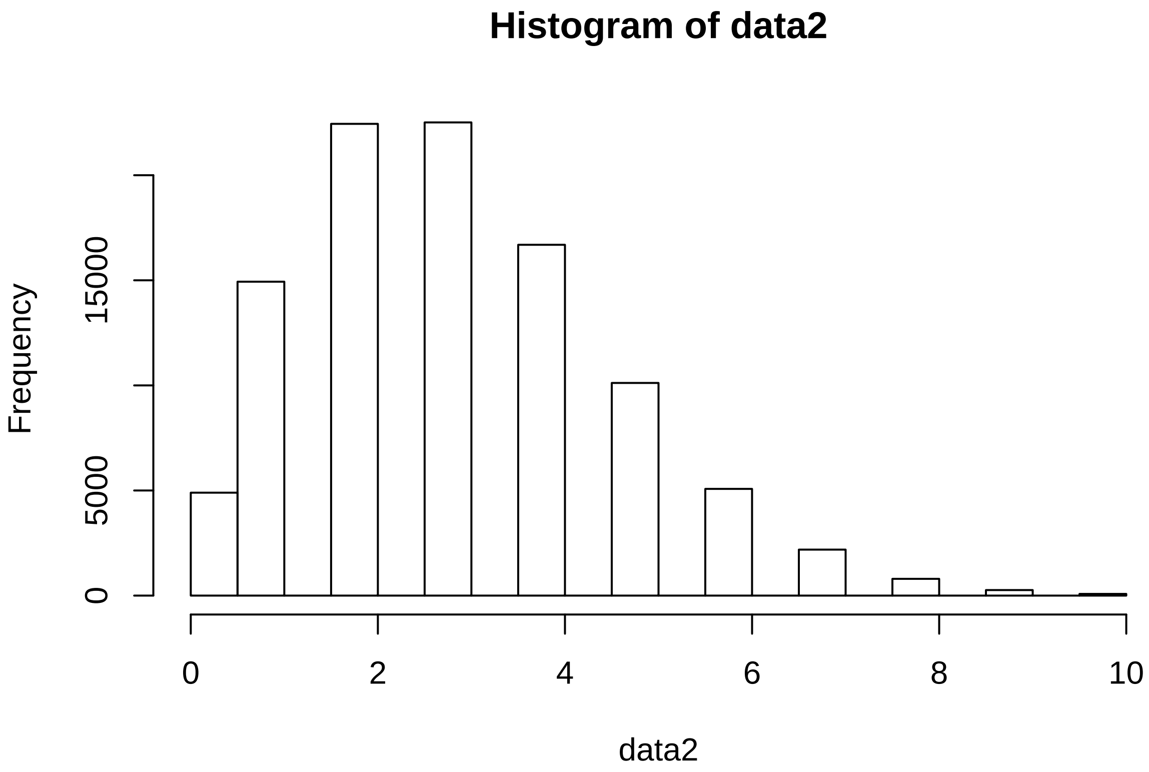
```r
###################################################
## 0.some useful functions
#  discrete uniform distribution of 0~k
rfunif <- function(n,k) floor(runif(n)*(k+1))
#  poisson density function
dfpois = function(i, k = 10, lambda = 1){
  exp(-lambda)*lambda^i/factorial(i)/sum(exp(-lambda)*lambda^(0:k)/factorial(0:k))
}

###################################################
## 1.rfpoi(n, k, lambda, ...): Finite Poisson distribution
#  Input: n         niter
#         k         finite element number
#         lambda    poisson distribution parameter
#  Output: Generate n random numbers x
#  methods = inverse transform; acceptance/rejection
rfpois = function(n, k = 10, lambda = 1, method = c("inverse","acceptance"))
{
  if(n < 0 | k < 0){
    stop("invalid input argument")
  }
  method = match.arg(method)
  result <- if (method == "inverse") {
