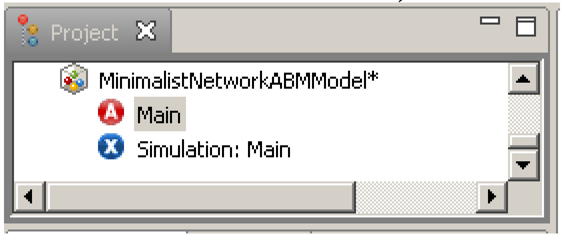
# BUILDING A MINIMALIST NETWORK-BASED MODEL FRAMEWORK

1. Enter AnyLogic
2. ***Please be advised that you may have to close a small "Update" notice before you reach the main window.***
   If the "Welcome to AnyLogic 6" screen appears, close it by clicking on the white "X" located on the right side of the tab entitled "Welcome".
3. Create a New model project entitled "MinimalistNetworkABMModel" by following these steps:
   a. Under the menu item "File", Choose "New", and select "Model"
   b. For model name, enter the word "MinimalistNetworkABMModel" (without quote marks)
   c. Press "Finish"

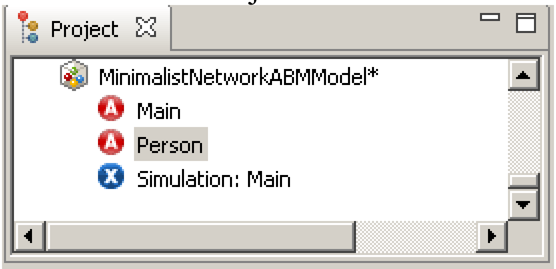
   The Main class will now be displayed.
   We will now create a "Person" "class". At a rough level, this class is a structure in the model that will specify what it means to be a person within our model – the information associated with a person, the type of behavior characteristic of a person, and the visual representation to be used for a person. The process for adding such a person has several steps:
   d. Right click on the project name (this is called "MinimalistNetworkABMModel in the "Project" menu that should be visible, and should look like the below)



MinimalistNetworkABMModel*
Main
Simulation: Main

   e. Choose "New", and select "Active Object Class" (labeled with a red "A" in a circle)
   You should be presented with a "Dialogue Box" entitled "Active Object Class". Fill out this information as follows.
      i. In the "Name" field, Type "Person" (without quote marks)
      ii. Press "Finish"

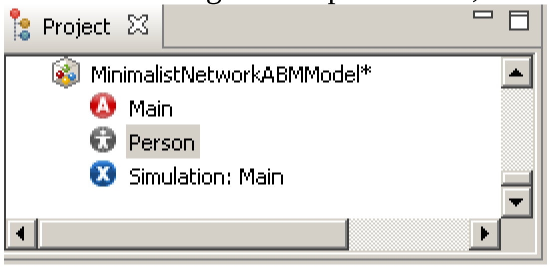
   There should now be a "Person" "active object" class shown in the hierarchical list in the "Project" window on the left, looking like this



