Introduction to probability with R

References:

1) The art of R programming by Norman Matloff
2) Introduction to probability with R by Kenneth Baclawski
3) https://stephens999.github.io/fiveMinuteStats/markov_chains_discrete_stationary_dist.html

Sequence of numbers in R

```
> 1:49
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27
[28] 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49
```

Square of numbers

```
> (1:9)^2
[1]  1  4  9 16 25 36 49 64 81
> ((1:9)^2)[3]
[1] 9
```

```
>  (1:20)+10
 [1] 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
```

Recycling

```
> (1:5)+(1:10)
 [1]  2  4  6  8 10  7  9 11 13 15
```

Factorial

```
> factorial(6)
[1] 720
```

Natural logarithm to factorial function

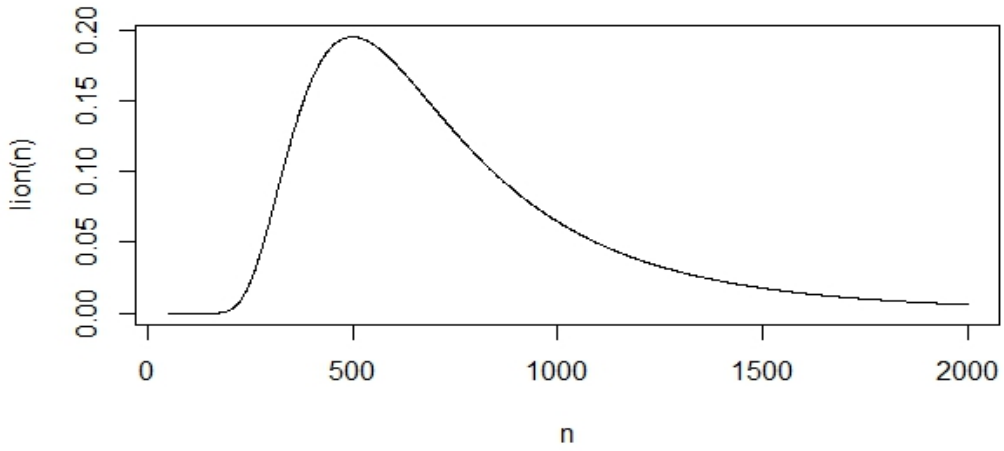
```
> lfactorial(5000)
[1] 37591.14
```

The value of $\binom{20}{10}$

```
> choose (20,10)
[1] 184756
```

Example: Let we want to estimate the lions in the Gir forest. First 50 lions are caught and tagged and released in the forest. Then again 50 lions are caught and there are 5 tagged lions. Plot the value of number of lions n with corresponding probability. Find the value of n for which the probability is highest.

```
> lion<-function(n) choose(50,5)*choose(n-50,45)/choose(n,50)
> n<-50:2000
> plot(n,lion(n), type='l')
> m<- max(lion(n))
> m
[1] 0.1948912
```



Cumulative Sums and Products

```
> x<-c(10, 6, 40)
> cumsum(x)
[1] 10 16 56
> cumprod(x)
[1]   10   60 2400
```

```
> M <- matrix(c(2:13), nrow = 4, byrow = TRUE)
> print(M)
     [,1] [,2] [,3]
[1,]    2    3    4
[2,]    5    6    7
[3,]    8    9   10
[4,]   11   12   13
> N <- matrix(c(2:13), nrow = 4, byrow = FALSE)
> print(N)
     [,1] [,2] [,3]
[1,]    2    6   10
[2,]    3    7   11
[3,]    4    8   12
[4,]    5    9   13
```

Minima and Maxima

```
> M <- matrix(c(1, 2, 3 ,5 , 6  ,2), nrow = 3, ncol=2)
> print(M)
     [,1] [,2]
[1,]    1    5
[2,]    2    6
[3,]    3    2
> min(M[,1], M[,2])
[1] 1
> pmin(M[,1], M[,2])
[1] 1 2 2
```

Function minimization/maximization

Use function nlm()

```
> nlm(function(x) return(x^3-cos(x)),9)
$minimum
[1] -9.255649e+13

$estimate
[1] -45234.41

$gradient
[1] 6138450044

$code
[1] 5

$iterations
[1] 6
```

Calculus

```
> D(expression(exp(x^3)),"x")#derivative
exp(x^3) * (3 * x^2)
> integrate(function(x) x^3,0,1)#integration
0.25 with absolute error < 2.8e-15
```

