HW 1

임효진

April 16, 2021

1

(a)

```
cdf=function(x){
  sum=x
  temp=x
  k=1
  while(abs(temp)>.Machine$double.eps){
    temp=(-1)^k*x^(2*k+1)/(factorial(k)*2^k*(2*k+1))
    sum=sum+temp
    k=k+1
  }
  return((1/sqrt(2*pi))*sum)
}

cdf(1)+0.5
```

```
## [1] 0.8413447
```

```
pnorm(1)
```

```
## [1] 0.8413447
```

```
0.5+cdf(-1)
```

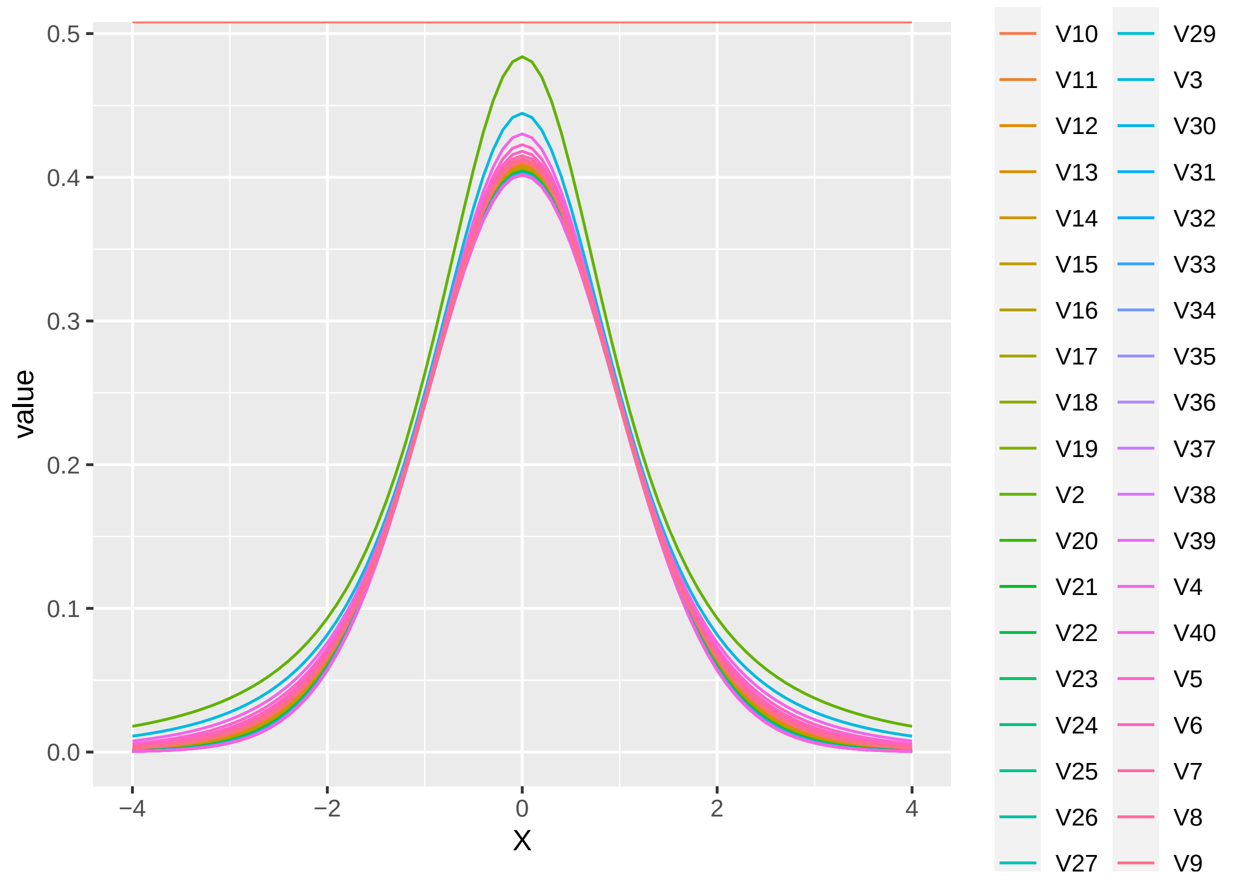
```
## [1] 0.1586553
```

```
pnorm(-1)
```