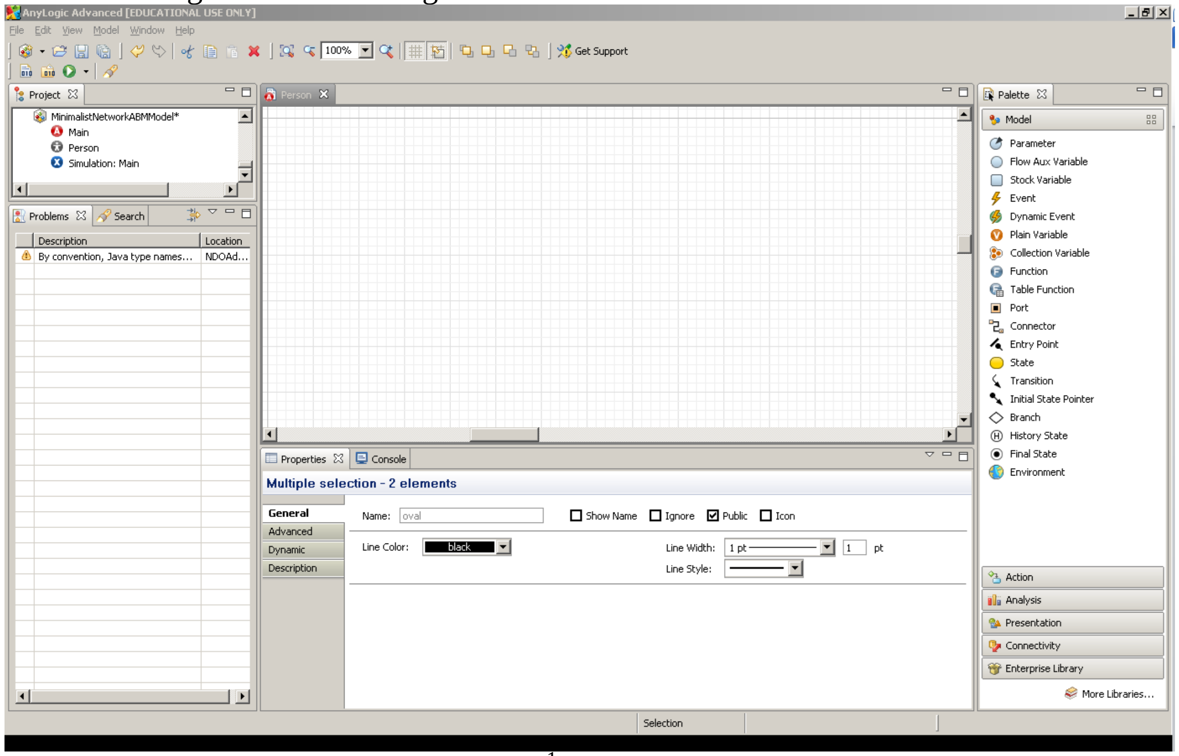
MinimalistNetworkABMModel*
Main
Person
Simulation: Main

   f. Click on the "Person" class (labeled with "Person").
   This will bring up the properties of "Person" in the "Properties" window.
   g. We need to make sure that AnyLogic is aware that "Person" is an Agent class – to be used to describe the agents in populations. We do this as follows: On

the “General” table of the properties for “Person”, click on the box that says “Agent”.
You will note that, after a brief delay, the icon for “Person” should change to a grey circle with the figure of a person in it, like this:



4. Because you just created the “Person” class, there should now be a tab for “Person” along the top of the screen showing on the right side of the screen. The screen should look something like the following



| Description | Location |
|---|---|
| By convention, Java type names... | NDOAd... |

