1

1 2 3 ... N → Indicas una lista con N y otra lista con N

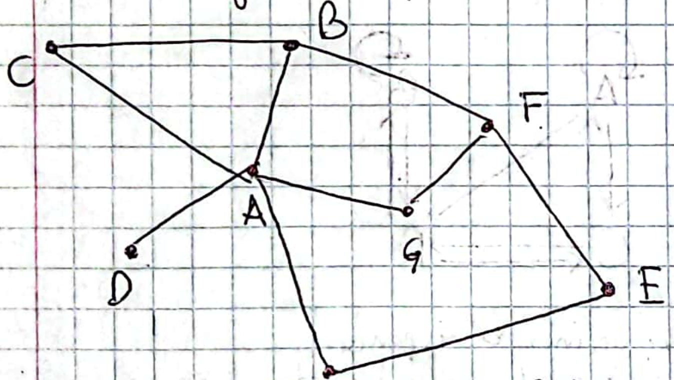
1
2
3
⋮
N

Lista 1	Lista 2
1	1
2	2
3	3
⋮	⋮
N	N

→ Ahora bajo una función al azar, generas una probabilidad o condición en la que si se cumple, entonces forma una unión o interacción
* Esto puede ser un número tal [0, 1], en el que $\lim_{x \to 1} f(x) \mid x \in [0,1]$
=> es mas probable que suceda, osea que si el número al caer en $< P$ => sucede interacción y si $> P$ entonces no sucede.

→ Ahora hay que saber que tan conectados estan entre si los nodos.

Tomemos el ejemplo de por acá



	A	B	C	D	E	F	G	H
A	0	1	1	1	0	0	1	1
B	1	0	1	0	0	1	0	0
C	1	1	0	0	0	0	0	0
D	1	0	0	0	0	0	0	0
E	0	0	0	0	0	1	0	1
F	0	1	0	0	1	0	1	0
G	1	0	0	0	0	1	0	0
H	1	0	0	0	1	0	0	0

Podemos considerar solo un lado, ahora para hacer calculos de clustering, considera cuanta nodos tiene conectados el nodo I y cuantos entre ellos estan conectados.

n = No. uniones entre vecinos del nodo I
k = Vecinos que conectan con el nodo I
C_I = Coeficiente de clustering

$$C_I = \frac{2n}{k(k-1)}$$

	A	B	C	D	E	F	G	H
A	0	1	1	1	0	0	1	1
B		0	1	0	0	1	0	0
C			0	0	0	0	0	0
D				0	0	0	0	0
E					0	1	0	1
F						0	1	0
G							0	0
H								0

$C_A = 0.1$ | $C_E = 0$
$C_B = 0$ | $C_F = 0$
$C_C = 0$ | $C_G = 0$
$C_D = 0$ | $C_H = 0$

D	M	A

¿Forma de ato analizar la conexión entre vecinos?

Si $A - B \wedge B - C \wedge A - C \Rightarrow$ triangle or $n = 1$

En secuencia si analiza 2 a la vez, ¿como resultaría?

- Pero en codigo no se puede, por ende nos serviríamos comparativas.

• Tenemos que para todos aquellos nodos que sean 1, serán usados para comparar aquellos nodos que hacen pareja consigo mismo, cuentan.

• Se harán comparaciones de B en 3

tomar A, B, C :

if $(A, B) = 1$ and $(B, C) = 1$ and $(A, C) = 1$:

$\Rightarrow$ triangle o están conectados.

Osea se hará una serie de combinaciones, de tal manera que se comparen entre todos, las combinaciones sean guardadas en cada nodo existente y que forme pareja.

Seudocodigo function (N, P)

Lista 1 = [ ]

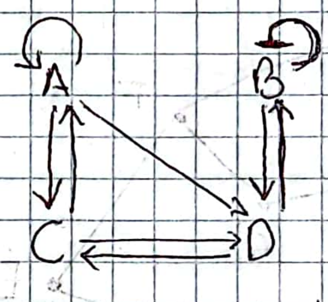
Lista 2 = [ ]

rango (N)

En este momento se me ocurrió, pensar que si se genera una matriz aleatoria, entonces tomaremos el ejemplo

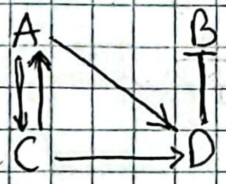
	A	B	C	D
A	1	0	1	1
B	0	1	0	1
C	1	0	0	1
D	0	1	1	0

Modelo visual



Solo hablaremos de si hay conexion entre vecinos o responder ¿Estan conectados? ¿Sí o No? Mas no especificar si es ~~dirigida~~ en una o otra direccion.