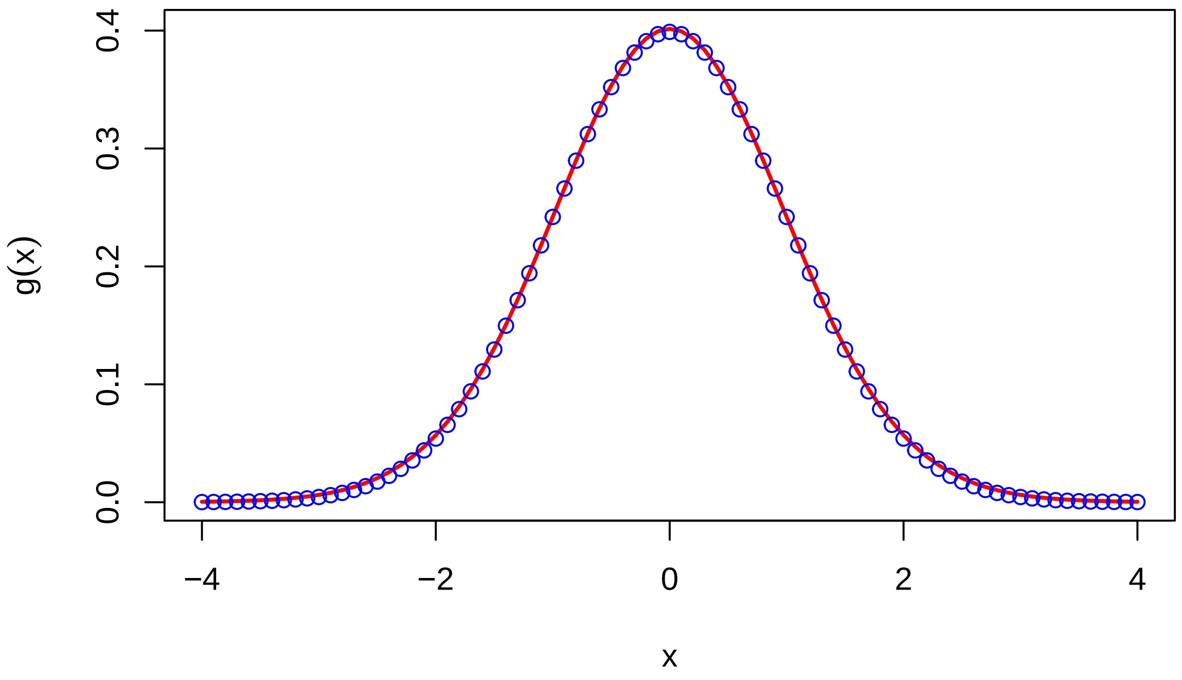
```
## [1] 0.1586553
```

(b)

```
library(dplyr)
library(tidyverse)

g=function(x, n){
  a=sqrt(2*pi*n/2)*(n/2/exp(1))^(n/2)
  b=(n*pi)^(1/2)
  c=sqrt(2*pi*(n-1)/2)*((n-1)/2/exp(1))^((n-1)/2)
  d=(1+(x^2)/n)^(-(n+1)/2)

  (a*d)/(b*c)
}
x=seq(-4, 4, by=0.1)

df=as.data.frame(sapply(1:40, function(n) g(x, n)))
df %>% mutate(X=x) %>%
  gather(key, value, 1:40) %>%
  ggplot(aes(X, value, col=key))+
  geom_line()
```



```
plot(x, g(x, 40), xlab="x",
     ylab=expression(g(x)), col="red", lwd=2, type="l")
lines(x, dnorm(x), col="blue", type="p")
```