Multiple selection - 2 elements

Name: oval ☐ Show Name ☐ Ignore ☑ Public ☐ Icon

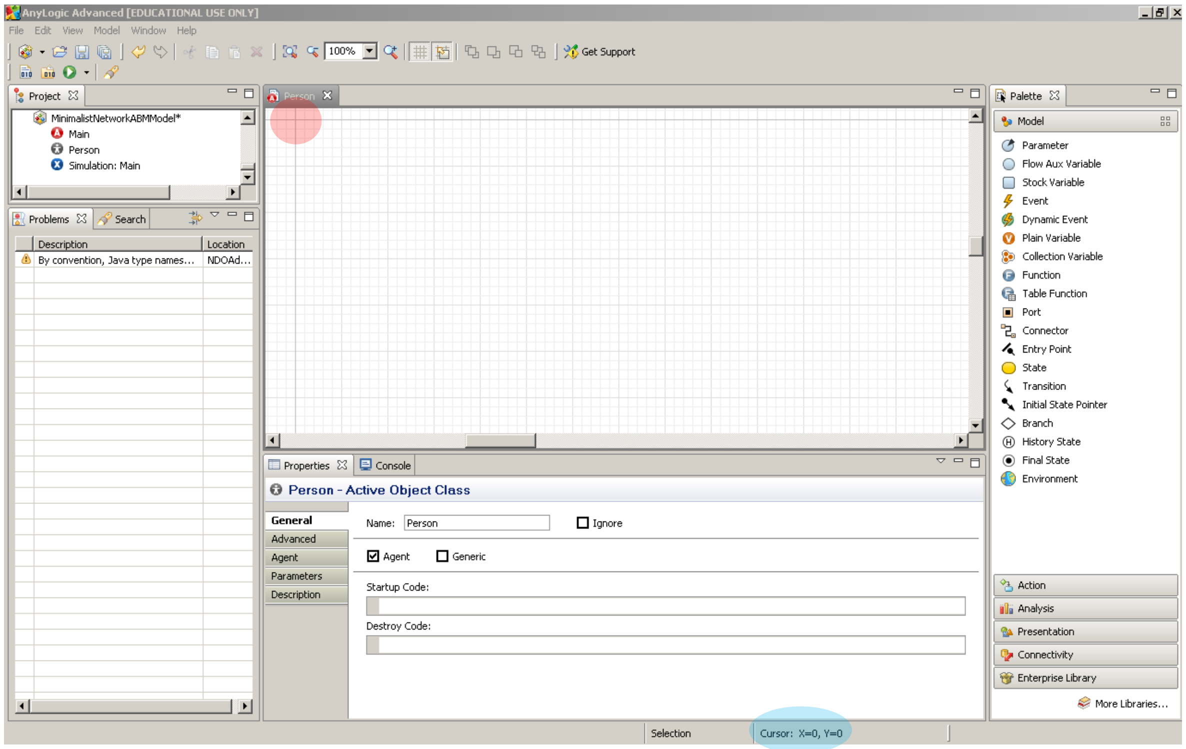
Line Color: black Line Width: 1 pt 1 pt

Line Style:

5. Add a visual representation for a “Person”[1]. We will first create a minimalist representation, and will then refine it. This includes several steps
   a. Within the canvas (Entitled “Person” to the right),
      i. use the scroll bar on the bottom to scroll slightly to the left and use the scroll bar on the right to scroll slightly up, until you seen a “cross” of two major lines appear. This is the “origin” of the presentation space. For now, just make sure that this “origin” is clearly visible. (Note that you can also locate this origin by clicking on this space, and watching the numbers next to the “Cursor” label at the bottom of the screen as

[1] Note that while it is possible to continue to refine a visual representation for a person once created, creating such a representation – using actions similar to the below – must be undertaken before we request a population of persons in Main in later steps.

per Figure 1 below. When this reads "X=0, Y=0", the cursor is pointing at the origin.)



Person - Active Object Class

Name: Person ☐ Ignore

☑ Agent ☐ Generic

Startup Code:

Destroy Code:

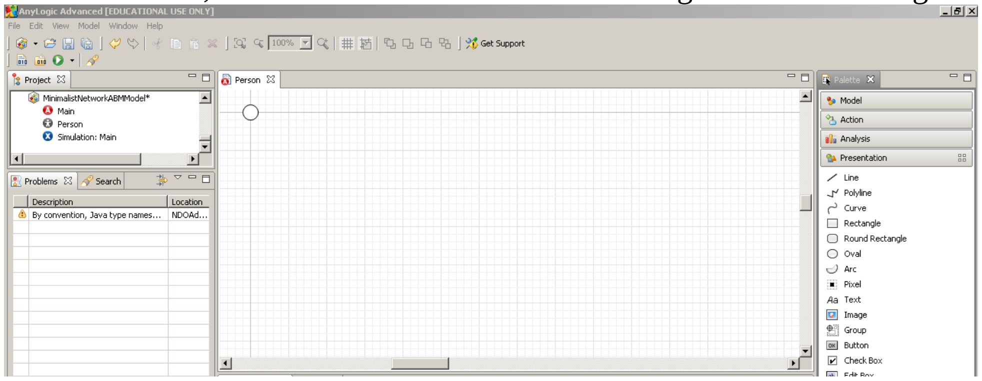
| Description | Location |
|---|---|
| By convention, Java type names... | NDOAd... |

