

Dynamic Populations and Networks

Nathaniel Osgood

MIT 15.879

March 16, 2012

Critical Role of Network & Population Dynamics

- We have introduced the basic mechanisms for
 - Creating populations of pre-specified sized
 - Creating network from a pre-specified set of network categories
- However,
 - Open populations (e.g. with immigration, death, birth) are the norm
 - Research suggests that many types of networks dynamics (serial partnerships, differing contact durations) are important to infection dynamics

AnyLogic's Support of Network & Population Dynamics

- Fortunately, AnyLogic provides strong support for
 - Adding & removing population members
 - Adding & removing connections
- However, this support does not yet have direct graphical interface support or specification
 - using this support does require that you call “methods” to accomplish this

AnyLogic Support for Changing Populations

- Adding to population
 - `add_populationname(parameters)`
 - *Allow explicit specification of agent parameter values*
 - `add_populationname()`
 - *Uses population specification of agent parameter values*
- Deleting from population
 - `remove_populationname(agentToBeRemoved)`

AnyLogic methods for Adding & Deleting Connections

- *agentA.connectTo(agentB)*
 - Connects *agentA* to *agentB*
 - NB: Connections are assumed to be undirected and symmetric (i.e. if *agentA* is considered to be connected to *agentB*, then *agentB* is considered to be connected to *agentA*)
- *agentA.disconnectFrom(agentB)*
 - Disconnects *agentA* and *agentB* from each other
- For more details and additional methods, see the slides for the *Networks* lecture



Hands on Model Use Ahead



Load Previously Built Model:
MinimalistNetworkABMModel

Suggest Saving as “MinimalistNetworkABMModelWithInterfaceDrivenPopulationDynamics”

Set Small Population Size (5)

The screenshot displays the AnyLogic Advanced software interface. The main workspace shows a simulation model with a grid background. A mouse cursor is positioned over a button labeled "Add a Person". Below the button, there is a "population [...]" object, an "environment" object, and a "totalPopulation" object. The left sidebar shows a project tree with the following structure:

- MinimalistNetworkABMModelWithInterf
- Main
- Parameters
- Environments
 - environment
- Embedded Objects
 - population
- Presentation
 - population_presentation
 - buttonAddPerson
- Person
 - Presentation
 - oval
 - line
- Simulation: Main

The bottom panel shows the "Properties" window for the "population - Person" object. The "General" tab is active, displaying the following settings:

- Name: population
- Show Name
- Ignore
- Public
- Show At Runtime
- Create Presentation
- Type: Person
- Package: minimalistnetworkabmmodel
- Environment: environment
- Replication: 5

Set Distance Based Network with High Connection Range Threshold

The screenshot displays the AnyLogic Advanced software interface for configuring a network model. The main workspace shows a grid with a mouse cursor pointing to a location, and a button labeled "Add a Person". Below the workspace, the "environment - Environment" properties panel is open, showing the following settings:

- Space type: Continuous Discrete GIS
- Width: 500
- Height: 500
- Columns: 100
- Rows: 100
- Neighborhood type: Moore
- Layout type: User-defined Apply on startup
- Network type: Distance based Apply on startup
- Connections per agent: 2
- Connection range: 1000
- Neighbor link fraction: 0.95
- M: 10

The left sidebar shows a project tree for "MinimalistNetworkABMModelWithInterf" with the following structure:

- Main
 - Parameters
 - Environments
 - environment
 - Embedded Objects
 - population
 - Presentation
 - population_presentation
 - buttonAddPerson
- Person
 - Parameters
 - Presentation
 - oval
 - line

The bottom-left pane shows a table with columns "Description" and "Location".

