**Boolean network modelling practical 14-10-2021**

In this practical, we will start by looking at a Boolean network model for the plant root stem cell niche (SCN). The paper by Azpeita et al. (2010) from which we have taken this model investigates how it is possible that this niche, though it contains stem cells that grow and whose progeny differentiates throughout the plant's life, nevertheless has a very stable spatial ordering (see Fig 2 of the paper).

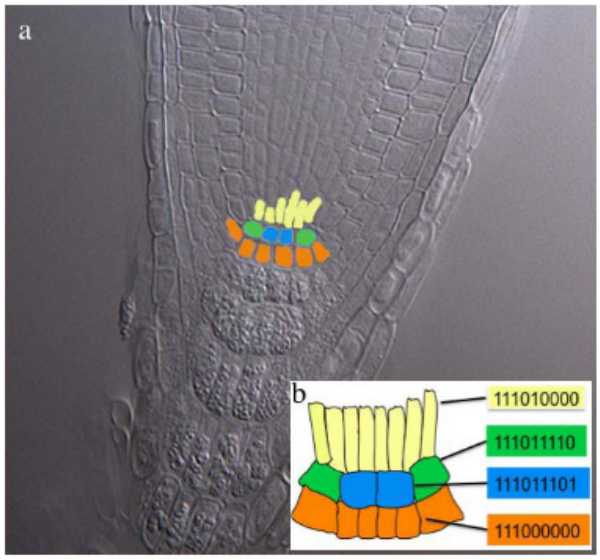
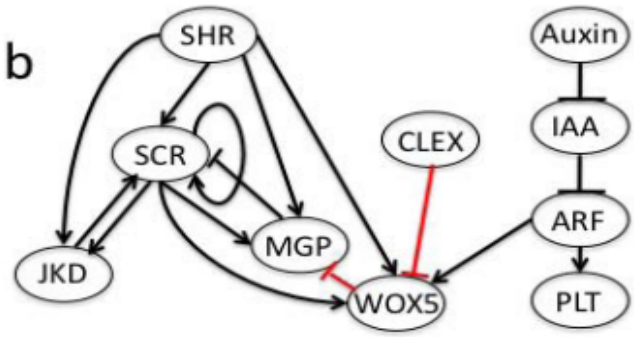



**Figure 2 Simplified cellular pattern of the root stem cell niche compared to a real root**. The four cell types within the SCN, each of which corresponds to one of four stable attractors recovered by the models in figure 1. The vascular initials (yellow), cortex-endodermis initials (green), quiescent center (blue) and columella-epidermis-lateral root cap initials (orange) (b) in the cleared root tip of *Arabidopsis thaliana* colorized to show the corresponding cell types represented by the attractors recovered by the single-cell GRN models in figure 1 and schematized in (a). The activation states of the nodes are represented by *0* and *1* in the following order: *PLT*, auxin, *ARF*, *Aux/IAA*, *SHR*, *SCR*, *JKD*, *MGP* and *WOX5*.

In the paper, the authors used experimentally known interactions between genes in the plant root to explain the existence of 4 distinct cell types (of which one is actually composed of 2 subtypes that the authors lumped together). In their investigations they tried out 3 different model variants. For this practical we will focus on model B (Fig 1B), which consists of 10 genes, including one that is unidentified but that the authors dubbed CLEX as it might be a CLE4-like factor. In table 3 of the paper the authors list the attractors of this model and their correspondence to the root tip cell types.




This practical aims to give you some practice with working with boolean network models. The chosen model is still of relatively limited complexity compared to e.g. the Toll receptor network illustrated in the lecture. The advantage is that it enables us to study it exhaustively, and keep track of which gene is which.

Before you start on the questions, have a look at the paper. Read the abstract, look at figure 1b, 2 and table 3 to familiarise yourself with some terms. Then, open *BooleanNetAssignmentFinalStudentVersion.R* and walk through it, completing the questions. Except for question 4, the first 6 questions only require you to step through the code already given and to look at the outputs. Good luck!

**Question 1**
Load the model "MODEL_Azpeitaetal.xml" and visualise it. Be aware that the plotNetworkWiring function sometimes plots self-interactions (feedback of a gene on itself) through other nodes, which can look confusing. How many states does this network have?

**Question 2**
What are the largest attractors of this network?

**Question 3**
Attractor 1 has a large basin of attraction and simply consists of AUX/IAA being on and all other genes being off. Looking at the network architecture, can you describe why this is such a common end state if we model the system from all possible starting points? Can you explain why this happens in exactly 256 states? Tip: if you want, use https://rumo.biologie.hu-berlin.de/boolesim/ in Google Chrome to easily visualise and play with the network. To do this, import the .txt file generated in the code.

