Tarea2Algebra_VictorGomez

September 27, 2019

1 Tarea2 Algebra Matricial, Victor Manuel Gómez Espinosa

2 1.- Determinante

Programa para calcular el determinante.

Datos de entrada: A matriz de nxn (no es necesario ingresar el renglon o columna, automáticamente elije el que tenga mas ceros o por default el primer renglon)

Salidas: Imprime los valores de los cofactores de la matriz principal que utilizó para calcular el determinante y regresa el valor del Determinante por metodo de los cofactores

2.1 Instrucciones:

2.2 1.1 se ejecuta el siguiente bloque de código

```
In [14]: check_mat<-function(A){#checa para cuando la matriz no es cuadrada, hay columnas o re
             di<-dim(A)
             n<-di[1] #renglones
             m<-di[2] #columnas
             ol<-matrix(1,1,m) #vector renglon de 1's
             or<-matrix(1,n,1) #vector columna de 1's
             rorcr<-0 #inicia en cero el contador de columnas o renglones repetidos
             if(n!=m){#si no es matriz cuadrada
                 cat("Error! Las dimensiones son diferentes \n")
                 d<-(-1)
             }else{#checar si hay renglones o columnas completos con ceros
                 mz<-A==0 #donde hay ceros
                 colz<-(ol%*%mz)/n  #promedio columnas
                 renz<-(mz%*%or)/m  #promedio renglones
                 colz<-sum(colz>=1) #mas de una columna con ceros?
                 renz<-sum(renz>=1) #mas de un renglon con ceros?
                 if(colz>=1|renz>=1){#hay ceros en columnas o renglones completos?
                     d<-0
                 }else{#checa si hay columnas o renglones iguales
                     i<-1
                     while(rorcr==0&i<=n){

                         #por renglones
```

```
                mrepr<-A==A[i,]
                rengr<-(mrepr%*%or)/m
                rengr<-sum(rengr>=1)-1

                #por columnas
                mrepc<-A==A[,i]
                colr<-(ol%*%mrepc)/n
                colr<-sum(colr>=1)-1
                i<-i+1
                if(rengr>=1|colr>=1){
                    rorcr<-1
                    d<-0
                }

            }



        }

    }
    d<-1#si no se cumplió ninguna de las condiciones
    return(d)

}

my_det<-function(A,ren=1,col=0,cont=0){#calcula el determinante, por default para ren


    ac<-0
    di<-dim(A)
    n<-di[1] #renglones
    m<-di[2] #columnas



    ol<-matrix(1,1,m) #vector renglon de 1's
    or<-matrix(1,n,1) #vector columna de 1's
    dc<-check_mat(A)#checa para cuando la matriz no es cuadrada, hay columnas o rengl
    if(dc==0){#filas o columnas con 0 o filas o columnas repetidas
        d<-0
    }else if(dc>0){#si no se cumple la condicion anterior
        if(n==2){#caso base
            d<-(A[1,1]*A[2,2])-(A[2,1]*A[1,2])
        }else{
            #determina fila o columna para calcular el determinante (con mas ceros)
            mz<-A==0 #donde hay ceros
            colz<-(ol%*%mz)/n  #promedio columnas
```

```
renz<-(mz%*%or)/m  #promedio renglones
mxc<-max(colz)
mxr<-max(renz)
if(mxr>mxc){#renglon con mas ceros
    i<-which(renz==mxr)
    j<-0
    ii<-which(A[i,]!=0)
}else if(mxr<mxc){#columnas con mas ceros
    i<-0
    j<-which(colz==mxc)
    ii<-which(A[,j]!=0)
}else{
    i<-ren
    j<-0
    ii<-which(A[i,]!=0)
}


if(i!=0){#por renglon
    for(j in ii){
        aij=A[i,j] #obtiene el elemento en la matriz principal
        c=(-1)^(i+j)
        Aij=A[-i,-j] #matrriz asociada al cofactor
        daij<-my_det(Aij,cont=cont+1)  #su determinante
        cij<-c*daij #cofactor
        ac<-ac+(aij*cij) #acumula para obtener el determinante


        if(cont==0){
           cat("i :",i,"j: ",j,"cofactor: ",cij,"\n" ) #imprime el cofact
        }

    }

}else if(j!=0){#por columna
    for(i in ii){
        aij=A[i,j] #obtiene el elemento en la matriz principal
        c=(-1)^(i+j)
        Aij=A[-i,-j] #matrriz asociada al cofactor
        daij<-my_det(Aij,cont=cont+1)  #su determinante
        cij<-c*daij #cofactor
        ac<-ac+(aij*cij) #acumula para obtener el determinante


        if(cont==0){
           cat("i :",i,"j: ",j,"cofactor: ",cij,"\n" ) #imprime el cofact
        }
```

```
                    }

                }
                d<-ac #guarda el valor a regresar


            }




        }

        return(d)

    }
```

2.3 Ejemplo:

matriz de ejemplo

```
In [15]: vec<-c(2,4,-6,4,3,7,-10,6,0,2,0,4,0,0,1,5) #llenar por columnas
         A<-matrix(vec,4,4) #matrix
         A
```

