

# Dynamic Populations and Networks (& Events!)

Using Modeling to Prepare for Changing  
Healthcare Needs

Duke-NUS

April 16, 2014

# Critical Role of Network & Population Dynamics

- We have introduced the basic mechanisms for
  - Creating populations of pre-specified size
  - 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 contagion 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 “MinimalistNetworkABMModelWithOpenPopulation”

# Add Immigration (using Event)

- Assume 5 people coming in per year
- We create an event to add people to the population

# Add Death (using Statechart)

- Define



# Add Birth (Using Statechart)

# Set Small Population Size (5)

The screenshot displays the AnyLogic Professional interface. The main workspace shows a simulation model with a central circle and a blue line extending to the right, labeled "population [...]". The left sidebar shows the project structure with "Main", "Person", and "Simulation: Main". The right sidebar shows the "Palette" with various agent types. The bottom panel shows the "Properties" window for the "population - Person" agent, where the "Initial number of agents" is set to 5.

AnyLogic Professional

File Edit View Draw Model Tools Help

Projects Search

MinimalistNetworkABMMoc

- Main
- Person
- Simulation: Main

Simulation Main Person

population [...]

Properties Progress

**population - Person**

Name:  ☒ Show name ☐ Ignore

Visible: ☒ yes

☐ Single agent ☒ Population of agents

Initial number of agents:

**Initial location**

These settings are applied only if the "User-defined" layout type is set in the "Environment for other agents" properties of the upper level agent.

X:

Time units: minutes

# Set Distance Based Network with High Connection Range Threshold

The screenshot displays the AnyLogic Professional software interface. The main workspace shows a grid with a blue line representing a network and a red star icon labeled "population [..]". The left sidebar shows the project structure with "Main", "Person", and "Simulation: Main". The right sidebar shows the "Palette" with various agent types like Agent, Parameter, Event, etc. The bottom panel shows the "Main - Agent Type" properties.

**Main - Agent Type Properties:**

- Space type: ☒ Continuous ☐ Discrete ☐ GIS
- Space dimensions:
  - Width: 500
  - Height: 500
  - Z-Height: 0
- Layout type: User-defined ☒ Apply on startup
- Network type: Distance-based ☒ Apply on startup
- Connection range: 1000

Time units: minutes

# To Main: Add Button to Request Adding Population Member

The screenshot displays the AnyLogic Professional software interface. The main workspace shows a simulation model with a central circle and a line extending to the right, ending in an arrow pointing to a button labeled "Add a Person". Below the button is a red star icon labeled "population [...]".

The left sidebar shows the project structure for "MinimalistNetworkABMMoc", including "Main", "Agents", "Presentation", "Parameters", "Links to agents", and "Person".

The right sidebar shows the "Palette" with various controls, including "Button", "Check Box", "Edit Box", "Radio Buttons", "Slider", "Professional", and "Combo Box".

The bottom panel shows the properties for the selected "buttonAddPerson - Button".

