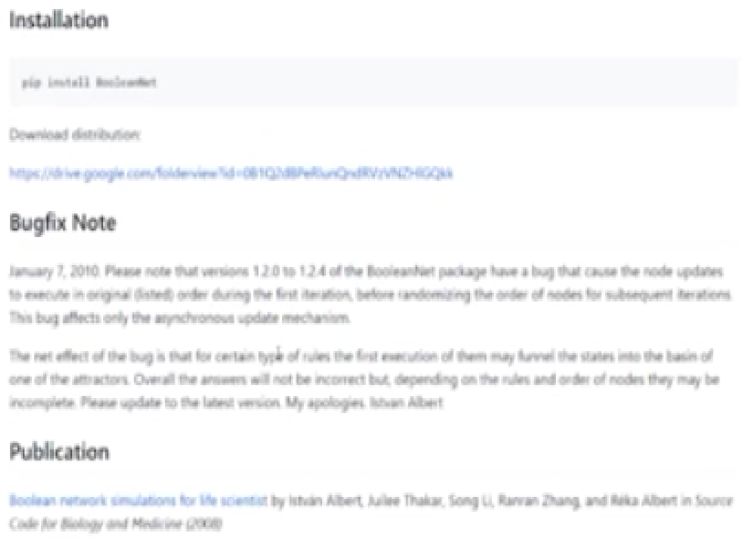
Computational Systems Biology
Karthik Raman
Department of Biotechnology
Indian Institute of Technology – Madras

Lecture - 90
Lab: Modelling Gene Regulatory Networks

In today's lab, we will continue with Boolean network modelling, and we will look at this interesting Python based tool called BolleanNet, which can be used to simulate Boolean networks and it has a nice format for specifying the Boolean transfer functions, and it is very easy to use.

(Refer Slide Time: 00:28)



Installation

```
pip install BooleanNet
```

Download distribution:

https://drive.google.com/folderview?id=0B1Q2d8PeRlunQndRVzVNZHlGQkk

Bugfix Note

January 7, 2010. Please note that versions 1.2.0 to 1.2.4 of the BooleanNet package have a bug that cause the node updates to execute in original (listed) order during the first iteration, before randomizing the order of nodes for subsequent iterations. This bug affects only the asynchronous update mechanism.

The net effect of the bug is that for certain type of rules the first execution of them may funnel the states into the basin of one of the attractors. Overall the answers will not be incorrect but, depending on the rules and order of nodes they may be incomplete. Please update to the latest version. My apologies. Istvan Albert

Publication

Boolean network simulations for life scientist by István Albert, Juilee Thakar, Song Li, Ranran Zhang, and Réka Albert in Source Code for Biology and Medicine (2008)

So let us look at BooleanNet. That is again a very nice tool. Examples, this is the absecic acid example, just look at the rule files, it is pretty straight forward. So there are few interesting things to know it here, especially in relation to what we discussed in the morning. So first of all you can say something is equals false or something equals true. These are initial conditions; these are practical aspects of the simulation that we touch upon in the morning.

So you have different nodes having a different initial state. This is very interesting. So I think not in this example, I can show you other examples where it is interesting. So this 1 colon tells you the rank of the update. We talked about 2 methods of update right, we talked about synchronous updates and asynchronous updates. So here you have an opportunity to preserve synchronous updates, but sort of enforce a ranking of which nodes are updated before which other nodes.

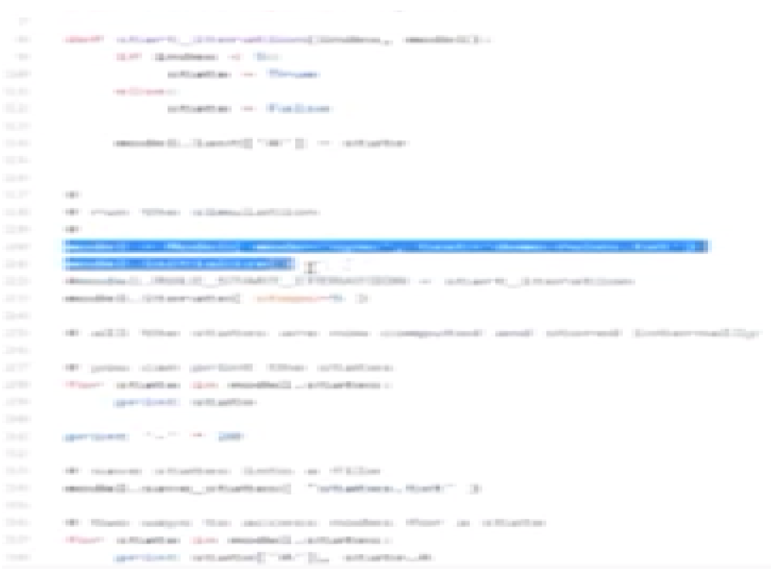
So if you have a gradation and what is the faster process and you do not even want to go in for a asynchronous update, you can go for a synchronous update and say I know that NOS is updated before calcium 2 is updated, because this maybe depends on that, of course this is more important in an asynchronous update. In a synronous update, it is any way going to use the previous states value, does not matter when it is updated.