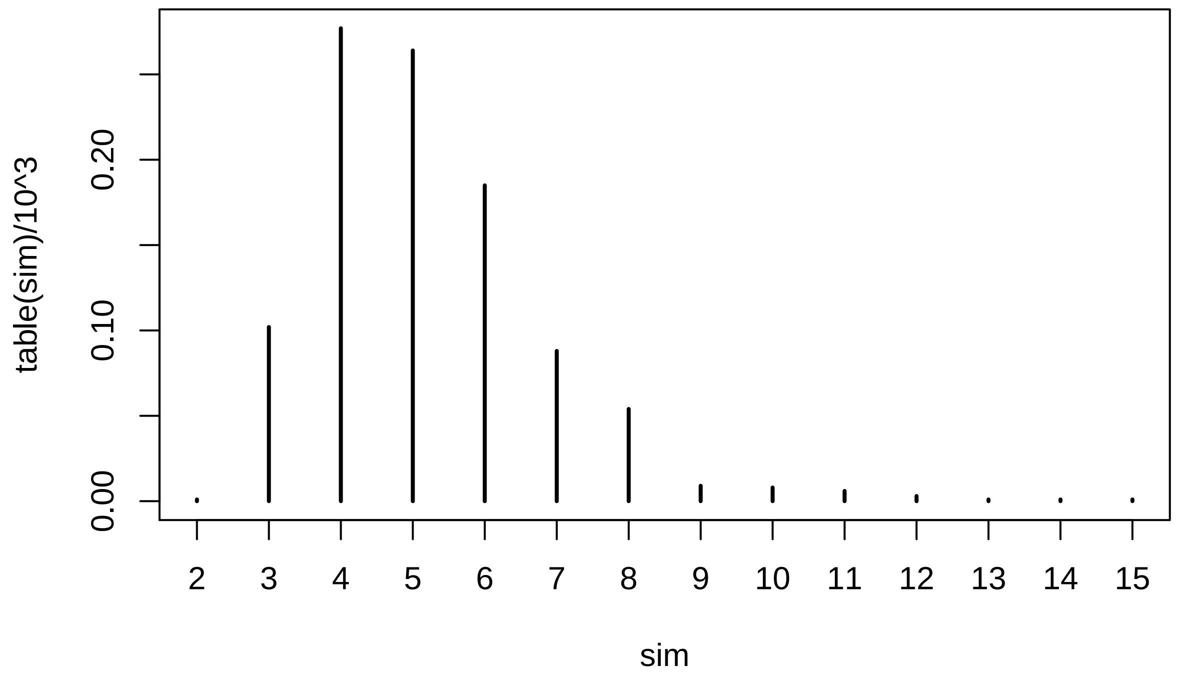
We can see that as n increases the distribution gets closer to the standard normal distribution.

2

```
library(dplyr)
seqTail=function(n){
  run=rle(rbinom(n, size=1, prob=0.5))
  x1=run$values
  x2=run$lengths
  x=data.frame(x1, x2)
  x %>% filter(x1==1) %>% max(x2)
}

sim=replicate(10^3, seqTail(30))
table(sim)
```

```
## sim
##    2   3   4   5   6   7   8   9  10  11  12  13  14  15
##    1 102 277 264 185  88  54   9   8   6   3   1   1   1
```

```r
plot(table(sim)/10^3)
```



3

##(a) $P(A_1) = P(A_2) = P(A_3)$

```r
round(choose(40, 10)/choose(60, 10), 6)
```

```
## [1] 0.011243
```

(b)

$P(A_1) \cap P(A_2) = P(A_1) \cap P(A_3) = P(A_2) \cap P(A_3)$

```r
round(choose(20, 10)/choose(60, 10), 6)
```

```
## [1] 2e-06
```

(c)

```r
round(3*(choose(40, 10)/choose(60, 10))-
  3*(choose(20, 10)/choose(60, 10)), 6)
```

```
## [1] 0.033722
```

##(d)

```
ball=function(){
  choose=sample(1:3, 10, replace = T)
  res=0
  if(length(unique(choose))!=3){
    res=1
  }
  return(res)
}

sim=replicate(10^3, ball())
mean(sim)
```

```
## [1] 0.056
```

4

##(a)

```
aWin=sample(c(-1, 1), size=1000, replace = T)
bWin=0-aWin

m=rbind(aWin, bWin)
m[, 1:10]
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
## aWin   -1    1   -1   -1   -1   -1   -1   -1   -1    -1
## bWin    1   -1    1    1    1    1    1    1    1     1
```