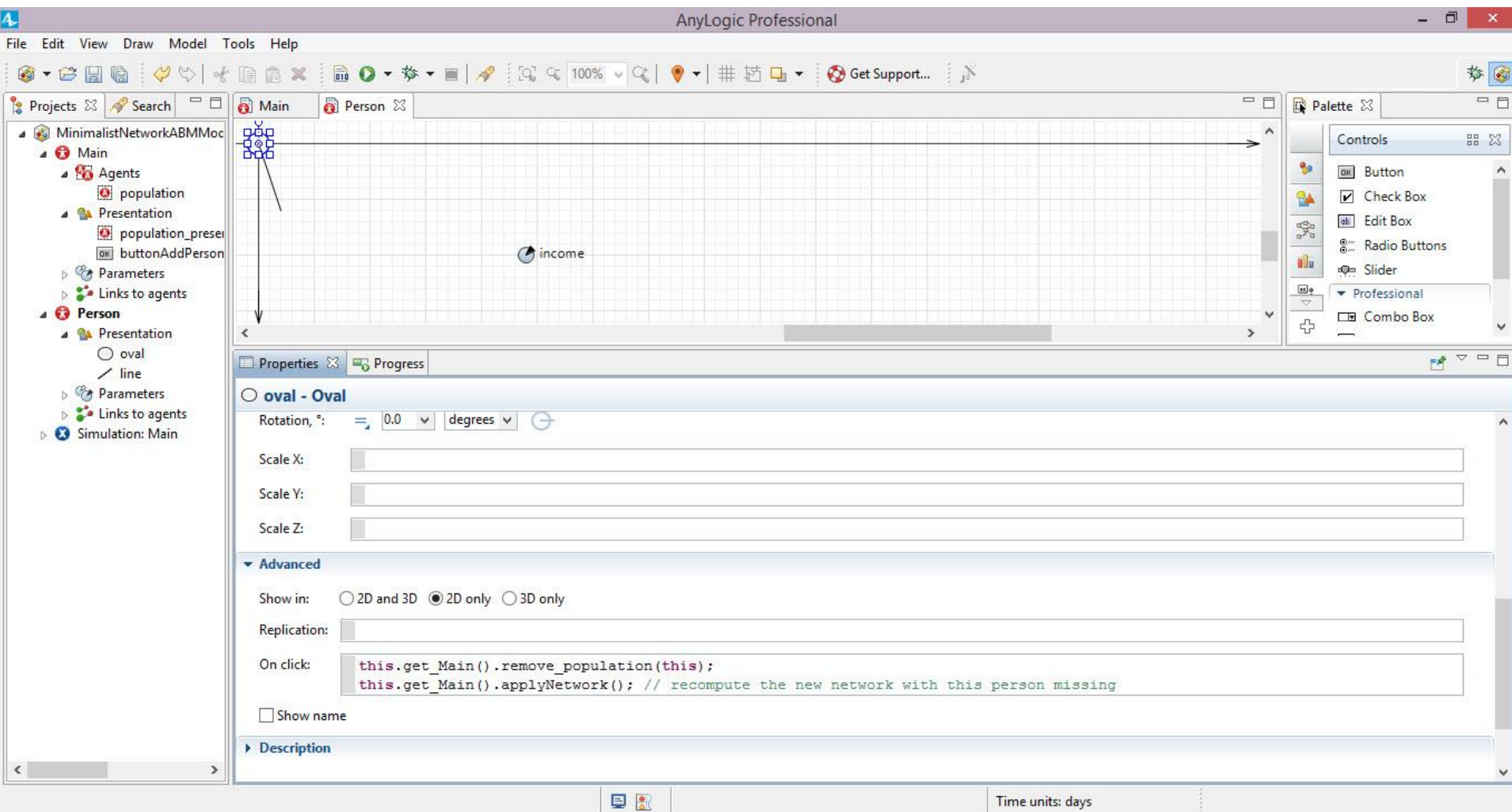
**buttonAddPerson - Button Properties:**

- Name:  ☐ Ignore ☒ Visible on upper level
- Label:
- Enabled: ☒ true
- Action**  

```
add_population(); // add with the population-given parameters  
applyNetwork(); // recompute the new network with this person added
```
- Appearance**
  - Background color:
  - Text color:
  - Font:   pt

Time units: days

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



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

The screenshot displays the AnyLogic Professional software interface. The main workspace shows a grid with two agents connected by a visual link. The 'line - Line' properties panel is open, showing the 'On click' event handler set to `this.getConnections().remove(index);`. The 'Advanced' section shows the 'Show in' options set to '2D only' and the 'Replication' set to `this.getConnectionsNumber()`.

**AnyLogic Professional**

File Edit View Draw Model Tools Help

Projects Search

MinimalistNetworkABMMoc

- Main
  - Agents
    - population
  - Presentation
    - population\_presen
    - buttonAddPerson
  - Parameters
  - Links to agents
- Person
  - Presentation
    - oval
    - line
  - Parameters
  - Links to agents
- Simulation: Main

line - Line

Position and size

X: 0 dX: `this.getConnectionAgent(index).getX() - this.getX()`

Y: 0 dY: `this.getConnectionAgent(index).getY() - this.getY()`

Z: 0 dZ: 0

Z-Height: 10

Rotation, rad:

Scale X:

Scale Y:

Scale Z:

Advanced

Show in: ☐ 2D and 3D ☒ 2D only ☐ 3D only

Replication: `this.getConnectionsNumber()`

On click: `this.getConnections().remove(index);`

☐ Show name

MinimalistNetworkABMMModelWi...aceDrivenPopulationDynamics

Time units: days

X=...69





# Hands on Model Use Ahead



Load Provided Model:  
**ABMModelWithBirthDeath**

# Adding an Immigrant to the Model Population

The screenshot displays the AnyLogic Professional software interface, which is used for building agent-based models. The main workspace is divided into several sections:

- Parameters:** Lists model parameters such as `infectionAmongImmigrants`, `TabDelimitedFile`, and `OnPeopleMatchingPredicate`.
- Statistics Dimension:** Lists statistical methods like `selectPeopleMatchingPredicate`, `countPeopleMatchingMultiplePredicates`, `countPeopleMatchingPredicate`, `reportSummaryStatistics`, `initializeSummaryStatistics`, and `reportSummaryStatisticsNaive`.
- Main Events:** Contains event definitions, including `eventReportSummaryStatistics` and `immigrantArrival` (highlighted with a lightning bolt icon).
- Model Entities:** Shows the `Population [...]` entity.

On the right side, there is a **Palette** with various controls like Button, Check Box, Edit Box, Radio Buttons, Slider, Combo Box, List Box, File Chooser, and Progress Bar.

The bottom section shows the **Properties** and **Progress** tabs. The **immigrantArrival - Event** properties are visible:

- Name:** `immigrantArrival` (with ☒ Show name and ☐ Ignore).
- Visible:** ☒ yes.
- Trigger type:** Rate (selected from a dropdown).
- Rate:** `immigrantsPerYear`.
- Action:** `add_Population(uniform(meanLifespan), Person.randomEthnicity(), Person.randomSex(), uniform() < prevalenceOfInfectionAmongImmigrants,`
- Description:** (Currently empty).

The status bar at the bottom indicates the model name `ABMMModelWithBirthDeathUseAnylogic7`, time units `days`, and a zoom level of `X=...06`.



# Add Population Options – Note Customization to Context

The screenshot displays the AnyLogic Professional software interface. The main workspace shows a diagram with a circle and a line, representing a population entity. The interface is divided into several panels:

- Left Panel (Project Explorer):** Shows the project structure for 'ABMModelWithBirthDeathU'. The 'Main' folder is expanded, showing sub-folders like 'Agents', 'Presentation', 'Parameters', 'Analysis Data', 'Functions', 'Links to agents', 'Collections', 'Events', 'Connectivity', and 'Person'. The 'Events' folder is expanded, showing 'eventReportSummary' and 'immigrantArrival'.
- Top Panel (Main Canvas):** Displays the main diagram area. It is divided into three sections: 'Parameters', 'Statistics Dimension', and 'Main Events'. The 'Main Events' section contains 'eventReportSummaryStatistics' and 'immigrantArrival'. The 'Model Entities' section contains 'Population [...]'. A blue line connects the 'immigrantArrival' event to the 'Population' entity.
- Right Panel (Palette):** Shows a list of controls available for the diagram, including 'Button', 'Check Box', 'Edit Box', 'Radio Buttons', 'Slider', 'Combo Box', 'List Box', 'File Chooser', and 'Progress Bar'.
- Bottom Panel (Properties and Code):** The 'Properties' tab is active for the 'immigrantArrival' event. It shows the event's name, visibility, trigger type (Rate), and rate. The 'Action' tab is also visible, showing the code for the event: `add_Population(uniform(meanLifespan), Person.randomEthnicity(), Person.randomSex(), uniform() < prevalenceOfInfectionAmongImmigrants,`. A tooltip for the `add_Population()` method is displayed, explaining its purpose and return value.

# Removing a Population Member

The screenshot displays the AnyLogic Professional software interface. The main workspace is divided into four panels: Person Parameters, System Parameters, Disease Parameters, and Pregnancy Parameters. The 'finalizeDeath' function is selected in the Pregnancy Parameters panel. The Properties panel at the bottom shows the function body, which contains the following code:

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

The left sidebar shows the project structure, including the 'Person' entity and its associated predicates and actions. The right sidebar shows the Controls palette, which includes various UI elements like Buttons, Check Boxes, Edit Boxes, Radio Buttons, Sliders, Combo Boxes, List Boxes, File Choosers, and Progress Bars.