Cursor: X=0, Y=0

**Figure 1: The Canvas, Showing the Origin (Cross of two Major Lines, Highlighted in red). This can also be located by reference to the cursor position (highlighted in blue)**

b. Select the "Presentation" label near the bottom of the window labeled "Palette" on the right side of the screen.

c. Add an "Oval", centered at the origin: drag from the "Oval" from the "Palette" window to the area around the origin
Adjust the oval so that it is circular, and is located at the origin. When you are finished, the result should look like something like the following:

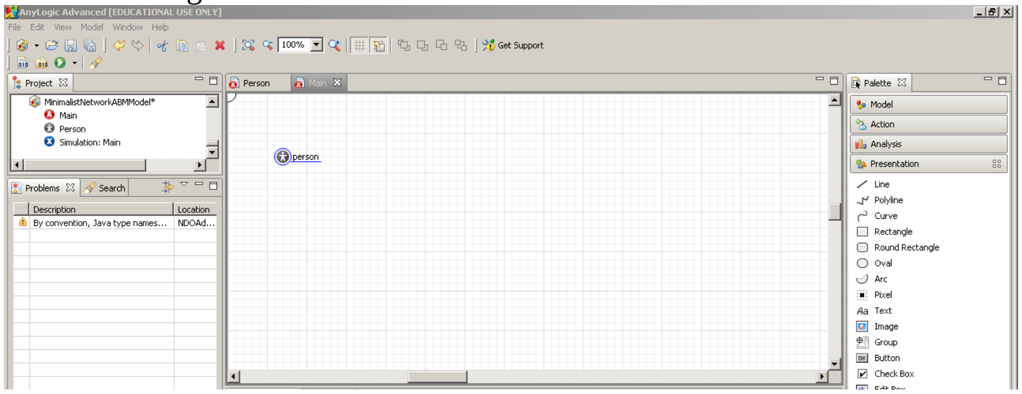


| Description | Location |
|---|---|
| By convention, Java type names... | NDOAd... |

6. Having created the visual presentation above, we can now create a population of such agents. To do this, double-click on "Main" (with the red circle with an "A" next to it) in the "Project" menu on the right hand side to open up the canvas for "Main" on the right. There should now be a tab for "Main" along the top of the screen showing

on the right side of the screen. Right now, that canvas associated with "Main" should be empty.

7. Click (just once!) on "Person" in the "Project" window on the left, and – keeping the mouse button depressed – drag to the canvas for "Main" to the right, and release the mouse button. There should now be an item labeled "person" to the right, as well as a visual representation associated with person on the upper left. The screen should now look something like the below.