But in asynchronous, it becomes important. So look at this, rank colon node star equals a Boolean expression. The first number is the rank, all are set to 1 here, means that the new value of a = the old value of a and b combined with the operator, then you have the model, you can have comments as well here. I will maybe just open a different model where you may be able to see this. Even here you do not have this.

So we had a paper where we talked about modelling the host-pathogen interaction in tuberculosis and we had ranks all the way up till 9 or 10, wherein we knew that nodes had to be updated before other nodes. If you have that knowledge ahead of time, you can employ that in the model, which is actually I think pretty cool feature of BooleanNet. **“Professor - student conversation starts”** Some nodes have initial states, some other nodes will not have initial states, which state will you take randomly?

You can specify that or by default I do not know if it assumes false, I will have to check that. Let us just look at the simulation you see here the code is really simple, 59 seems a lot of lines, but if you see a lot of it is just comments and so on. This has some concentration and so on, let us look at a simpler model where you will have even fewer. So literally you just have 4 lines of code that are important.

(Refer Slide Time: 04:14)


[illegible]

So model is mode sync and text rule file, initialise the model, iterate, actually 3 lines of course. And of course then you can print, for state in model or states print state. This is all the Python that you need to know, if you are not familiar with Python this is still very understandable, even if you are familiar with MATLAB or any of the other things we have been doing in the course.

So we just say model is model, mode is sync, synchronous you can also have asynchronous if you want, and the Boolean transfer function file is this demorules.txt, then you initialise the model and iterate, how many of that steps you want. “Professor - student conversation starts” I did not even have a network X in my bash. I should have just write it off my conda. No, no, this bash on windows 10, the best rater on bash on windows.

Better than bash, (()) (06:08) why, because it will auto compute your commands. Shreya what is your error? It is not installing. What is it throwing up? It is throwing something weird. I downloaded the (()) 06:30) and started and there it is showing some error with some installation. There is no module called UTIL. That sounds like some inbuilt BooleanNet module.

So if you see it is a very simple thing, just has modify states and state, test and so on, built in functions. No, there is no BooleanNet for Python 3 I think, maybe one of you can try and rewrite it. There is nothing, just print and do it, that is all. And division for individualization. See you can actually write Python 3 proof code by using from underscore feature import print function, so that can be Python 3 proof, and you write for Python 2 even.

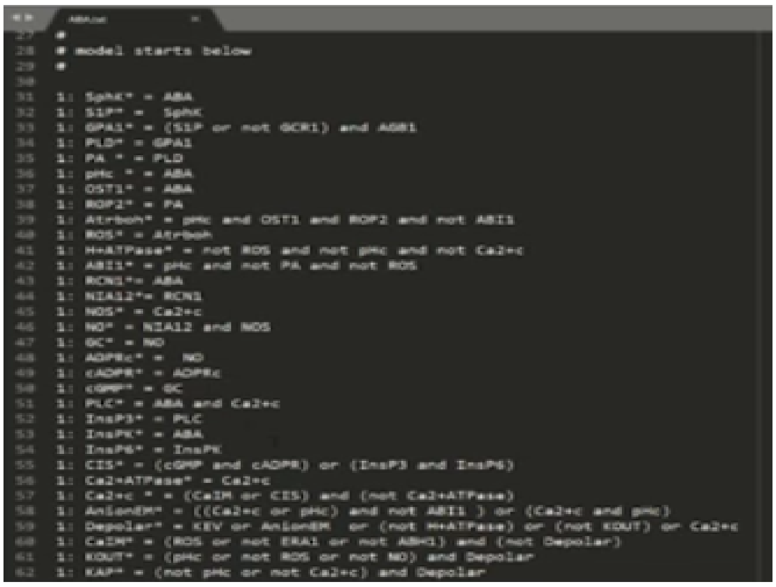
Any luck running the demos, Smitha. **"Professor - student conversation ends"** It says you need to initialise all the nodes, I do not know how they initialise here and trying to use the ABA model here, you can say missing = something, so random, and model.iterate steps = 10, that is it. It is as simple as doing that and now you can just, for state and model.states, print state.