(b)

```
# using for loop
system.time({
  cumA=numeric(10^3)
  cumA[1]=aWin[1]
  for(i in 2:10^3){
    cumA[i]=cumA[i-1]+aWin[i]
  }
})
```

```
##    user  system elapsed
##    0.02    0.00    0.01
```

```
cumA[1:10]
```

```
##  [1] -1  0 -1 -2 -3 -4 -5 -6 -7 -8
```

```r
# using built-in function
system.time({cumsum(aWin)})
```

```
##    user  system elapsed 
##       0       0       0
```

```r
cumsum(aWin)[1:10]
```

```
##  [1] -1  0 -1 -2 -3 -4 -5 -6 -7 -8
```

(c)

```r
# using for loop
system.time({
  cumB=numeric(10^3)
  cumB[1]=bWin[1]
  for(i in 2:10^3){
    cumB[i]=cumB[i-1]+bWin[i]
  }

  b=cumB[1]
  for(i in 2:10^3){
    if(cumB[i]>b){
      b=cumB[i]
    }
  }
})
```

```
##    user  system elapsed 
##    0.03    0.00    0.03
```

```r
b
```

```
## [1] 35
```

```r
# using built-in function
system.time(max(cumsum(bWin)))
```

```
##    user  system elapsed 
##       0       0       0
```

```r
max(cumsum(bWin))
```

```
## [1] 35
```

5

(a)

```r
x=vector()
n=1L
for(i in 1:1750){
    if(i%%2!=0&i%%3!=0&i%%7!=0){
    x[n]=i
    n=n+1
    }
}
h=matrix(x, 100, 5, byrow = F)

library(psych)
a1=apply(h, 1, geometric.mean)
a1
```

```
##   [1] 205.3197 284.0642 334.4660 346.8327 367.4462 376.7899 394.0178 402.0959
##   [9] 415.5293 422.6173 440.5364 450.8362 456.9919 466.3843 480.2425 485.1193
##  [17] 493.8405 498.4728 507.8949 512.6938 520.2207 524.8522 536.2634 542.8972
##  [25] 547.5394 553.9232 563.9663 567.7523 574.3787 578.0799 585.7860 589.7925
##  [33] 595.7268 599.6663 609.0449 614.4156 618.5371 623.7911 632.3268 635.6708
##  [41] 641.4035 644.7051 651.6545 655.3023 660.4870 664.0990 672.4750 677.1994
##  [49] 681.0591 685.7159 693.4408 696.5464 701.7808 704.8612 711.3852 714.8283
##  [57] 719.5764 722.9973 730.7746 735.1051 738.8082 743.0947 750.3130 753.2703
##  [65] 758.1863 761.1268 767.3788 770.6896 775.1521 778.4477 785.8269 789.8921
##  [73] 793.4914 797.5256 804.3979 807.2542 811.9492 814.7937 820.8572 824.0754
##  [81] 828.3369 831.5441 838.6394 842.5138 846.0396 849.8911 856.5123 859.2958
##  [89] 863.8283 866.6031 872.5285 875.6784 879.7909 882.9325 889.8154 893.5462
##  [97] 897.0172 900.7303 907.1612 909.8899
```

(b)

```r
a2=apply(h, 1, harmonic.mean)
a2
```

```
##   [1]   4.970556  24.283752  51.675702  60.426513  77.392459  85.633499
##   [7] 101.681314 109.507342 124.639032 132.078785 153.518034 167.159668
##  [13] 173.971685 186.982773 205.854110 212.035835 224.062382 230.012153
##  [19] 241.808849 247.667119 258.727723 264.405724 280.535291 290.782328
##  [25] 296.200231 306.108789 320.762831 325.692364 335.218441 340.022991
##  [31] 349.700570 354.580228 363.447044 368.228240 381.469020 389.798763
##  [37] 394.492003 402.616349 414.846310 419.069725 427.135773 431.283978
##  [43] 439.749418 444.070417 451.604507 455.865605 467.340149 474.465047
##  [49] 478.735740 485.725134 496.417785 500.202345 507.330449 511.066092
##  [55] 518.769654 522.739566 529.395117 533.325613 543.629320 549.938919
##  [61] 553.942077 560.156718 569.804291 573.296461 579.779528 583.237993
##  [67] 590.428690 594.161800 600.202927 603.908519 613.388822 619.115994
##  [73] 622.938929 628.596459 637.495966 640.782892 646.799239 650.061889
##  [79] 656.888919 660.453473 666.045228 669.589663 678.464310 683.757873
##  [85] 687.453396 692.694041 701.036377 704.173268 709.838434 712.957214
##  [91] 719.515683 722.955098 728.206499 731.630659 740.043894 745.003997
##  [97] 748.605908 753.524646 761.438719 764.462322
```