# Establishing Baby's Connection Looping over Connections

The screenshot displays the AnyLogic Professional interface. The main workspace shows the 'Person' agent's parameters organized into four columns: Person Parameters, System Parameters, Disease Parameters, and Pregnancy Parameters. The 'establishOffspringConnectionsBasedOnMothersConnections' function is selected in the 'Pregnancy Parameters' column.

Person Parameters	System Parameters	Disease Parameters	Pregnancy Parameters
sex	color	dictLastTimeInfectedForPerson	performBirth
mother	circlesize	dictLastTimeExposedForPerson	<b>establishOffspringConnectionsBasedOnMothersConnections</b>
initialAge	appearanceTime	dictAllTimesInfectedForPerson	establishOffspringLocationBasedOnMothersLocation
ethnicity	nextIdForNewPerson	reportAllTimesInfectedForPersons	finalizeDeath
randomSex	strName	saveInfectionHistoryInformation	reportChildren
randomEthnicity	circleSize	saveExposureAndInfectionMessageInformation	

The 'Function body' section shows the following code:

```
// 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);
    }
```



# Code to Perform Birth

The screenshot displays the NetLogo interface with a workspace containing four parameter lists: Person Parameters (sex, mother, initialAge), System Parameters (color, circlesize, appearanceTime), Disease Parameters (dictLastTimeInfectedForPerson, dictLastTimeExposedForPerson, dictAllTimesInfectedForPerson), and Pregnancy Parameters (performBirth, establishOffspringConnect, establishOffspringLocation). The 'performBirth' function is selected in the Pregnancy Parameters list. Below the workspace, the 'performBirth - Function' editor shows the following code:

```
Person mother = this;
Person offspring = get_Main().add_Population(0.0, 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 across field of view *)
establishOffspringLocationBasedOnMothersLocation(offspring, mother);
mother.children.add(offspring);
```

At the bottom of the interface, there are expandable sections for 'Advanced' and 'Description'.

# Establishing Baby's Connection

## Looping over Connections

The screenshot displays a software development environment with two main panels. The top panel shows a visual model with three sections: Parameters, Disease Parameters, and Pregnancy Parameters. The Parameters section includes a 'children' icon. The Disease Parameters section lists three variables: dictLastTimeInfectedForPerson, dictLastTimeExposedForPerson, and dictAllTimesInfectedForPerson. The Pregnancy Parameters section lists three functions: performBirth, establishOffspringConnectionsBasedOnMothersConnections, and establishOffspringLocationBasedOnMothersLocation. A diagram on the right shows a flow from 'InfectionStatechart' to a diamond-shaped decision node, which then branches to a green box labeled 'Susceptible' and a red box. The bottom panel shows a code editor with the following code:

```
// 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
```

The code editor also has a 'Properties' tab and a 'Progress' indicator. The bottom of the code editor shows a 'Advanced' button.

# Setting Offspring Location

The screenshot displays a software development environment with a main workspace and a function editor. The main workspace contains a diagram with a central circle, a vertical line, and a green box labeled 'Susceptible'. A red box labeled 'InfectionStatechart' is also visible. The function editor shows the following code:

```
double dOffspringDirectionFromMotherInRadians = uniform(2 * 3.14159);  
double offspringDistanceFromMother = get_Main().offspringDistanceFromMother;  
  
offspring.setXY(mother.getX() + offspringDistanceFromMother * Math.cos(dOffspringDirectionFromMotherInRadians),  
               mother.getY() + offspringDistanceFromMother * Math.sin(dOffspringDirectionFromMotherInRadians));
```

The interface includes a 'Parameters' panel on the left with sections for 'Disease Parameters' and 'Pregnancy Parameters'. The 'Disease Parameters' section lists:

- dictLastTimeInfectedForPerson
- dictLastTimeExposedForPerson
- dictAllTimesInfectedForPerson

The 'Pregnancy Parameters' section lists:

- performBirth
- establishOffspringConnectionsBasedOnMothersConnections
- establishOffspringLocationBasedOnMothersLocation

The 'Properties' panel on the right shows a 'Controls' section with various UI elements like Button, Check Box, Edit Box, Radio Buttons, Slider, and Combo Box.