To Main: Add Button to Request Adding Population Member

The screenshot displays the AnyLogic Advanced software interface. The main workspace shows a diagram with a button labeled "Add a Person" being added to the "Main" model. The button is connected to a "population" object. The "Properties" panel for the button is visible, showing the following configuration:

- General**
 - Name: Show Name Ignore Public Icon
- Advanced**
 - Label:
- Dynamic**
 - Enabled: true
- Description**
 - Action:

```
add_population(); // add with the population-given parameters
environment.applyNetwork(); // recompute the new network with this person added
```

The "Palette" on the right side of the interface lists various components, including Model, Action, Analysis, Presentation, and various shapes like Line, Polyline, Curve, Rectangle, Round Rectangle, Oval, Arc, Pixel, Text, Image, Group, Button, Check Box, Edit Box, Radio Buttons, Slider, Combo Box, List Box, File Chooser, Progress Bar, CAD Drawing, and GIS Map.

To Person's "Oval", Add a "Handler" to Delete a Person if their Node is Clicked

The screenshot displays the AnyLogic Advanced software interface. The main workspace shows a diagram with a grid background. A blue oval node is selected, and a mouse cursor is hovering over it. A line connects this oval to a circular node labeled "income".

The left sidebar shows a project tree with the following structure:

- MinimalistNetworkABMModelWithInterf
 - Main
 - Parameters
 - Environments
 - environment
 - Embedded Objects
 - population
 - Presentation
 - population_presentation
 - buttonAddPerson
 - Person
 - Parameters
 - Presentation
 - oval
 - line

The bottom right pane shows the properties for the selected "oval - Oval" object:

- General: Radius X: [input field]
- Advanced: Radius Y: [input field]
- Dynamic: Replication: [input field]
- Description: Visible: [input field]
- X: [input field]
- Y: [input field]
- Fill Color: [input field]
- On Click:

```
this.get_Main().remove_population(this);  
environment.applyNetwork(); // recompute the new network with this person missing
```
- Rotation: [input field]



Hands on Model Use Ahead



Load Provided Shared Model:
ABMModelWithBirthDeath

Adding an Immigrant to the Model Population

The screenshot displays the AnyLogic Advanced software interface. The main workspace shows a model diagram with several entities and variables: Population [..], datasetInfective, environment, offspringDistanceFromMother, initialPrevalenceOfInfection, immigrantsPerYear, ImmigrantArrival (highlighted with a lightning bolt icon), prevalenceOfInfectionAmongImmigrants, and MeanLifespan. The left sidebar contains a tree view of the model structure, including ABMModel, Main, and Person.

The bottom panel shows the configuration for the **ImmigrantArrival - Event**. The **General** tab is active, displaying the following settings:

- Name:
- Show Name Ignore Public Show At Runtime
- Trigger Type:
- Rate:
- Action:

```
add_Population(uniform(MeanLifespan), Person.RandomEthnicity(), Person.RandomSex(), uniform() < prevalenceOfInfectionAmongImmigrants)
```

Add Population Options – Note Customization to Context

The screenshot displays the AnyLogic Advanced software interface. The main workspace shows a model diagram with various components: Population [..], datasetInfective, environment, offspringDistanceFromMother, initialPrevalenceOfInfection, immigrantsPerYear, ImmigrantArrival, prevalenceOfInfectionAmongImmigrants, and MeanLifespan. A tooltip is visible over the 'Action' field of the 'ImmigrantArrival' event, providing details about the 'add_Population' method.

add_Population(double InitialAge, Ethnicity ethnicity, Sex sex)
add_Population() Person - Main

This method creates and adds new embedded object in the replicated embedded object collection Population
This method uses given parameter values to setup created embedded object
Index of this new embedded object instance can be obtained through calling `Population.size()` method **before** this method is called

Parameters:
InitialAge
ethnicity
sex
isInitiallyInfected
mother

Returns:
newly created embedded object

`add_Population(uniform(MeanLifespan), Person.RandomEthnicity(), Person.RandomSex(), uniform() < prevalenceOfInfectionAmongImmigrants)`

Establishing Baby's Connection

Looping over Connections

The screenshot displays the AnyLogic Advanced software interface. The main workspace shows a statechart with a 'PregnancyStatus' state that transitions between 'NonPregnant' and 'Pregnant' states. A list of functions is visible, including 'EstablishOffspringConnectionsBasedOnMothersConnections'. The bottom panel shows the code for this function, which loops through the mother's connections to establish links between the baby and each connection.