    u = runif(n)*sum(exp(-lambda)*lambda^(0:k)/factorial(0:k))
    sapply(u, function(u) {
      p = exp(-lambda);f = p
      for(i in 0:k){
        if(u<f){return(i)}
        p = lambda*p/(i+1);f = f + p
      }
    })
  }
  else if (method == "acceptance"){
    c = (k+1)*max(sapply(0:k,dfpois, k = k, lambda = lambda))
    x = rep(0,n)
    for(i in 1:n){
      repeat{
        x[i] = rfunif(1,k)
        U = runif(1)
        if(U < (k+1)*dfpois(x[i],k,lambda)/c)
          break
      }
    }
    x
  }
}
#  inferring parameters
data1 = rfpois(100000, 10, 3, method = "inverse")
hist(data1)
```



```
mean(data1)
```

```
## [1] 3.00255
```

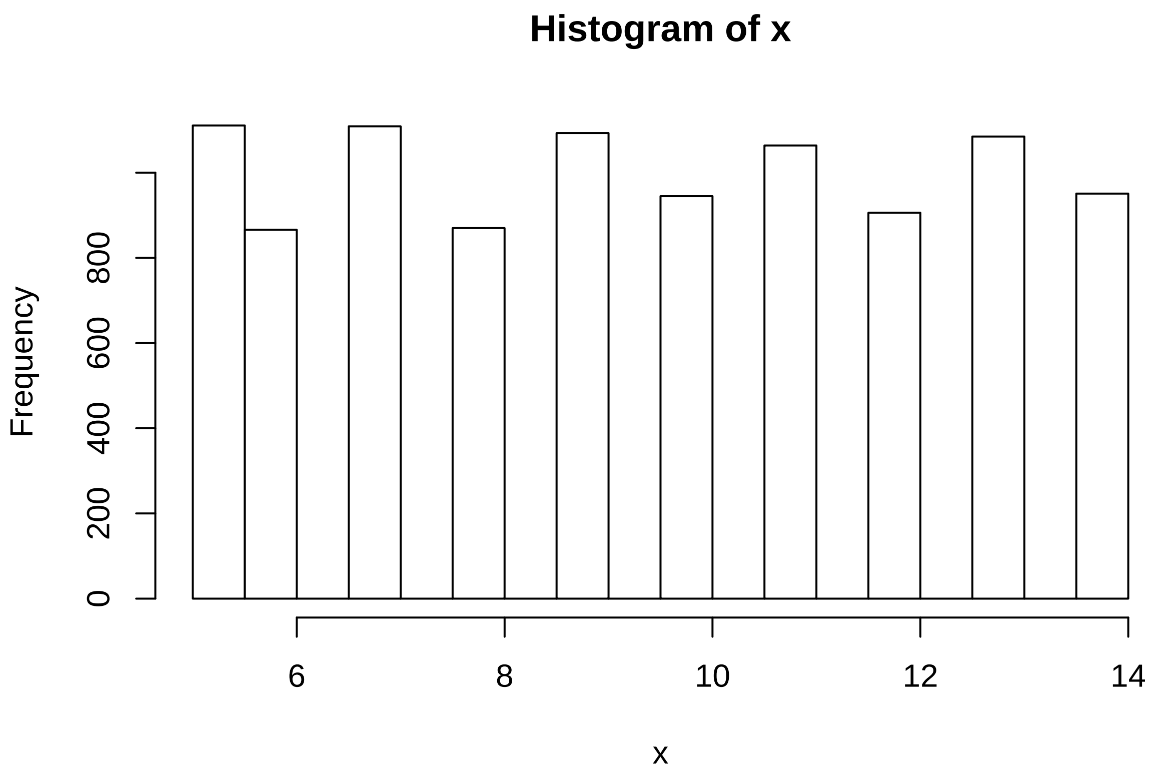
```
data2 = rfpois(100000, 10, 3, method = "inverse")
hist(data2)
```



```
mean(rfpois(1000, 10, 3, method = "acceptance"))
```

```
## [1] 3.033
```

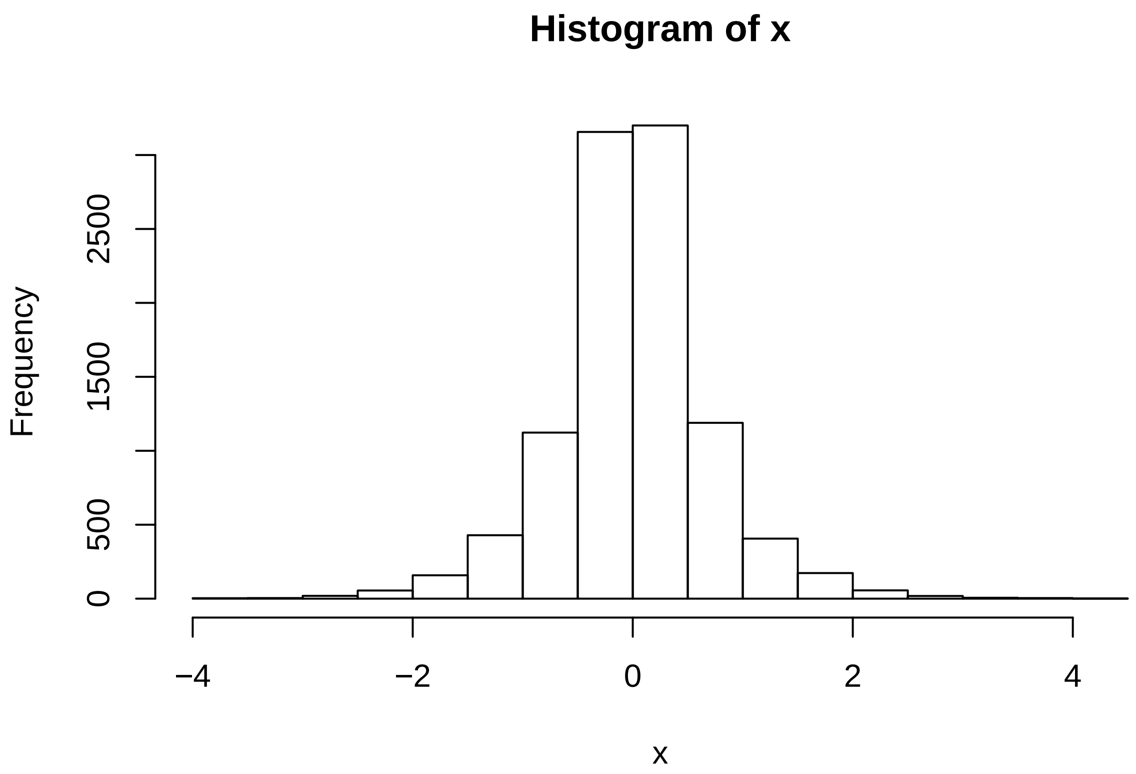
```
###################################################
