

# Dynamic Populations and Networks

Nathaniel Osgood

CMPT 394

February 14, 2013

# 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 Provided (or 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 diagram with a grid background. A mouse cursor is positioned over a small circle on the grid. A button labeled "Add a Person" is visible. Below the workspace, the "Properties" panel is open, showing the configuration for the selected "population - Person" object.

The "Properties" panel for "population - Person" includes the following fields and options:

- General:** Name: ;  Show Name;  Ignore;  Public;  Show At Runtime;
- Parameters:** Type:
- Statistics:** Package:
- Description:** Environment:
- Replication:

The "Project" pane on the left shows the following structure:

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

# Set Distance Based Network with High Connection Range Threshold

The screenshot displays the AnyLogic Advanced software interface, specifically the configuration for an environment. 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 "Properties" panel is open, showing the configuration for the "environment - Environment".

**environment - Environment**

General

Space type:  Continuous  Discrete  GIS

Advanced

Description

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



# 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:

- Name:** buttonAddPerson
- Show Name:**
- Ignore:**
- Public:**
- Icon:**
- Label:** Add a Person
- Enabled:** true
- 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 grid with a blue oval object labeled "oval" and a circular object labeled "income". A mouse cursor is positioned over the "oval" 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
      - Parameters
      - Presentation
        - oval
        - line

The bottom right pane shows the properties for the selected "oval - Oval" object. The "On Click" property is set to the following code:

```
this.get_Main().remove_population(this);  
environment.applyNetwork(); // recompute the new network with this person missing
```

# To Delete a Connection Between two Agents when Clicking on a Visual Link

The screenshot displays the AnyLogic University software interface. The main workspace shows a grid with a blue line connecting a central blue circle to a smaller blue circle below it. The interface includes a menu bar (File, Edit, View, Draw, Model, Tools, Help), a toolbar, and several panels:

- Projects:** A tree view on the left showing a project structure with folders like 'Person' and 'Simulation'.
- Properties:** A panel at the bottom center showing the properties of the selected 'line - Line' object.
- Console:** A panel at the bottom center showing the console output.
- Palettes:** A panel on the right showing various objects and components available for use.

The 'line - Line' properties panel is expanded to show the 'Dynamic' section, which contains the 'On click' event handler:

```
this.getConnections().remove(index);
```

The 'Description' section contains the following properties:

- Replication: `this.getConnectionsNumber()`
- Visible:
- X:
- Y:
- Z:
- Z-Height:
- On click: `this.getConnections().remove(index);`
- Rotation, rad:
- Scale X:
- Scale Y:
- Scale Z:
- dx: `this.getConnectedAgent(index).getX() - this.getX()`
- dy: `this.getConnectedAgent(index).getY() - this.getY()`

The 'Problems' panel at the bottom left shows 25 warnings, including several instances of 'The static field Main...' and 'The static field Deer...'.



Hands on Model Use Ahead



Load Provided 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 components: Population [..], datasetInfective, environment, offspringDistanceFromMother, initialPrevalenceOfInfection, immigrantsPerYear, ImmigrantArrival (highlighted with a lightning bolt icon), prevalenceOfInfectionAmongImmigrants, and MeanLifespan. The bottom panel is open to the 'Properties' view for the 'ImmigrantArrival - Event'.

**ImmigrantArrival - Event**

**General**

Name:   Show Name  Ignore  Public  Show At Runtime

**Description**

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)`

# Removing a Population Member

The screenshot displays the AnyLogic Advanced software interface. The main workspace shows a statechart model with three states: Susceptible, Infective, and NonPregnant. Transitions are labeled with events like 'PregnancyStatus' and 'FinalizeDeath'. The Susceptible state transitions to Infective, and both Susceptible and Infective states transition to Death. The NonPregnant state is reached from the Infective state via the 'PregnancyStatus' event.

The Properties panel for the 'FinalizeDeath' function is visible, showing the following code:

```
Function body:  
println("Population member " + this + " has died.");  
get_Main().remove_Population(this);
```

The Project Explorer on the left shows a hierarchy of collections and statecharts, including 'InfectionStatechart' and 'Statecharts'. The Properties panel at the bottom shows a table with columns for 'Description' and 'Location'.

# Establishing Baby's Connection

## Looping over Connections

The screenshot displays the AnyLogic Advanced software interface. The top-left pane shows a project tree with a 'Person' object selected. The main workspace shows a statechart with a 'PregnancyStatus' state and two states: 'NonPregnant' and 'Pregnant'. The 'NonPregnant' state is currently active. The right side of the workspace lists several functions, with 'EstablishOffspringConnectionsBasedOnMothersConnections' selected. The bottom-right pane shows the code editor for this function, which contains the following code:

```
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` method is highlighted in the diagram. The code editor shows the implementation of the `PerformBirth` method, 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 two states: **NonPregnant** and **Pregnant**. A transition labeled **PregnancyStatus** leads to the **NonPregnant** state. The **Pregnant** state has a self-loop and a transition to the **NonPregnant** state. The **NonPregnant** state has a transition to the **Pregnant** state. The **Person** object has several properties: **ethnicity**, **CurrentAge**, **mother**, **FinalizeDeath**, **FertilityRateAgeSexEthnicity**, **PerformBirth**, **EstablishOffspringConnectionsBasedOnMothersConnections**, and **EstablishOffspringLocationBasedOnMothersLocation**.

The **EstablishOffspringConnectionsBasedOnMothersConnections - Function** is defined in the **Code** tab. The function body is as follows:

```
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 'PregnancyStatus' and 'FinalizeDeath'. A function 'EstablishOffspringLocationBasedOnMothersLocation' is highlighted in the bottom panel, showing its code body.

**Statechart Diagram:**

- States: Susceptible, Infective, NonPregnant, Death.
- Transitions: Susceptible to Infective (event: colorForRelation), Infective to Susceptible (event: colorForRelation), Infective to Death (event: FinalizeDeath), NonPregnant to Susceptible (event: PregnancyStatus).

**Function Code:**

```
EstablishOffspringLocationBasedOnMothersLocation - Function  
General  
Code  
Description  
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
```