Así que si por ejemplo esta esto



	A	B	C	D
A	0	0	1	1
B	0	0	0	0
C	1	0	0	1
D	0	1	0	0

Contemplando que va de

Así :

A → N

B → N

C → N

D → N

Combinaciones

Cualquiera de las 2 cuenta como conexion entre vecinos incluso si tienen 2 ⇄ no importa, solo se contempla si ¡Sí o No!

3

Seudo codigo
funcion (N, P)
Lista 1 = [ ]
Lista 2 = [ ] } rango (N)

- Crear. dataframe (N)
for i in rango (N) :
    for j in rango (N) :
        random ( ) -> que de un numero entre [0, 1]
        if random ( ) < p :
            Nodo [1][1] = 1, ..., Nodo [1][N] = 1
            or Nodo [1][1] = -1, ..., Nodo [1][N] = -1
        else :
            Nodo [1][1] = 0, ..., Nodo [1][N] = 0

p.imprimir (Data.frame.aleatorio N( ))

count autoregulations -> count (Nodo [N][N] = 1 or -1)

for a in rango (N) :
    for b in rango (N) :
        for c in rango (N) : } Puede cambiar a while.
            if a ≠ b ≠ c :
                ~~Nodo~~ if (Nodo [a][b] and Nodo [b][c] and Nodo [a][c]) = 1
                triangle
                elif (Nodo [b][a] and Nodo [b][c] and Nodo [a][c]) = 1
                triangle
                elif etc ~~tenemos 3 combinaciones diferentes~~
                ~~cada una hace 2~~
                $3^2 = 9$

N
N+1
N+2

Permutaciones
• (a,b) • (a,c) • (b,c)
× (b,a) • (a,c) • (b,c)
• (a,b) • (c,a) • (b,c)
• (a,b) • (a,c) • (c,b)
~~(a,b)~~ (a,b) • (c,a) • (c,b)
× (b,a) • (
↓ Combinaciones de nu.

(a,b) • (a,c) • (b,c) — (a,b) • (c,a) • (b,c) — (a,b) • (a,c) • (c,b)
(a,b) • (c,a) • (c,b) — ~~(a,b) • (~~

(b,a) • (a,c) • (b,c) — (b,a) • (c,a) • (b,c) — (b,a) • (a,c) • (c,b)
(b,a) • (c,a) • (c,b)

Donde de una forma se condiciona a 2 formas que de (a,b) da 4 y de (b,a) otro
(C_{1}) (C_{2}) C 4, donde cada forma se llamara com f_1, f_2 f_8 ε
de tal manera qi fuera la secuen :

4

$f_1 = (a,b) \cdot (a,c) \cdot (b,c)$
$f_2 = (a,b) \cdot (a,c) \cdot (c,b)$
$f_3 = (a,b) \cdot (c,a) \cdot (b,c)$
$f_4 = (a,b) \cdot (c,a) \cdot (c,b)$
$f_5 = (b,a) \cdot (a,c) \cdot (b,c)$
$f_6 = (b,a) \cdot (a,c) \cdot (c,b)$
$f_7 = (b,a) \cdot (c,a) \cdot (b,c)$
$f_8 = (b,a) \cdot (c,a) \cdot (c,b)$

	f_1	f_2	, . . . ,	f_8
Intento 1	1	0		0
Intento 2	1	1		1
Intento 3	0	0		0

Condicion de an almenos 1, entonces se considera que es una conexion entre vecinos.