Project: Deer

Person Statechart:

- ethnicity
- CurrentAge
- mother
- FinalizeDeath
- FertilityRateAgeSexEthnicity
- PerformBirth
- EstablishOffspringConnectionsBasedOnMothersConnections
- EstablishOffspringLocationBasedOnMothersLocation

Function: EstablishOffspringConnectionsBasedOnMothersConnections - Function

Function body:

```
// now establish links between the baby and all of the mother's connections
if (mother.getConnections() != null) // guard against a mother with no connections
    for (Agent a : mother.getConnections())
    {
        Person p = (Person) a;
        offspring.connectTo(p);
    }

// Finally, establish a link between the baby and the mother
// (we do this last so we don't have to worry that one of
// the mother's connections is to this offspring!

offspring.connectTo(mother);
// note that the "mother" property of the baby has already been set when it was created
```

Code to Perform Birth

The screenshot displays a software development environment with a class diagram and a code editor. The class diagram shows a **PregnancyStatus** class with two subclasses: **NonPregnant** and **Pregnant**. The **Pregnant** class has a **mother** property. The **PerformBirth** function is highlighted in the diagram. The code editor shows the implementation of the **PerformBirth** function, which creates an offspring, establishes connections, and positions the baby.

```
Person mother = this;
Person offspring = get_Main().add_Population((double) 0, ethnicity, RandomSex(), this.IsInfected(), mother);
traceln("A baby has been born! Baby's id is " + offspring + " while the mother is " + this);
// establish connections of infant
EstablishOffspringConnectionsBasedOnMothersConnections(offspring, mother);
// now position the baby to be close to the mother (otherwise leads to stretching of mother's connections)
EstablishOffspringLocationBasedOnMothersLocation(offspring, mother);
```


Establishing Baby's Connection

Looping over Connections

The screenshot displays the AnyLogic Advanced software interface. The main workspace shows a statechart with a 'PregnancyStatus' state that transitions between 'NonPregnant' and 'Pregnant' states. A list of functions is visible, including 'EstablishOffspringConnectionsBasedOnMothersConnections'. The bottom panel shows the code for this function, which loops through the mother's connections to establish links between the baby and all of the mother's connections.

Project: Deer

Statechart: PregnancyStatus (Start) → NonPregnant ↔ Pregnant

Functions: ethnicity, CurrentAge, mother, FinalizeDeath, FertilityRateAgeSexEthnicity, PerformBirth, **EstablishOffspringConnectionsBasedOnMothersConnections**, EstablishOffspringLocationBasedOnMothersLocation

Function: EstablishOffspringConnectionsBasedOnMothersConnections - Function

Function body:

```
// now establish links between the baby and all of the mother's connections
if (mother.getConnections() != null) // guard against a mother with no connections
    for (Agent a : mother.getConnections())
    {
        Person p = (Person) a;
        offspring.connectTo(p);
    }

// Finally, establish a link between the baby and the mother
// (we do this last so we don't have to worry that one of
// the mother's connections is to this offspring!

offspring.connectTo(mother);
// note that the "mother" property of the baby has already been set when it was created
```

Setting Offspring Location

The screenshot displays the AnyLogic Advanced software interface. The main workspace shows a statechart diagram with states: **Susceptible**, **Infective**, **NonPregnant**, and **Death**. Transitions are labeled with events like **color**, **circlesize**, **CircleSize**, **colorForRelation**, **isInitiallyInfected**, **PregnancyStatus**, **sex**, **ethnicity**, **mother**, **appearanceTime**, **InitialAge**, **CurrentAge**, and **FinalizeDeath**. A function **EstablishOffspringLocationBasedOnMothersLocation** is highlighted in the bottom panel.

EstablishOffspringLocationBasedOnMothersLocation - Function

Function body:

```
double dOffspringDirectionFromMotherInRadians = uniform(2 * 3.14159);
double offspringDistanceFromMother = get_Main().offspringDistanceFromMother;

offspring.setXY(mother.getX() + offspringDistanceFromMother * Math.cos(dOffspringDirectionFromMotherInRad
               mother.getY() + offspringDistanceFromMother * Math.sin(dOffspringDirectionFromMotherInRad
```