**Question 4**
Test your answer from the question above by fixating the two genes to only obtain this single attractor. Use the fixGenes() function for this together with the original network. There's an example usage to show you how the function works.

**Question 5**
Can you now also explain why attractor 2 is so common?

**Question 6**
You can also plot the entire state space with all the attractors. Here, you plot every state of the system as a node, and visualise the transitions between them with arrows. Attractors are the final states that the whole network ends up in. Visualise these attractors now. Be sure to use the zoom function of your R plot window. What do you notice about the basin structure of attractor 1 and attractor 2?

**Question 7**
Find which of the other attractors we found correspond to which attractors discussed in the paper (table 3). What is the basin of attraction (size) of these 3 other paper attractors?

**Question 8**
You now know which attractors correspond to those in the paper, and that attractor 1 is not so interesting. There are other attractors. Let's zoom in on attractor 5, what does that correspond to, and what does it tell you about the promoter of WOX5 (i.e. how it is activated)? Check this in the rules.

**Question 9**
So far we have applied the updating rules at the same time for all nodes, called synchronous updating. However, transcription, translation and protein degradation do not necessarily occur at the

same rate for all genes, and additionally these processes are inherently noisy. Thus, synchronous updating of all genes may not be realistic. Let us instead consider asynchronous updating, were we update every node, one at a time, in a random order until we have updated every node (and repeat this until the network converges on some attractor). We will first show the difference on an example network of the mammalian cell cycle. Load it by typing *data("cellcycle")* Plot this network, and first look at the attractors when updating synchronously as we did before (i.e. use *plotStateGraph()*). What do you see?

**Question 10**
Which attractor might correspond to an actively dividing cell, and which to a quiescent cell?

**Question 11**
Now, let's update the network asynchronously (use type*='asynchronous'*, *method = 'random'*, and *startStates = 800*, for example) and plot the attractors you get. What happens? Why would this happen?

**Question 12**
We go back to our original root tip model. Get asynchronous attractors from the root SCN network. What do you observe? (Note: if you use *plotAttractors()* or when you *print()* the attractors, the order that the attractors appear in will be different).

**Question 13**
If you fix SHORTROOT at 0 (loss of function mutation), do you expect that SCARECROW (SCR) could still keep part of the network activated? Try it (Go back to synchronous updating!) What happens, and can you check the rules to see why this is?

**Question 14**
We might wonder how the results change if we change the rules of the network, rather than the states of the nodes (i.e. mutate regulatory interactions rather than genes). For this, we can use the function *perturbNetwork()*, which changes one of the regulatory interactions. Be sure to set *perturb = "functions"*, *method = "bitflip"*, *readableFunctions = TRUE* inside the function. Run it once on the initial network. What has changed? Do you see the change when plotting or in the Boolean functions? Do this two or three more times and see what changes.

To get a better idea of what is happening here:
Assume genes A and B control the expression of gene C such that C=(A AND B), so in other words, C=1 if A=1 and B=1 and 0 otherwise. Another way of writing this is as follows: First, gene C has 2 input genes, so there is a total of 2^2=4 possible input states (AB: 00, 01, 10, 11), let us order these according to the numeric value of their binary code, since 00=0, 01=1, 01=2, 11=3, this will be 00; 01; 10; 11. Now, let us write down how gene C responds to each of these input states: only A and B on gives C=1 so 00 → 0, 01 → 0, 10 → 0, 11 → 1. The update rule for gene C can now be written by only writing the ordered output for gene C: 0001. To see for our network what an update rule for e.g. SCR looks like you can use *YOURNETWORKNAME$interactions$SCR*, and look what is listed under func. What the perturbNetwork() function does is flip a bit in this update rule. If eg. 0001 changes into 0000 in the above example it means that gene C is switched on by none of the possible inputs it receives and hence is effectively knocked out. Note that to interpret the binary update rules you need to know which genes affect the gene, and in which order.

**Question 15:** Although Boolean network models are also suitable for simulating large networks this is not entirely trivial. Consider a network with 100 nodes. How many possible states does this network have? What consequences does this have?

***Optional final questions (Really optional. Feel free to stop and celebrate that you're done!)***

**Question 16**
(Feel free to take a peek in the answer script if you don't manage to do this yourself)
Use set.seed(666999) (just to ensure your findings match ours). Make a simple loop that perturbs the network randomly 70 times. Save the resulting networks to a list, and save the attractors in a different list. What are the minimum and maximum number of attractors you obtained?

**Question 17**
Take a look at the original network as well as your previous answers. Which regulatory change would cause a change to 2 attractors? And which one to 12? If you can not guess have a look at the rules of the perturbed networks generating these attractor numbers and compare them to the original network.