Intento 1 a, b, c
Intento 2 a, b, d

Recorrido
N Lista = [a, b, c, d] a b
N+1 Lista = [a, b, c, d] b c
N+2 Lista = [a, b, c, d] C d

Queda resuelto, ya que ahora se consideran combinaciones unicas, para una matriz que al menos 1, considerar del clustering

	f_1	f_2			f_8	neighthbours
a, b, c	1	0	0	, . . . ,	0	1
a, b, c	1	1	1	, . . . ,	1	1
a, b, c	0	0			0	0
⋮			⋮			

a Suma de ellos intentos

* Cuando N = b, pasaremos a buscar a todos intentos de a, que tienen a "b" y luego adicionarlos al Data frame de b para hacer lo mismo de calcular el n
* Cuando N = C pasamos a buscar en el Data frame de a, b los que tienen C para incluirlos al dataframe de C y continuar con el proceso como se hizo con a.

Pensandolo bien, no es mala idea los for, porque para a seria una matriz y para b, seria otra matriz y para C otra matriz y así hasta N

K sera calculada a partir de la matriz de N nodos al cuadrado
Sumas de vecinos en el nodo I

1 2 3 4 5 1

$N_1 = 1 \quad N_2 = 1 \quad N_3 = 1$

$N_1 = N_2 \Rightarrow N_0$

$(N_1 = 1 \; != N_2 = 2) \Rightarrow \checkmark$

$N \quad \checkmark \quad (N_1 \; != N_3)$

$(N_2 \; != N_3)$

N_1	N_2	N_3
1	1	1
2	2	2
3	3	3
4	4	4

$N_1 \neq N_2 \Rightarrow N_2 \neq N_3 \Rightarrow N_1 \neq N_3$

1 1   1 1   1 1

2 2   2 2   2 2

3 3   3 3   3 3

final

Siguendo la secuencia anterior

$N_1 \neq N_2 \neq N_3$

1 1 1

2 2 2

3 3 3

$N_1 \neq N_2 \Rightarrow N_2 \neq N_3 \Rightarrow N_1 \neq N_3$

1 1   1 1   1 1

2 2   2 2   2 2

3 3   3 3   3 3

$N_1 \neq N_2 \neq N_3$ Se cumple 3.f

1 1 1

2 2 2

3 3 3

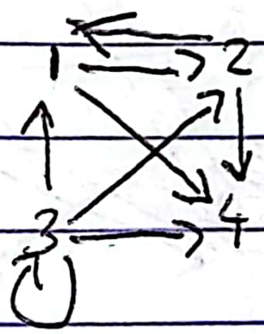
Primera instruccion

$N_1 \neq N_2 \neq N_3$

N_1	N_2	N_3
1	1	1
2	2	2
3	3	3

Ahora para el caso en el que suceda

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



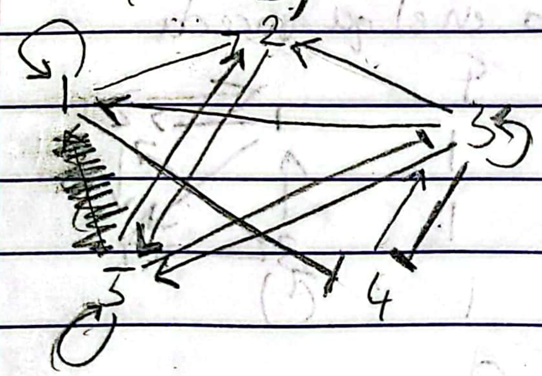
~~[1,1]=0~~ $\overset{1}{[1,2]}$ [1,3] [2,3

$[N_1, N_2]$, $[N_1, N_3]$, $[N_2, N_3]$

$[N_1, N_2]$	$[N_1, N_3]$	$[N_2, N_3]$
1, 2	1 3	2, 3
1 2	1 4	2, 4
3, 1	3 4	1 4
3 1	3 2	1 2

El algoritmo va a servir para calcular el clustering de cada nodo por lo que si se vuelve a replantear el ~~mate~~ análisis, será para asignar un valor a cada nodo.