## 2. simulate from a certain discrete distribution
#  output: x ~ p_x=0.11(x = 5,7,...,13); p_x = 0.09(x = 6,8,...,14)
foo <- function(n){
  if(n < 0){stop("invalid input")}
  sapply(runif(n),function(u){
    for(i in 0:4){
      if(u <= 0.2*i+0.11){return(2*i+5)}
      else if(u<=0.2*(i+1)){return(2*i+6)}
    }
  })
}
x = foo(10000)
hist(x)
```



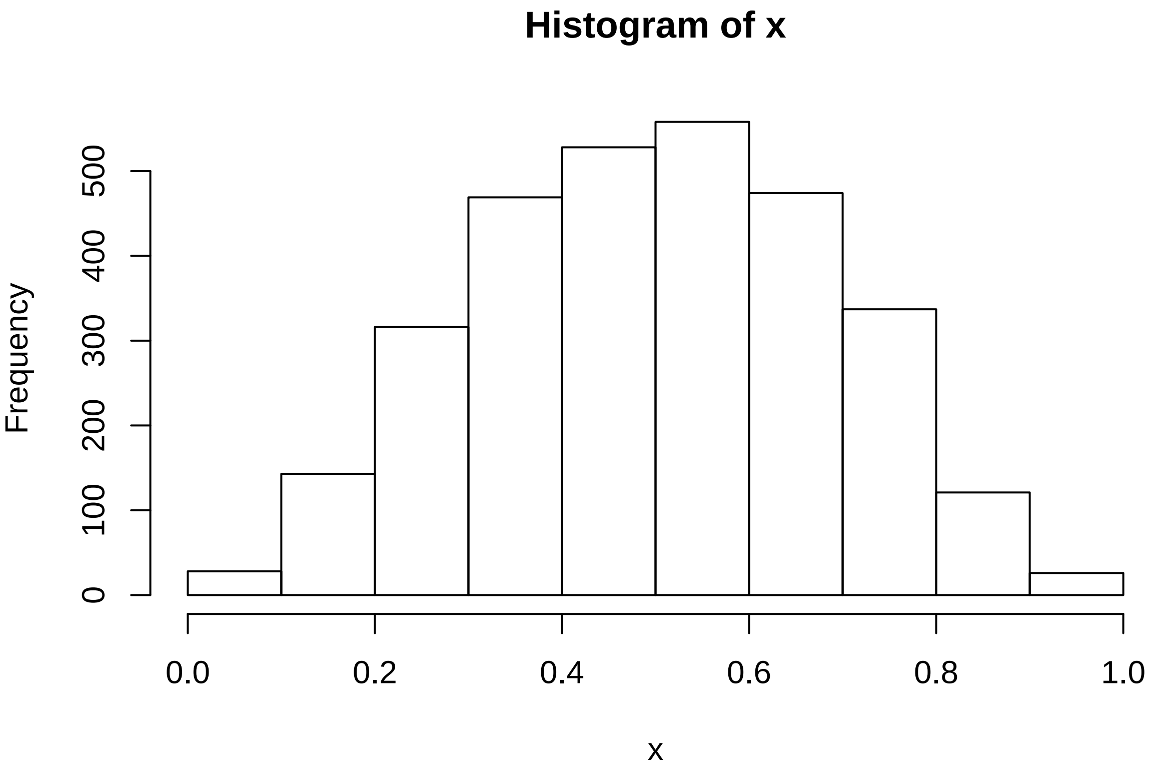
```
mean(x)
```

```
## [1] 9.4756
```

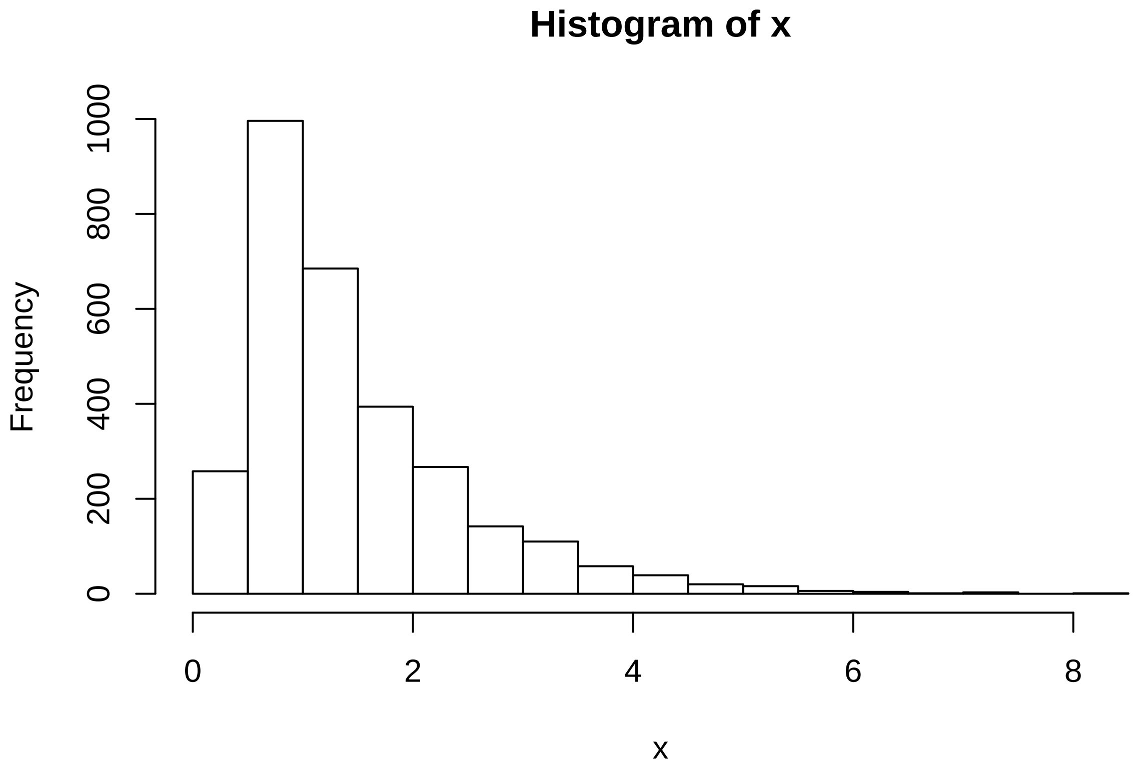
```
###################################################
## 3.simulate from two-sided exponential distribution
#  output: x ~ F(x) = 0.5*exp(2*x)*(x<=0) + (0.5*exp(2*x)+0.5)*(x>0)
rbexp <- function(n){
  if(n < 0){stop("invalid input")}
  sapply(runif(n),function(u){
    0.5*log(2*u)*(u<=0.5)-0.5*log(2*(1-u))*(u>0.5)
  })
}
x = rbexp(10000)
hist(x)
```



```
####################################################
