Computational Statistics Exam

Mowniesh Asokan(mowas455)

1/12/2021

Assignment 1

Q1.1 Factorial Fucntion

```r
mainfn<-function(m,n,r)
{
  stopifnot((m+n)<=r)
  n1=factorial(m+1)
  n2=factorial(n)
  n3=factorial(r-m-n)
  d1=factorial(r-2)
  s_m_n_r=(n1*n2*n3)/d1
  return(s_m_n_r)
}
mainfn(20,30,50)
```

```
## [1] 1.09168e-09
```

```r
mainfn(10,20,30)
```

```
## [1] 0.0003185221
```

```r
#mainfn(75,100,150)
#mainfn(200,250,290)
mainfn(100,150,250)
```

```
## [1] NaN
```

The executed function is able to handle medium level values of numbers, if it goes high the values are not able handle by the function and it may lead to the chances of overflow and underflow.

Q1.2 Experiment it Numerically

```r
s1<-((factorial(21)*factorial(25)*factorial(30-20-25))/factorial(30-2))
```

```
## Warning in gamma(x + 1): NaNs produced
```

```r
s<-((factorial(201)*factorial(200)*factorial(290-250-200))/factorial(290-2))
```

```
## Warning in gamma(x + 1): NaNs produced
```

```r
s1
```

```
## [1] NaN
```

```r
factorial(20)
```

```
## [1] 2.432902e+18
```

```r
factorial(200)
```

```
## [1] Inf
```

- No, the above implemented function is cannot calculate the correct values of m,n,r.For the larger values of m,n,r, the function can not computed.To find the factorial of all the values that leads to overflow. For example factorial of 200 is infinite and it can be represented in terms
- Gentle(2009) suggest that the numerical computational perspective a number with larger and a significant that represent the INF and if the case is indeterminate like zero by infinite, then it may be "NaN"

Q1.3

```r
mainfn<-function(m,n,r)
{
  stopifnot((m+n)<=r)
  n1=factorial(m+1)
  n2=factorial(n)
  n3=factorial(r-m-n)
  d1=factorial(r-2)
  s_m_n_r=(n1*n2*n3)/d1
  return(s_m_n_r)
}
mainfn(20,30,50)
```

```
## [1] 1.09168e-09
```

```r
mainfn(10,20,30)
```

```
## [1] 0.0003185221
```

```
mainfn(75,100,200)
```

```
## [1] 0
```

```
mainfn(200,250,700)
```

```
## [1] NaN
```

```
##Binomial function
bin<-function(m,n){
  n<-factorial(n)
  d<-(factorial(m))*(factorial(n-m))
  return(n/d)
}
cat("binomial(20,30)",bin(20,30))
```

```
## binomial(20,30) 0
```

To improve the numerical stability of the function , if we check the value if m+n is less than or equal to the r value. This execution will help to find the value for certain limit.Eventhough the number goes high that cannot find the factorial values then we can use binomial function to implement the binomial coefficient inside to reduce the overflow and catastropical complexity of the function.

Assignment 2

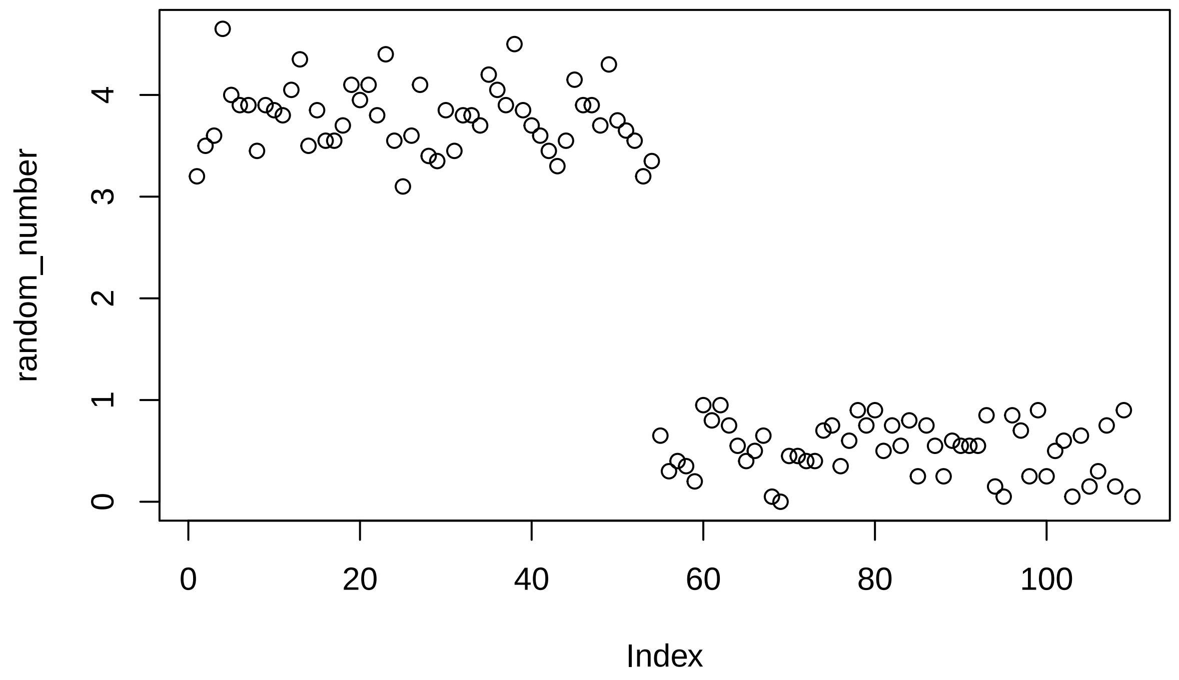
Q2.1

Implement the random number generator function.By using the binomial function it can evaluate the position of the values and generate a random numbers according to additive generation.In this case probability is mentioned as 0.5 to get value in a size of 150.

```
rand_generator<-function(m,k)
{
  x<-rbinom(54,150,0.5)
  for(i in 55:k)
  {
    x[i]<-(x[i-24]+x[i-54])%%m
  }
  return(x/m)
}
random_number<-rand_generator(20,110)
length(random_number)
```

```
## [1] 110
```