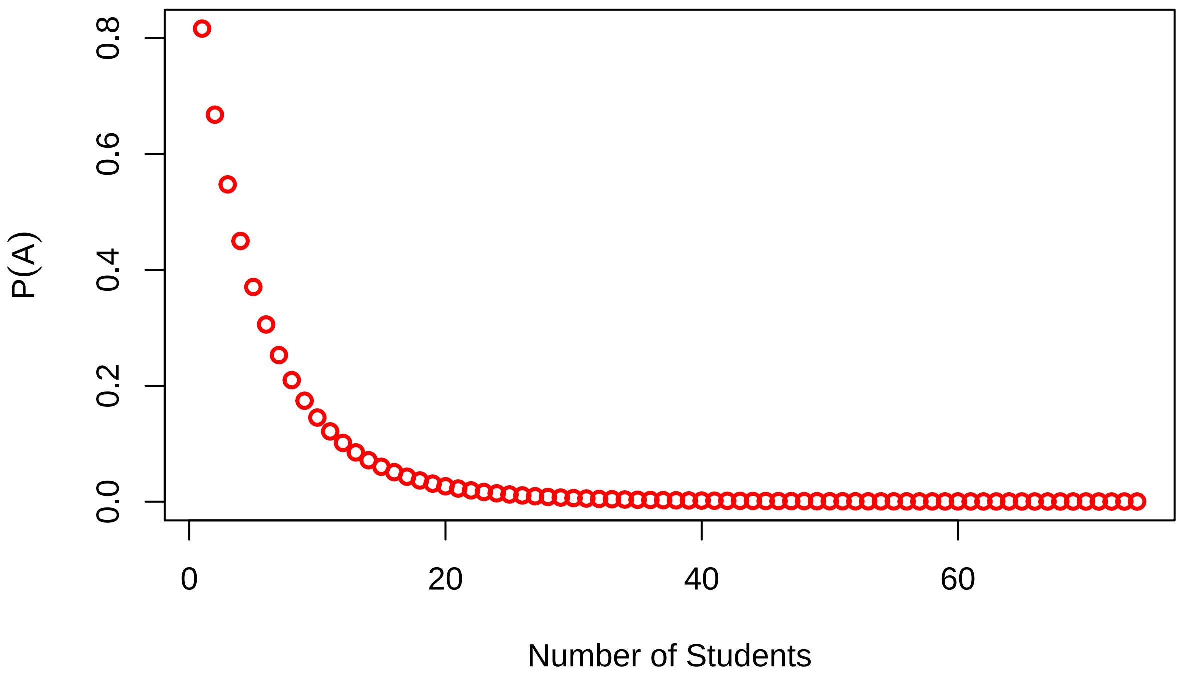
```
a2<=a1
```

```
##   [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
##  [16] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
##  [31] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
##  [46] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
##  [61] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
##  [76] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
##  [91] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

6

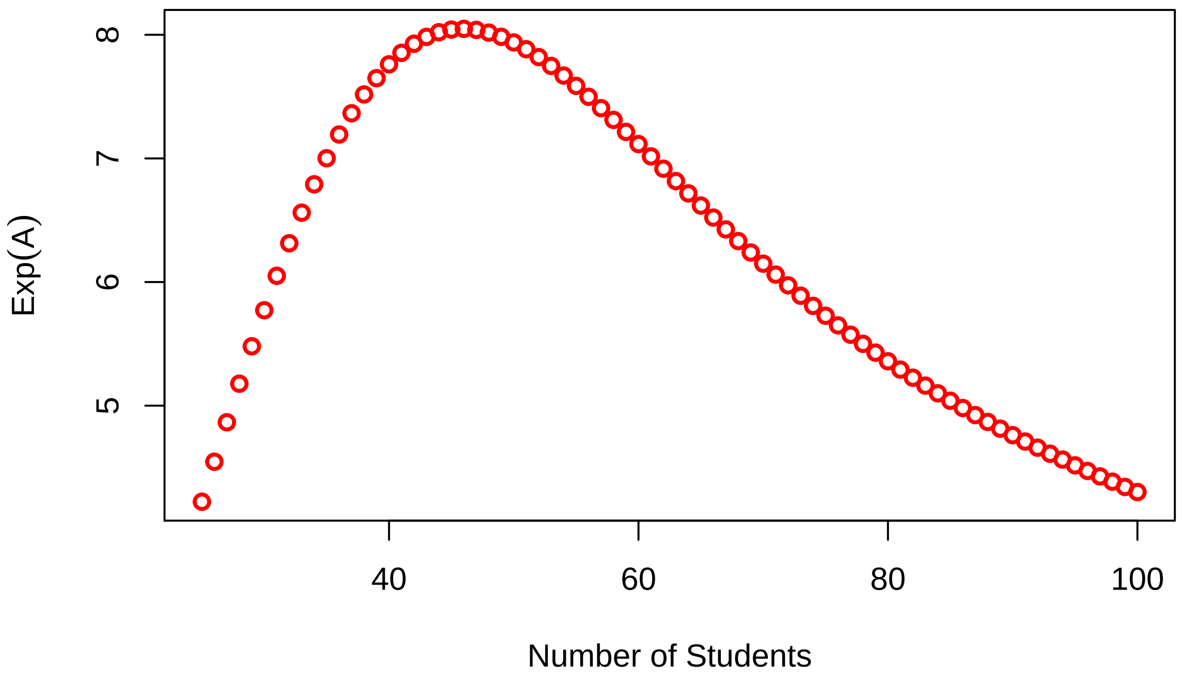
(a)