## 4. simulate x ~ f(x), f is continuous
#  interval:  01              x > 0 && x < 1
#             real_positive   x > 0
#             real            x is arbitrary real number
#  method:    "inverse"       FUN = F^(-1)(x)
#             "acceptance"    FUN = f
rcont <- function(n,FUN,interval = c("01","real_positive","real"),method = c("inverse","acceptance")){
  if(n < 0 | !(is.function(FUN))) {
    stop("invalid input argument")
  }
  interval = match.arg(interval)
  #if necessary we can use more accurate maximization algorithms to calculate c (say gradiant method).
  c <- if (interval == "01"){max(FUN(linspace(0,1,100000)))}
  else if (interval == "real_positive"){max(FUN(rexp(10000000)))}
  else if (interval == "real"){max(FUN(rnorm(10000000)))}
  method = match.arg(method)
  result <- if (method == "acceptance") {
    i = rep(0,n)
    sapply(i,function(y){
      if (interval == "01"){
        repeat{
          u = runif(1);v = runif(1)
          if(u < (FUN(v)/c)){return(v)}
        }
      }
      else if (interval == "real_positive"){
        repeat{
          u = runif(1);v = rexp(1)
          if(u < (FUN(v)/c/dexp(v))){return(v)}
        }
      }
      else if (interval == "real"){
        repeat{
          u = runif(1);v = rnorm(1)
          if(u < (FUN(v)/c/dnorm(v))){return(v)}
        }
      }
    })
  }
  else if(method == "inverse"){
    sapply(runif(n),FUN)
  }
}
#  in this case f(x) = 30*(x^2-2*x^3+x^4)
f4 <- function(t){30*(t^2-2*t^3+t^4)}
x = rcont(3000,f4,"01",method = "acceptance")
hist(x)
```



```
###################################################
##5. in this case f(x) = 0.5*x^2*exp(-x)
f5 <- function(x){0.5*x^2*exp(-x)}
x = rcont(3000,f5,"real_positive",method = "acceptance")
hist(x)
```


Histogram of x
Frequency
0
200
400
600
800
1000
0
2
4
6
8
x