Sorting

```
> x<-c(5,11,5,9)
> sort(x)
[1]  5  5  9 11
> x
[1]  5 11  5  9
> order(x)
[1] 1 3 4 2
```

Linear Algebra operations

```
> y<-c(1,3,5,10)
> 2*y
[1]  2  6 10 20
> crossprod(c(1,2,3),c(2,4,6))
     [,1]
[1,]   28
```

Matrix multiplication

```
>  M <- matrix(c(1, 2, 3 ,5 ), nrow = 2, ncol=2)
> M
     [,1] [,2]
[1,]    1    3
[2,]    2    5
> N <- matrix(c(1,0,0,1 ), nrow = 2, ncol=2)
> N
     [,1] [,2]
[1,]    1    0
[2,]    0    1
> M%*%N
     [,1] [,2]
[1,]    1    3
[2,]    2    5
```

Solution of system of linear equations

```
> A <- matrix(c(1, 2, 3 ,5 ), nrow = 2, ncol=2)
> b<-c(1,2)
> solve(A,b)
[1] 1 0
> solve(A)
     [,1] [,2]
[1,]   -5    3
[2,]    2   -1
```

Set Operations

```
> x<-c(1,2,3)
> y<-c(4,5,6,3)
> union(x,y)
[1] 1 2 3 4 5 6
> intersect(x,y)
[1] 3
> 2%in%y
[1] FALSE
> choose(3,2)
[1] 3
```

Simulations in R

Statistical distributions

Distribution	pdf/pmf	cdf	Quantiles	Random numbers
Normal	dnorm()	pnorm()	qnorm()	rnorm()
Binomial	dbinom()	pbinom()	qbinom()	rbinom()
Chi square	dchisq()	pchisq()	qchisq()	rchisq()

```
> mean(rchisq(2000,df=4))
[1] 4.114423

> x<-rbinom(200000,5,0.5)
> mean(x>=4)
[1] 0.18875
```

Bernoulli process

```
> rbinom(20,1,1/2)
 [1] 1 0 0 0 0 1 0 1 0 1 0 0 0 0 0 1 1 0 1 0
> rbinom(5,1,1/2)==1
[1] FALSE  TRUE  TRUE FALSE  TRUE
> (1:15)[rbinom(15,1,1/2)==1]
[1] 1 2 3 5 8
```

Here 1 occur then TRUE when 0 occur then FALSE.

When an expression gets bigger and complicated, use name (variable)

```
> tails<-function(n,q) (1:n)[rbinom(n,1,q)==0]
> tails(10,3/4)
[1]  1  6  7  8  9 10
```

Geometric Distribution:

Waiting for first head, i.e., eg. $W_1(TTTTHHHTTTHH) = 5$

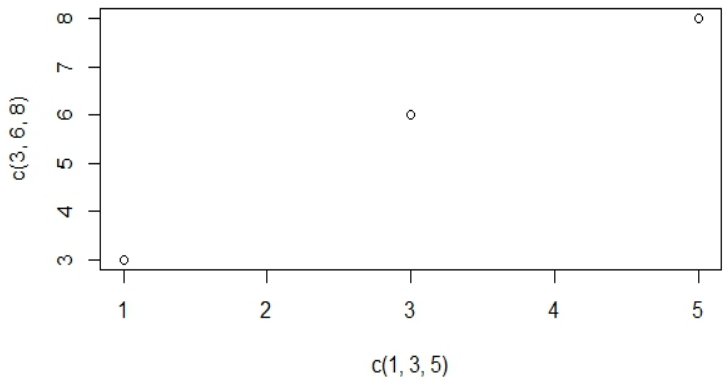
```
> value<-1:2000
> sample<-rbinom(2000,1,0.1)
>  value[sample==1][1]
[1] 6
```

Function so that we can run the program more than once

```
> w1<-function()(1:2000)[rbinom(2000,1,0.1)==1][1]
> w1()
[1] 6
> w1()
[1] 8
> replicate(15,w1())
 [1] 15 17 28  2 61  9 10  4  8 13  5 10  7  1 12
```

Graphics:

```
> plot(c(1,3,5),c(3,6,8))
```



Linear regression

```
> x<-c(1,3,5)
> y<-c(3,6,8)
> lmout<-lm(y~x)
> abline(lmout)
```