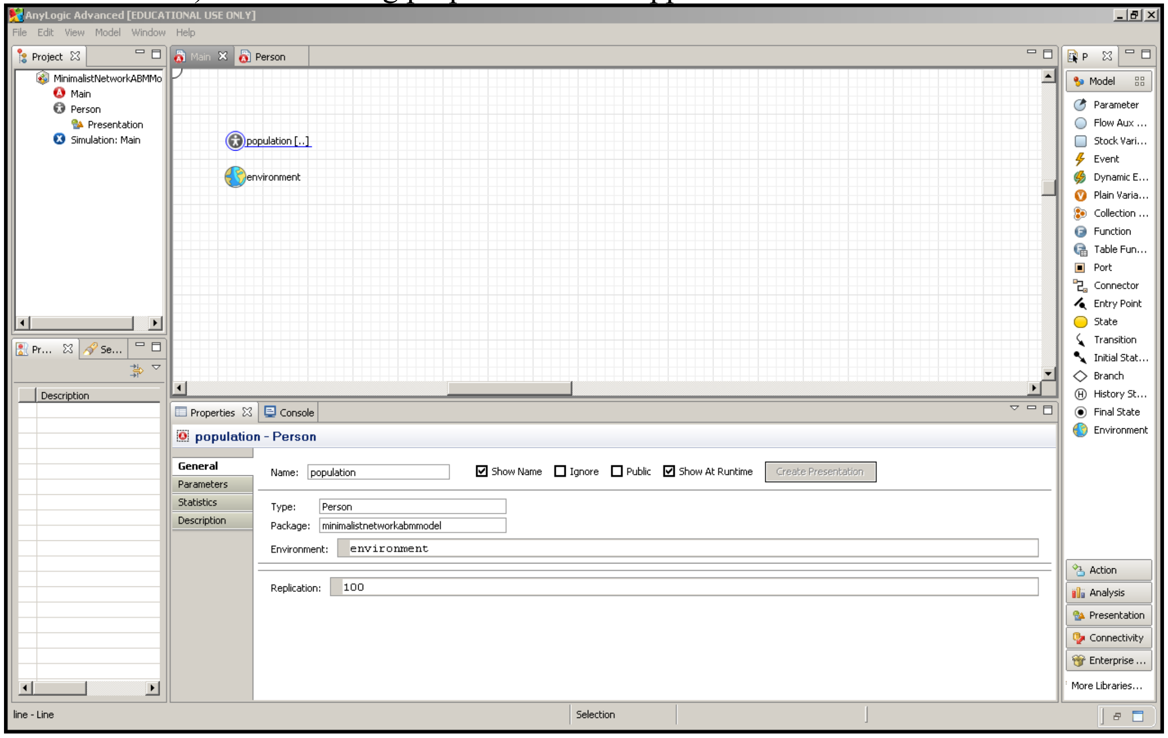


| Description | Location |
|---|---|
| By convention, Java type names... | NDOAd... |

   a. Click on the new item labeled "person" (in case it isn't already selected), and the "properties" for person should show below in the "Properties" window.
      i. In the "Name" field, change the name from "person" to "population".
      ii. In the "Replication" field, enter "100"

8. We now have a population (if a rather uninspiringly homogenous population) of 100 agents. We'll introduce a bit of context by introducing an *environment*, which will allow agents to be placed at different locations, and to be connected to different agents. To do this, proceed to follow the steps below:
   a. In the "Palette" window, click on "Model". This will show a large number of items related to model logic that can be added to the "Main" canvas, all grouped under the "Model" heading.
   b. Add an Environment to the main canvas. This follows a similar procedure for adding the Oval above –drag "Environment" from the "Oval" from the palette to the area around the origin
   The name for this environment can be left as its default ("environment"). No other properties need to be modified at present.
9. Now that an environment has been added, we have to associate the population of agents with it.
10. Click on the population of agents (the item labeled "population" within the canvas for Main). The properties for the population should now be shown. In the field labeled "Environment", type the name of the environment (in other words, the word

"environment"). The resulting properties should appear as follows:



**population - Person**

Name: population [✓ Show Name] [ ] Ignore [ ] Public [✓ Show At Runtime]