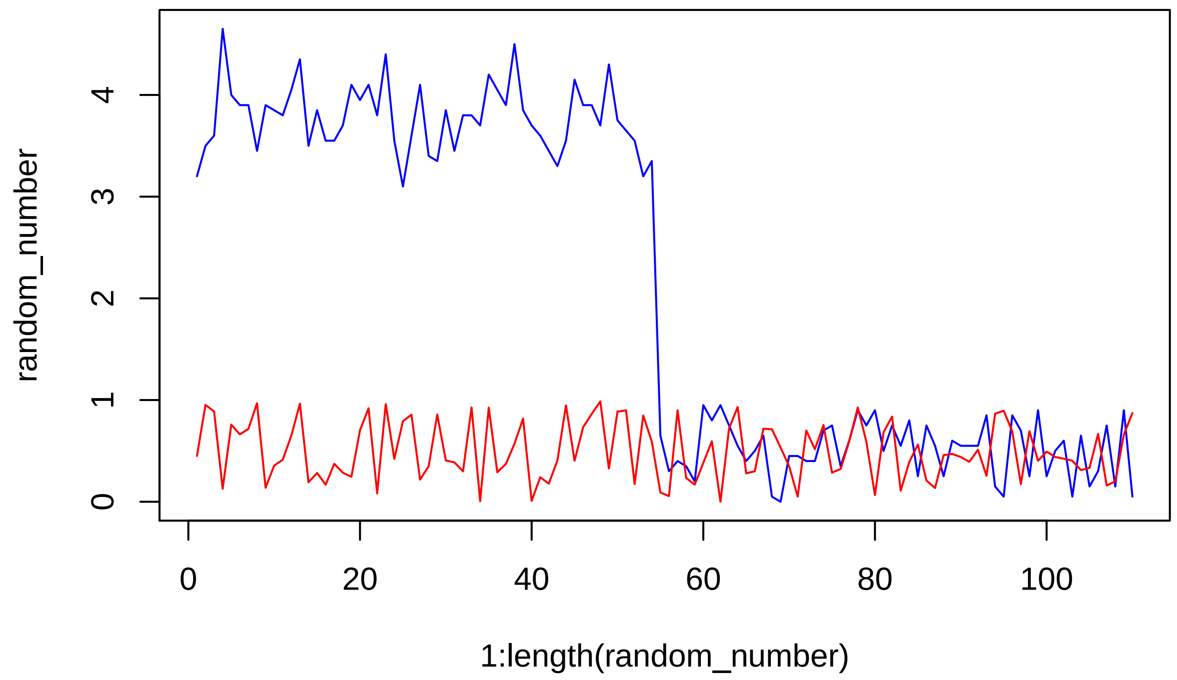
```
plot(random_number)
```



```
# hist(random_number)
```

Q2.2 Comparing the Generated values and uniform values

```
plot(1:length(random_number),random_number,
     type = "l",
     col = "blue")
lines(1:length(random_number),runif(length(random_number),0,1),col ="red")
```



The random numbers are generated by using the additive generator function are more upon particular limit of the given value of m.where the values are generated in random but that are lies between the 0 to 24 and 55 to 0 to 110. while i compare my additive generator function with uniform number generator function it can generate the continuous value between o to 1.This plot represent the numbers are generated in certain limit to obey our function.

Q2.3

Generate a random sample when$ X0, . . . , X54$ are all even and another one when they are not. Investigate, compare both to the uniform on [0, 1] distribution, you may use runif() function to generate uniform numbers forcomparison. Report on what you find.

```
rand_generator<-function(m,k)
{
  x<-rbinom(54,150,0.5)
  for(i in 55:k)
  {
    x[i]<-(x[i-24]+x[i-54])%%m
  }
  return(x/m)
}
random_number<-rand_generator(20,110)
random_number
```

```
##   [1] 4.00 4.40 3.50 3.65 4.05 3.25 4.20 3.65 3.80 3.70 3.85 4.05 3.75 3.90 3.55
##  [16] 3.75 4.05 4.35 3.90 3.65 3.45 3.80 3.40 3.70 3.85 4.40 3.35 3.35 4.45 3.45
##  [31] 3.65 4.00 3.75 3.60 3.25 3.95 4.15 3.30 3.45 3.40 4.00 3.80 4.20 3.60 3.85
##  [46] 3.75 4.45 3.90 3.85 4.00 4.10 3.40 3.95 3.80 0.65 0.40 0.25 0.25 0.30 0.20
##  [61] 0.35 0.95 0.25 0.10 0.85 0.85 0.95 0.50 0.40 0.50 0.50 0.25 0.75 0.65 0.55
##  [76] 0.20 0.35 0.50 0.50 0.80 0.60 0.60 0.75 0.65 0.00 0.95 0.00 0.70 0.10 0.80
##  [91] 0.10 0.80 0.85 0.90 0.50 0.05 0.95 0.25 0.40 0.95 0.80 0.40 0.35 0.80 0.70
## [106] 0.00 0.70 0.45 0.65 0.35
```

Mathematically saying the numbers are random that can be influenced by random values of linear congruential generator and this one lies between the values of 24 and 55 as probability of taking the random values into the generation.