(Refer Slide Time: 14:43)

```
[illegible]
```

So can you see this, so it just gives you the whole set of states, you can pull out whatever you want from this. So all I have to do is model.initialise, with some how you initialise the missing one, iterate, and for state and model.states, print states, it is as simple as that, all I needed was a file that showed you, rules in this fashion.

(Refer Slide Time: 15:28)

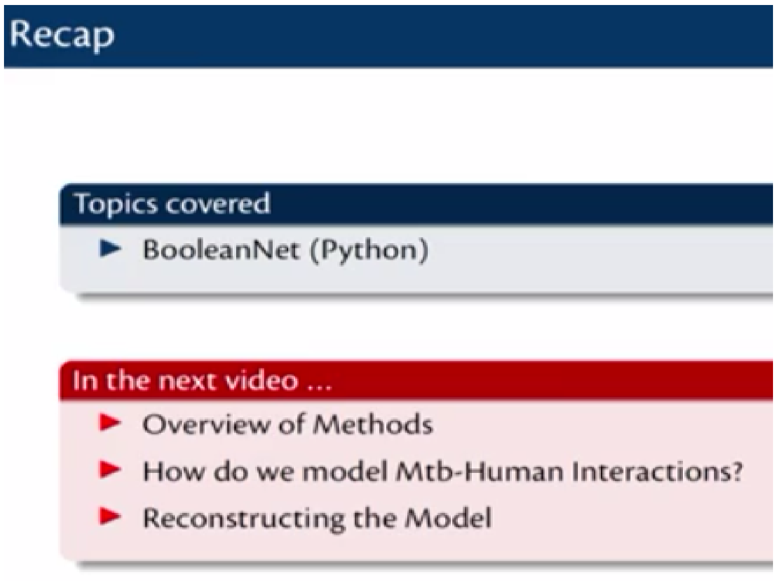


```
#
# model starts below
#

1: SphK* = ABA
1: S1P* =  SphK
1: GPA1* = (S1P or not GCR1) and AGB1
1: PLD* = GPA1
1: PA * = PLD
1: pHc * = ABA
1: OST1* = ABA
1: ROP2* = PA
1: Atrboh* = pHc and OST1 and ROP2 and not ABI1
1: ROS* = Atrboh
1: H+ATPase* = not ROS and not pHc and not Ca2+c
1: ABI1* = pHc and not PA and not ROS
1: RCN1*= ABA
1: NIA12*= RCN1
1: NOS* = Ca2+c
1: NO* = NIA12 and NOS
1: GC* = NO
1: ADPRc* =  NO
1: cADPR* = ADPRc
1: cGMP* = GC
1: PLC* = ABA and Ca2+c
1: InsP3* = PLC
1: InsPK* = ABA
1: InsP6* = InsPK
1: CIS* = (cGMP and cADPR) or (InsP3 and InsP6)
1: Ca2+ATPase* = Ca2+c
1: Ca2+c * = (CaIM or CIS) and (not Ca2+ATPase)
1: AnionEM* = ((Ca2+c or pHc) and not ABI1 ) or (Ca2+c and pHc)
1: Depolar* = KEV or AnionEM  or (not H+ATPase) or (not KOUT) or Ca2+c
1: CaIM* = (ROS or not ERA1 or not ABH1) and (not Depolar)
1: KOUT* = (pHc or not ROS or not NO) and Depolar
1: KAP* = (not pHc or not Ca2+c) and Depolar
```

Where you can even apply the ranks for asynch and so on. We will stop here and if you have any doubts, you can always raise them, we can discuss further. That is very easy, BoolSim of course you will agree as trivially easy to use and so on, and so is BooleanNet.

(Refer Slide Time: 15:50)



Recap

Topics covered

- BooleanNet (Python)

In the next video ...

- Overview of Methods
- How do we model Mtb-Human Interactions?
- Reconstructing the Model

In today's lab session, we looked at the Python based package called BooleanNet, which is very useful for simulating Boolean networks and it has a very simple format for specifying the Boolean transfer functions and so on and in the next video, we will look at methods to model, host-pathogen interactions and I am talking about it right after Boolean networks, because we actually have a very interesting piece of work, where in we used Boolean networks to simulate, how the human host and mycobacterium tuberculosis interact.