Type: Person
Package: minimalistnetworkabmmodel
Environment: environment

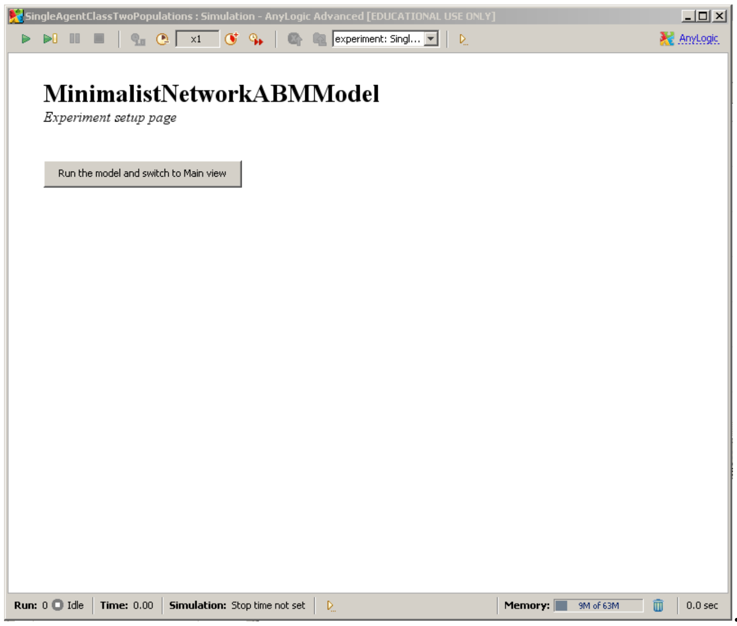
Replication: 100

We should now have a model which is runnable, and where the agents are associated with different locations.

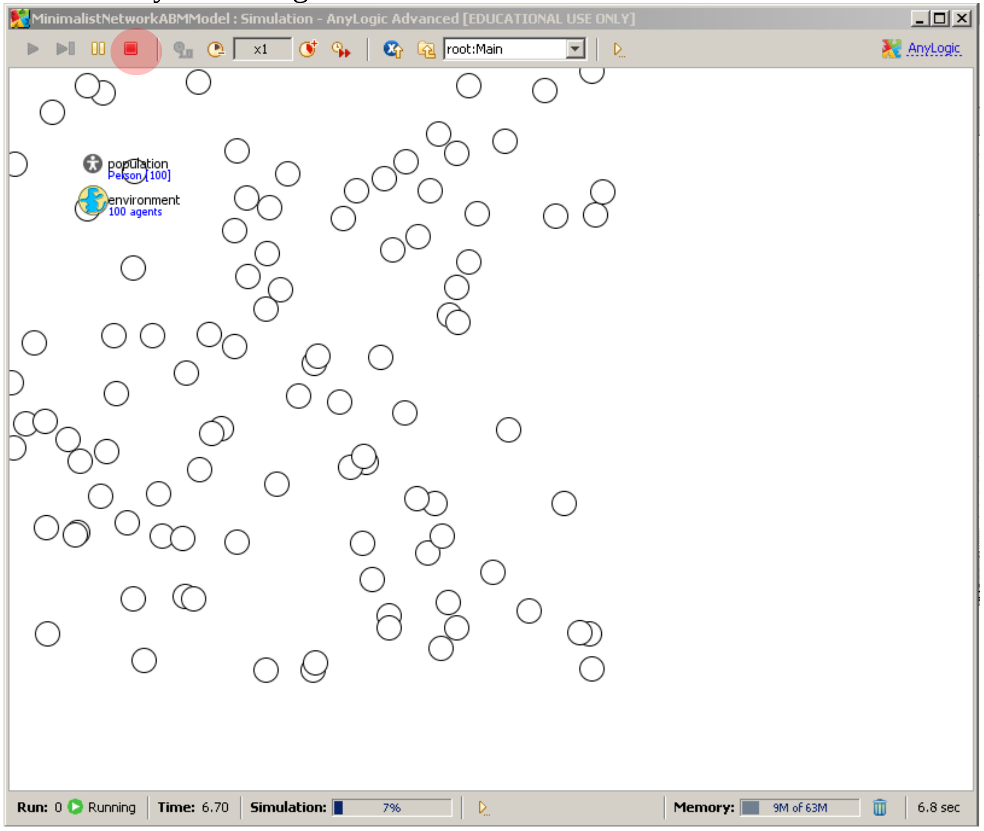
In order to simulate this model within AnyLogic, we need what AnyLogic terms an "experiment". Such an experiment will essentially tell AnyLogic what to assume about the values of "parameters" – that is, the constant values to assume for different model quantities (such as the mean recovery time from a disease, or the size of the population).

