SPIDDOR package vignette

Itziar Irurzun-Arana, Iñaki Tróconiz José David Gómez-Mantilla

October 28, 2016

Contents

1 Installation

Before starting, make sure you have installed an R version >3.2.0. Additionally, Rtools is needed to use the simulation algorithm in C++.

To install Rtools:

```
install.packages("installr")
library(installr)
install.Rtools()
```

If you have the most recent version of R, you should select the most recent Rtools download (at the top).

2 Introduction

SPIDDOR is an R package which consist on a set of tools to perform Boolean modeling in the context of development therapies for complex diseases. *SPIDDOR* allows users to simulate synchronous and asynchronous Boolean networks and analyze the results in terms of the average dynamic evolution of the nodes or in terms of attractors.

From a methodological point of view the Boolean analysis performed by *SPIDDOR* involves certain novelties. Common Boolean modeling approaches only define direct activation-inhibition relationships between the components of the network. In our models, we incorporate the modulation interactions, which are used to modulate the intensity of the activations of inhibitions produced by the regulator nodes. The package also allows users to specify the activity level of their nodes as a percentage ON in order to perform mutational studies or evaluate the inputs of the network with background noises. Additionally, *SPIDDOR* incorporates new visualization techniques to evaluate the attractors of the system and the effects of perturbations.

3 Assembling networks

Networks can be loaded from text files in which the user describe the Boolean functions (*BFs*) of the network extracted from literature data. Nodes can be genes, proteins, metabolites, cellular states, stimulus, etc. *BFs* consist on a set of rules specifying how the nodes' states change as a function of the current or past values of its regulator nodes.

3.1 Boolean operators

The main operations of Boolean algebra are the conjunction AND (& in the txt file), the disjunction OR (| in the txt file), and the negation NOT (! in the txt file). Some examples:

```
GeneA = TF1 & TF2
GeneB = TF1 | TF3
GeneC = ! TF3
```

Apart from these basic operators, some convenience operations have been defined:

- *U*: threshold operator used to check whether a regulator of a Boolean function is activated in the last n iterations that the user selects (n 3 by default). The *U* operator requires a duration argument which indicates the number of previous iteration that must be evaluated for a regulator node. This argument can be numerical or a character in which case its name will be saved inside a list where the user can select the value afterwards. Example:

  ```
  CTLA4 = U_TO_ACT[3] or CTLA4 = U_TO_ACT[TO_ACTmax]
  ```

- *MOD*: modulator operator that makes the same threshold function as the *U* operator but only affecting to the nodes that have a modulation interaction in the Boolean functions of the network. Example:

  ```
  IL12 =(CD40 & CD40L) | ((IL12 & ICOS) &! (MOD_IL12[4] & MOD_ICOS[4]))
  ```

- *ANY*: operator used to check whether a regulator of a Boolean function is activated in any of the last n iterations of the simulation that the user selects (n 3 by default). This operator is also used to modulate the Boolean functions of the network and has an argument to indicate the iterations that has to be evaluated (it can also be a number of a character). Example:

  ```
  NK = IL23 &! (NK & ANY_Treg[neg_modulator])
  ```

  Here, the Boolean functions of a toy network are shown as they should appear in the text file:

  ```
  APC-Ag = APC-Ag
  B71 = APC-Ag
  ICOS = APC-Ag
  CD40 = APC-Ag
  B7H2 = ICOS
  CD28 = ! CTLA4
  CTLA4= U_TO_ACT[3]
  CD40L = ICOS & B7H2 &! (CD40 & CD40L)
  TO_ACT = ((CD28 & B71) | (TO_ACT & B7H2) &! (MOD_TO_ACT & MOD_B7H2))
            &! (CTLA4 & B71)
  IL2 = TO_ACT
  IL6 = CD28
  IL12 = (CD40 & CD40L) | (IL12 & ICOS) &! (MOD_IL12 & MOD_ICOS)
  ```

  If this equations are saved to a ”Example_network.txt” file in the working directory, the network can be loaded to R via

```
BN<-read.Boolean.functions(file="Example_network.txt")
```

The same network is also included in SPIDDOR as an example and can be accessed via

```
data(Example_network)
```

3.2 Boolean inputs