```
pbday=function(n, x){
  temp=lfactorial(365)+(n-x)*log(365-x)-
    lfactorial(365-x)-n*log(365)
  return(exp(temp))
}

nstudents=1:74
prob=pbday(75, nstudents)
plot(nstudents, prob, xlab="Number of Students",
     ylab=expression(P(A)), col="red", lwd=2)
```



(b)

```
meanPbday=function(n){
  x=c(1:n)
  sum=sum(x*(pbday(n,(x-1))-pbday(n,x)))
  return(sum)
}

nstudents=25:100
temp=numeric(length(nstudents))
for (i in nstudents){
  temp[i-24]=meanPbday(i)
}

plot(nstudents, temp, xlab="Number of Students",
     ylab=expression(Exp(A)), col="red", lwd=2)
```



(c)

```r
simPbday=function(n){
  temp=sample(365, n, replace = T)
  idx=0
  for(i in 2:n){
    if(temp[i]%in%temp[1:i-1]){
      idx=i
      break
    }
  }
  return(idx)
}

sim=replicate(10^3, simPbday(75))
mean(sim)
```

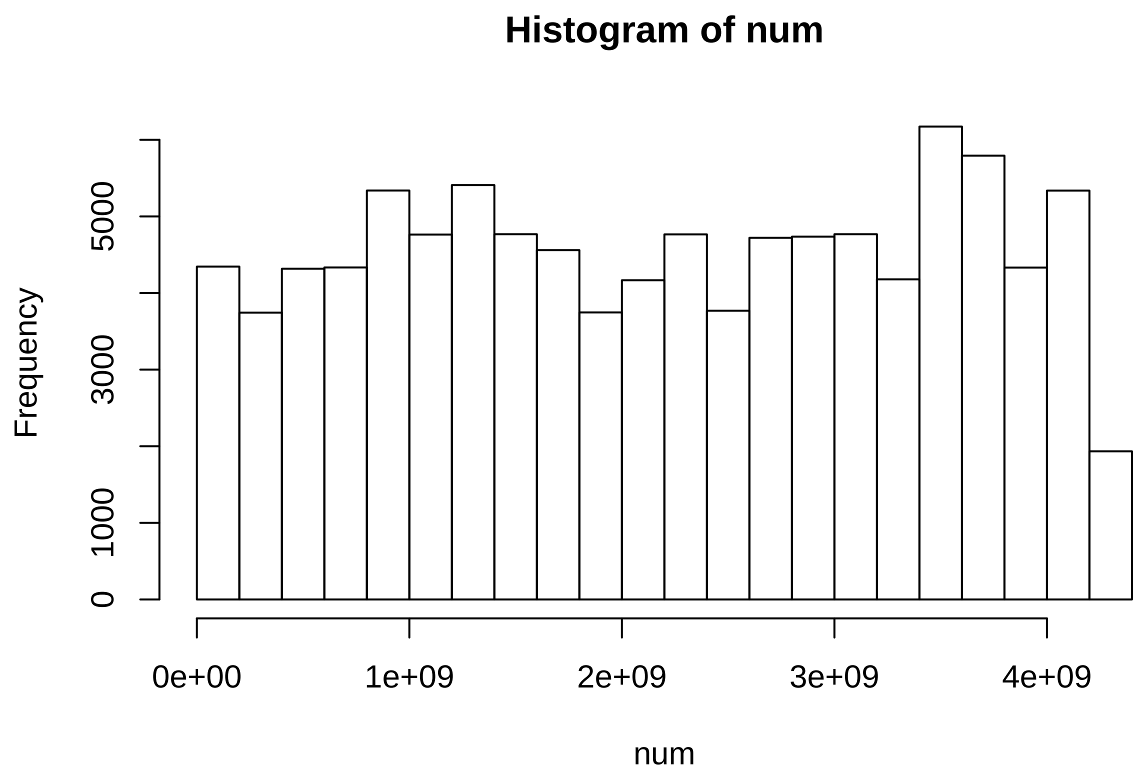
```
## [1] 24.87
```

7

```
LCG=function(n, m, a, c, x0){
  x=numeric(n)
  xn=x0
  for(i in 1:n){
    xn=(a*xn+c)%%m
    x[i]=xn
  }
  return(x)
}

num=LCG(10^5, 2^32,  1103515245, 12345, 1)
length(unique(num))
```

```
## [1] 15595
```

```
hist(num, nclass=30)
```

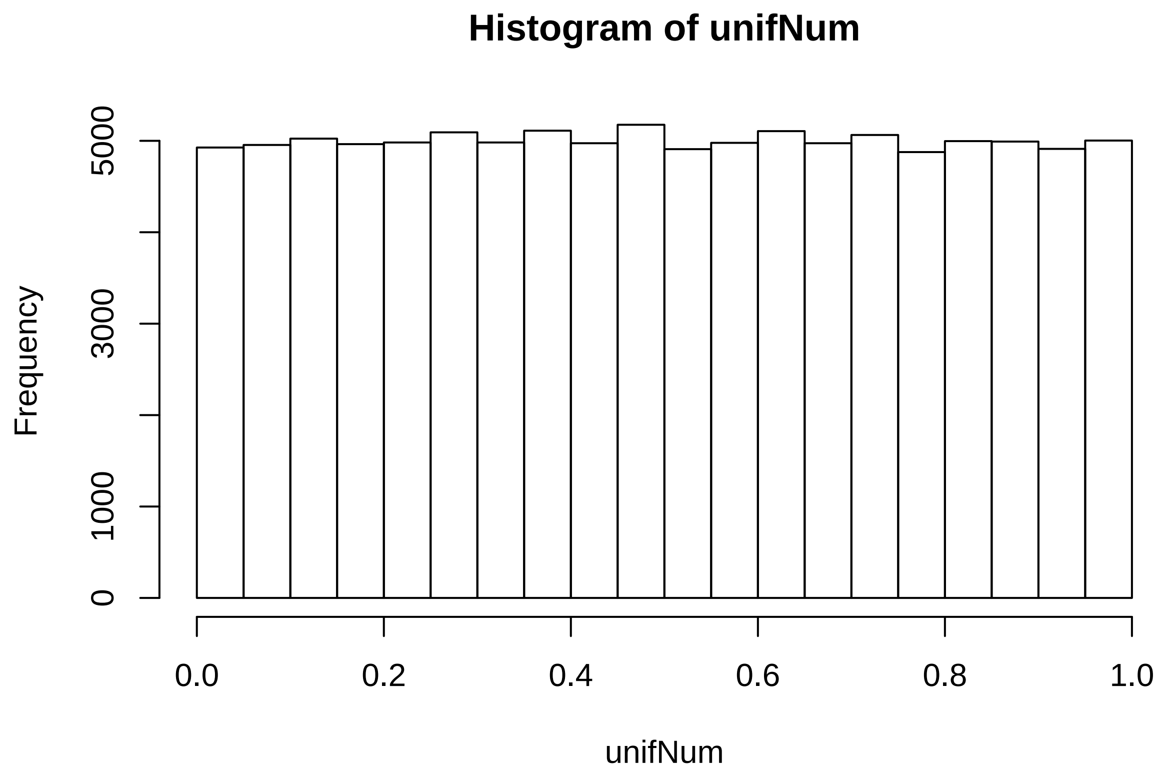


(b)