2	3	0	0
4	7	2	0
-6	-10	0	1
4	6	4	5

probando la función

```
In [16]: my_det(A) #mi funcion
         det(A) #funcion de r
```

```
i : 1 j:  1 cofactor:  84
i : 1 j:  2 cofactor:  -52


12
12
```

3 2.- Eliminación de Gauss y solución del sistema Ax=B

3.1 Funcion para hacer la factorizacion

Datos de entrada: A matriz de nxn
Salidas:
P (matriz de permutacion)
L
U

3.2 Instrucciones:

3.3 2.1 se ejecuta el siguiente bloque de código

```
In [17]: swaprows<-function(p,r1,r2){#devuelve la matriz de permutación
             di<-dim(p)
             n<-di[1] #renglones
             m<-di[2] #columnas
             #I<-diag(x=1,n,m) #Identity matrix
             vt<-matrix(0,1,m) #vector renglon de 0's
             if(r1!=r2){
                 vt=I[r1,] #vector temporal
                 p[r1,]=I[r2,] #intercambia renglones
                 p[r2,]=vt
             }
             return(I)

         }

         lowerZ<-function(A,j){#devuelve una matriz de ceros excepto los elementos debajo del
             di<-dim(A)
             n<-di[1] #renglones
             m<-di[2] #columnas
             z<-matrix(0,n,m) #vector renglon de 1's
             i<-1:j
             z[-i,j]<-A[-i,j] #selecciona elementos debajo del pivote
             return(z)

         }

         my_lu<-function(A){#resuleve
             di<-dim(A) #dimensiones
             n<-di[1] #renglones
             m<-di[2] #columnas
             if(n==m){
                 z<-matrix(0,n,m) #vector renglon de 0's
                 l<-z #inicializa matrices y vectores
                 u<-z
                 tk<-z
```

```
        li<-z
        I<-diag(x=1,n,m) #Identity matrix
        p<-I
        i<-0
        j<-0
        pivot<-0

        #realiza el esscalonamiento por lu
        for(j in 1:n){
            pivot<-A[j,j] #pivote

            if(pivot!=0){#sin hacer cambios de renglon
                tk<-lowerZ(A,j)/pivot  #obtiene los elementos debajo del pivote
                li<-I-tk #matriz del tipo 3
                A<-li%*%A #se eliminan los elementos debajo del pivote

            }else if(pivot==0){#cambios de renglon (el de mayor magnitud)
                r1<-j
                r2<-which(abs(A[,j])==max(abs(A[,j]))) #encuentra el renglon a interc
                p<-swaprows(p,r1,r2)
                A<-p%*%A#intercambia los renglones

                pivot<-A[j,j]

                if(pivot!=0){
                    tk<-lowerZ(A,j)/pivot
                    li<-I-tk
                    A<-li%*%A
                }



            }

            z<-z+tk #acumula los elementos debajo del pivote

        }
        l<-I+z #obtiene la matriz l
        u<-A #u
        vres <- list(p,l, u)

        return(vres)

    }else{
        cat("Error! Las dimensiones son diferentes \n")
    }

}
```

3.4 Ejemplo:

Probando la funcion para obtener la matriz L y U

```
In [18]: listt<-my_lu(A)
         p<-listt[[1]] #obtiene P
         p
         l<-listt[[2]] #obtiene L
         l
         u<-listt[[3]] #obtiene U
         u
```

1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1
1	0	0	0
2	1	0	0
-3	-1	1	0
2	0	2	1
2	3	0	0
0	1	2	0
0	0	2	1
0	0	0	3

3.5 Funcion para resolver el sistema Ax=B

Datos de entrada: L y U matrices de nxn
B vector renglon 1xn
Salidas:
x vector renglon 1xn

3.6 Instrucciones:

3.7 2.2 se ejecuta el siguiente bloque de código

```
In [19]: my_forwards<-function(l,b){#sustitucion hacia adelante
             n<-length(b)
             y<-matrix(0,1,n) #vector renglon de 0's

             y[1]<-b[1]

             for(i in 2:n){
                 suma<-0
                 for( j in 1:(i-1)){
                     suma<-suma+(l[i,j]*y[j])

                 }
                 y[i]<-b[i]-suma
```

```
    }
    return(y)
}

my_backs<-function(u,y){#sustitucion hacia atras
    n<-length(y)
    x<-matrix(0,1,n) #vector renglon de 0's

    x[n]<-y[n]/u[n,n]


    i<-n-1
    while(i>=1){

        suma<-0
        for( j in 1:(i+1)){
            suma<-suma+(u[i,j]*x[j])

        }
        x[i]<-(1/u[i,i])*(y[i]-suma)

        i<-i-1
    }
    return(x)
}

my_solve<-function(l,u,b){#utilizando ambas
    y<-my_forwards(l,b)
    x<-my_backs(u,y)

    return(x)
}
```

3.8 Ejemplo:

Vector para probar resolver el sistema

```
In [20]: B<-c(1,2,1,0) #vector para resolver el sistema Ax=B
         B
```

1. 1 2. 2 3. 1 4. 0

Probando la función para resolver el sistema

```
In [21]: x<-my_solve(l,u,B) #solucion
         x
```

11.5 -7.333333 3.666667 -3.333333