Análisis de redes booleanas con BoolNet

*Elisa Domínguez Hüttinger*

**Sesión 1: Obtener atractores, tamaños de las vasijas de atracción**

# entrar a R (Rstudio)

# así ponemos comentarios – esto no lo lo ve R

getwd() #¿dónde estoy?

setwd("C:\Users\Elisa\Dropbox\Docencia adscripción biomédicas\Bio Mates PCBIOL 2021 1\Prácticas computacionales\1 Intro a BoolNet") #¿dónde quiero estar?

help(c) # ¡cómo pedir ayuda!

install.packages("BoolNet") #ojo – necesitamos internet para cargar paquetes

library(BoolNet) # cargar (invocar) la librería

|  |
| --- |
| # en un documento de texto (ej. notepad; terminación .txt) en el mismo directorio, poner la red:  targets, factors  a, a  b, a & ! c  c, c & ! b  #y guardarla como le queramos llamar, en este caso red\_ejemplo.txt |

# de regreso en R:

net <- loadNetwork("red\_ejemplo.txt") # cargar la red

net #ver la red como la escribimos en notepad

plotNetworkWiring(net) # ver las conexiones regulatorias entre elementos de la red

attr <- getAttractors(net) #calcular los atractores

attr # ver los atractores

plotAttractors(attr) #graficarlos

getPathToAttractor(net, c(1,0,0)) # transiciones desde el estado 100 hasta el atractor

stateTransition(net, c(1,0,0)) # este comando es equivalente, pero no muestra el estado inicial (c(1,0,0))

path <- getPathToAttractor(net, c(1,0,0))

plotSequence(sequence=path)

tt <- getTransitionTable(attr) # tablas de transición

tt

plotStateGraph(attr) # vasijas de atracción –fragmentación del espacio fenotípico

**########################### mutantes#################################**

Mut <- getAttractors(net, genesON=c(0,0,0,0,1,0,0,0,0)) #apagamos el gen c

mut #¿cuántos atractores quedan?

knockedOut <- fixGenes(net, "c", 0) # este comando es equivalente al anterior – c siempre apagado

originalNet <- fixGenes(knockedOut, "c", -1) # resetea la red

overExpressed <- fixGenes(net, "c", 1) # c siempre prendido

**############### ruido estocástico #################################**

*# "****perturbTrajectories"*** *measures the influence of noise that is applied to the current network state. It generates a* ***set of initial states*** *and creates perturbed copies of these states by randomly flipping bits. It then measures the* ***influence of the flips on the further dynamic behaviour of the network.*** *For example:*

r <- perturbTrajectories(net, measure="hamming",numSamples=100,flipBits=1)

*randomly generates 100 states and 100 copies with one bit flip and performs a* ***single state transition for each state. It then measures the normalized Hamming distance (the fraction of different bits)*** *between each state and the corresponding perturbed copy. A robust network is assumed to yield a low Hamming distance. (Hamming distance between two strings of equal length is the number of positions at which the corresponding symbols are different. In another way, it measures the minimum number of substitutions required to change one string into the other, or the minimum number of errors that could have transformed one string into the other. Hamming distance is thus sort of the equivalent to the euclidean distance between two points)*

*#The long-term behaviour can be evaluated by comparing the attractors that are reache from the initial states and their perturbed copies.*

r <- perturbTrajectories(net, measure="attractor", numSamples=100,flipBits=1)

r$value

discusión: uno podría comparar diferentes tipos de redes y ver qué tan robustas son. Redes de todo tipo – aleatorias vs diferentes redes naturales.