Here, our model remains so simple that it does not yet have parameters – but AnyLogic has created a default "experiment" for it. This is shown in the "Project" window, and has been given name "Simulation". Because this experiment is based around assumptions to the "Main" class, AnyLogic has labeled it "Simulation: Main". It should be visible in the "Project" window just to the right of a blue circle containing a white "X".

11. Check that the model is runnable by right clicking (Control-Command on a Mac) on the experiment in the "Project" window labeled ("Simulation: Main"), and selecting "Run".
    a. If you have been following the steps above very closely, a screen something like the following should pop up.


SingleAgentClassTwoPopulations : Simulation - AnyLogic Advanced [EDUCATIONAL USE ONLY]
x1
experiment: Singl...
AnyLogic
MinimalistNetworkABMModel
Experiment setup page
Run the model and switch to Main view
Run: 0 Idle
Time: 0.00
Simulation: Stop time not set
Memory: 9M of 63M
0.0 sec

Press "Run the model and switch to Main view", and you should see a screen like the below. The area labeled "Time" should show a continuously increasing number next to it.



Press the "Stop" button (highlighted above in red)

Sanity check: If you were unable to see the above, we suggest you double-check some items.

If the model won't run, look in the "Problems" window for clues

**Exercise Rating: Basic**

1. Is it possible that you misspelled-one of the items that you had to add? (i.e. in a field, or in the name in the model)
2. Is it possible that you didn't fill in "100" for the size of the population?
3. Is it possible that the population in "Main" is mislabeled "Person" instead of "population"?

If the model does run, but doesn't show the above, suggestions will vary by the circumstances

ii. If you do see circles, but they are not spread out across the screen
   1. Double check that you've associated the population with the environment (per step 9).

iii. If you don't see circles at all
   1. Is it possible you didn't add the Oval to the "Person" before creating the population? To check that,
      a. Make sure that you've created the person's visual representation, per Step 5 above
      b. Delete the population ("population") from the main class.
      c. Recreate that population by dragging in from the "Person" class to the Main canvas (per Step 6), and then make sure that this population is associated with the environment by setting the "Environment" field to refer to the name of the environment ("environment"), per step 9.
      d. Try re-running the model.

12. The model at present associates people with physical locations, but suppose (for the purpose of understanding the transmission of pathogens, norms, ideas, or other factors) we'd like to see these physical locations affect the contact patterns between individuals. We accomplish this task here using two steps.
   a. First, we set the connections between people to depend on their distance from each other. Individuals less than a certain threshold of distance (a "connection range'') from each other will be treated as in contact; those with a further distance from each other will be treated as not connected.
      i. Make sure that the canvas for the "Main" class is showing by double-clicking on "Main" within the "Project" window.
      ii. Select the environment (labeled "environment") by clicking on it. The "Properties" window should now be showing the properties for the environment.
      Now click on the "Advanced" tab. This shows an area where we can state assumptions about the spatial and/or network environment in which agents should run (and that are to be controlled by the "Environment" object).
      iii. Click on the menu item to the right of "Network type", and scroll down until you select "Distance based" (note that "Distance based" may not be visible when this opens, and in order to see it in the list, you may have to scroll down.).

iv. Note that the "Connection range" Field is showing. For the sake of this assignment, you may leave it at its default value (50), but recognize that it may change.

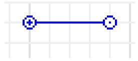
b. Second, we allow people's visualization to indicate whether or not they are connected to another person. Accomplishing this will demonstrate one of AnyLogic's most powerful and appealing features – declarative logic for properties – but will also entail use of some logic that is a bit more involved.

i. Double click on "Person" in the "Project" window. The visual representation of person (the circle you created earlier) should be visible in the upper left. Click on the scrollbars on the bottom and right of the window until you see the entirety of the circle.