	1	2	3	4	5		no rep
1	1	1	0	-1	0	4	4
2	0	-1	0	0	1	4	3
3	1	1	1	-1	1	5	3
4	0	0	1	0	0	2	0
5	0	1	-1	0	1	3	1
	4	4	5	2			



Combinaciones

$5C3 = 10$

11

$N_1 = 1$, $N_2 = 2$ $N_3 = 5$

$[N_1, N_2]$, $[N_1, N_3]$, $[N_2, N_3]$

1 2 ✓ 1 5 × 2 5 ✓

No todos son $[N_i, N_j] = 1$ or -1

Va a faltar agregar condicional para validar algún $[N_i, N_j] = -1$

P(k) es el numero de conexiones que forma con sus ~~vecinos~~ vecinos

C(k) cuantos triangulos forma.

fila (1,2), (1,3), (1,4), (1,5) Salida

1 0 -1 0

Col (2,1), (3,1), (4,1), (5,1) Entrada

0 1 0 0

Tomo k = 3

(2,1), (2,3), (2,4), (2,5)

fila 0 0 0 1

Col 1 1 0 1

Se ~~ref~~ repite (2,5), (5,2)

y aqui con algunos 1 cuenta como solo conexion

Entonces sera solo 1 ∴ k = 3

Consideramos para el caso de que ~~k~~ $N_i = N_j$

$\cancel{P(n_i)} = \dfrac{2 n_i}{k_i(k_i - 1)}$ n = unión

$C(k_i)$ n =

Por lo que será necesaria, por como se analizará si será necesaria otra condicional o algo fa

Para calcular P(k), tomamos ~~de~~ el Dataframe que está en la matriz inicial.

- El avance será por fila y por columna

Ya que se contará en la fila uno, con cuantos se conecta así contamos el numero de conexiones y contando consigo mismo

- Ahora calculamos el número de k con respecto al no de triangulos.

Graficamos para cada uno:

k_i número total de conexiones que tiene dicho nodo

n no de triangulos en dicho nodo

	k_{mat}	k_{in}	k	...
1				
2				
⋮				
n				

Ahora toca contar el total de k que hay

	1	2	3	4
1				
2				
3				
4				

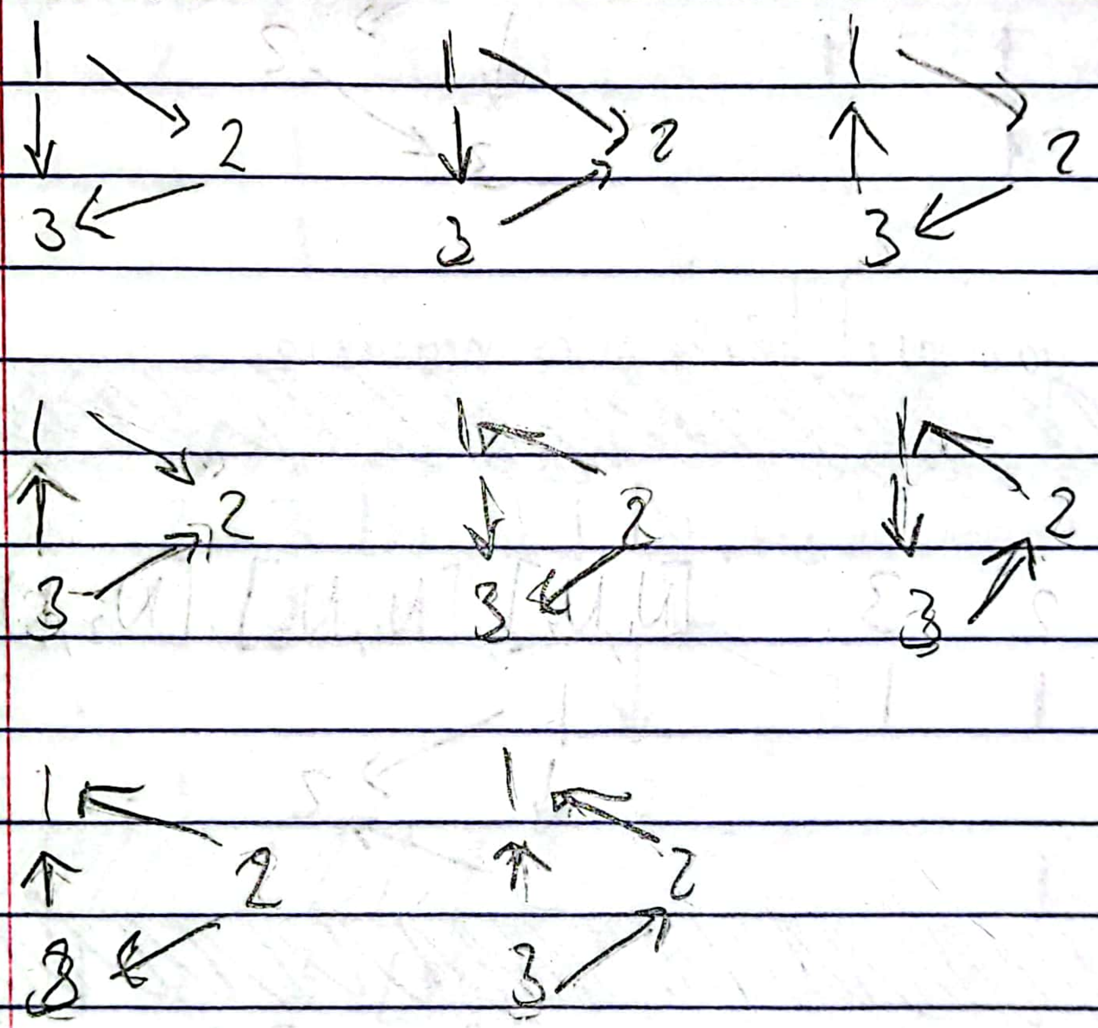
	1	2	3	4
1				
2				
3				
4				

	1	2	3	4
1				
2				
3				
4				

	1	2	3	4
1				
2				
3				
4				

Con un while podría iter que se cambie, sin tener que regresar a hacer una cuenta regresiva

Ahora regresando al calculo de triangulos (n) con todas las combinaciones <Condicional n>



Va de la fila a la columna, Direccion de flecha

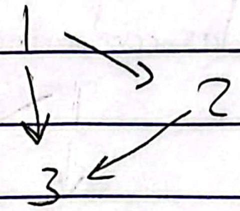
parece ser que es innecesario

row, col

1 2 × 3 $[N_1,N_2], [N_1,N_3], [N_2,N_2]$

Seguimos los graficos, a como se ordenaron en el script

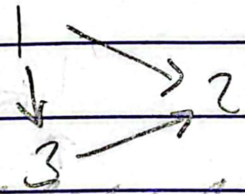
	1	2	3
1		I	I
2		I	
3			



A no, ya vi que el tercero si es necesario.

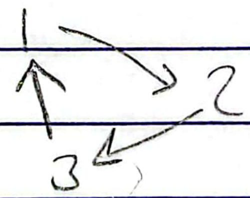
$[N_1,N_2], [N_1,N_3], [N_3,N_2]$

	1	2	3
1		I	I
2			
3		I	



$[N_1,N_2], [N_3,N_1], [N_2,N_3]$

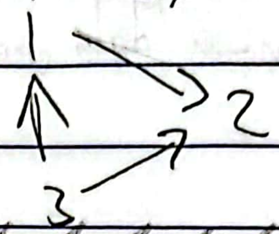
	1	2	3
1		I	
2			I
3	I		



Scribe

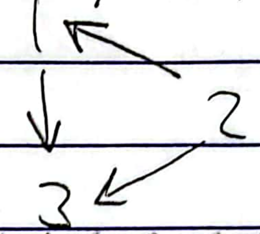
	1	2	3
1		1	
2			
3	1	1	

$[N_1, N_2], [N_3, N_1], [N_3, N_2]$



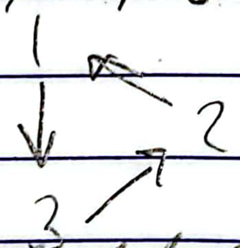
	1	2	3
1			1
2	1		1
3			

$[N_2, N_1], [N_1, N_3], [N_2, N_3]$



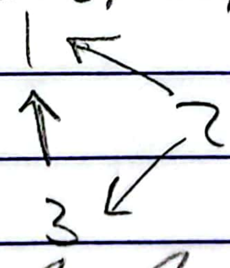
	1	2	3
1			1
2	1		
3		1	

$[N_2, N_1], [N_1, N_3], [N_3, N_2]$



	1	2	3
1			
2	1		1
3	1		

$[N_2, N_1], [N_2, N_1], [N_2, N_3]$



	1	2	3
1			
2	1		
3	1	1	

$[N_2, N_1], [N_3, N_1], [N_3, N_2]$