```
unifNum=runif(10^5)
length(unique(unifNum))
```

```
## [1] 100000
```

```
hist(unifNum, nclass=30)
```



(c)

```
# 15595 unique values are achieved from LCG.

expUnique=function(n, m, a, c, x0){
  num=LCG(n, m, a, c, x0)
  return(length(unique(num)))
}

sim=replicate(10,
              expUnique(10^5, 2^32,  1103515245, 12345, 1))
mean(sim)
```

```
## [1] 15595
```

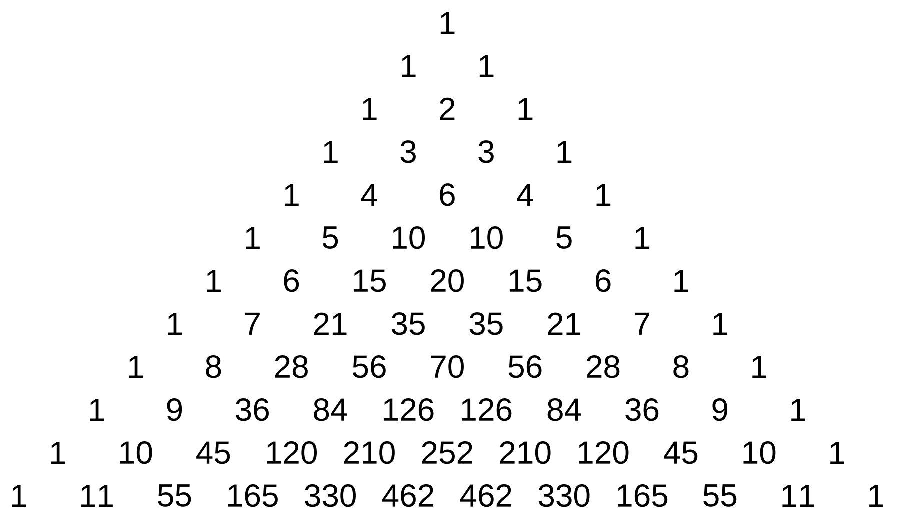
8

(a)

```r
returnPascal=function(n){
  return(choose(n, 0:n))
}
```

##(b)

```r
cnt=0
plot(0, 0, type="n", xlim=c(-12, 12), ylim=c(1, 12), axes = F,
     xlab = "", ylab = "")
for(i in 1:12){
  for(j in 1:i){
    text(-(i+1)+2*j, (12+1)-i, label=returnPascal(cnt)[j])
  }
  cnt=cnt+1
}
```



```
                                  1
                              1       1
                          1       2       1
                      1       3       3       1
                  1       4       6       4       1
              1       5      10      10       5       1
          1       6      15      20      15       6       1
      1       7      21      35      35      21       7       1
  1       8      28      56      70      56      28       8       1
1       9      36      84     126     126      84      36       9       1
1      10      45     120     210     252     210     120      45      10       1
1      11      55     165     330     462     462     330     165      55      11       1
```