As a tip, note that an alternative approach to viewing this circle would be to select the "oval" in the "Project" window. To do this, click on the "+" to the left of "Person" in the "Project" window, and then on the "+" to the left of "Presentation" in the same window. The oval should now be visible. Simply double-click on that oval, and you will be presented with the canvas for "Person".

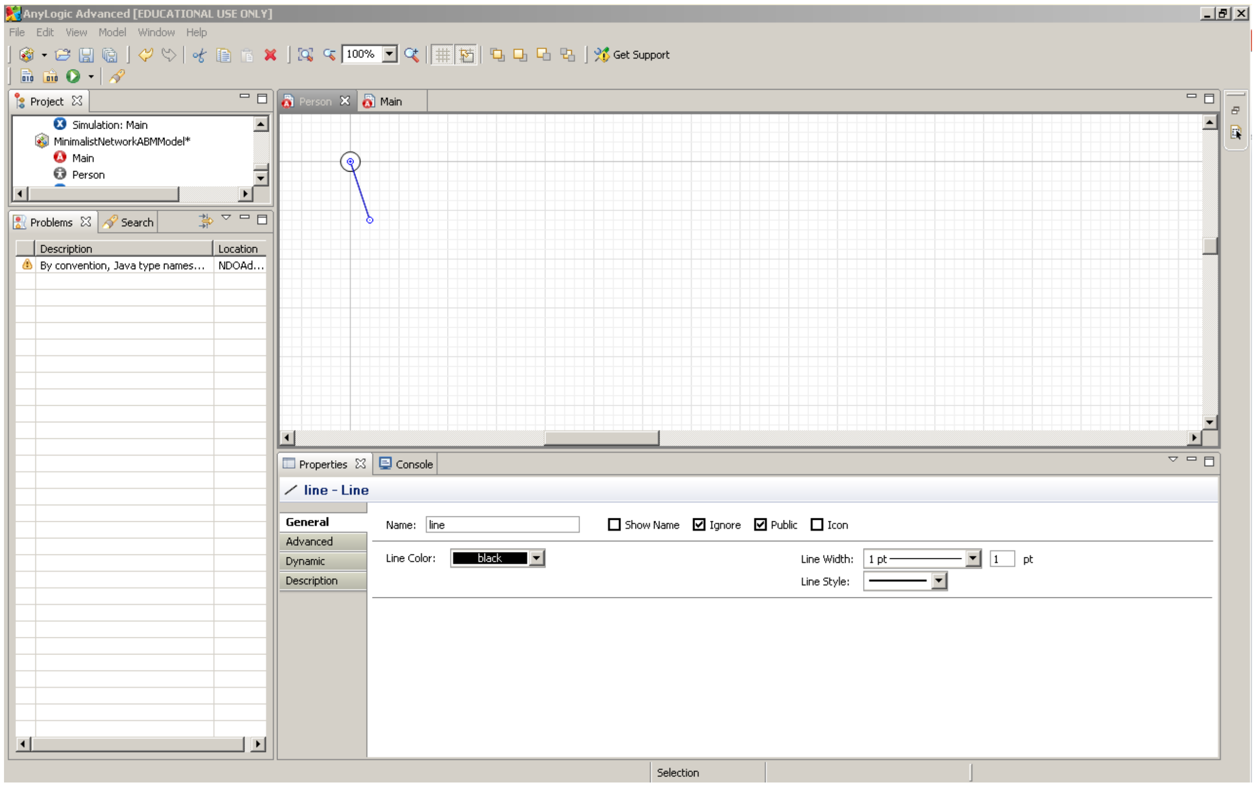
ii. Using the same general procedure used in the previous steps for adding items from the palette to the canvas, add to the diagram a line from the "Presentation" tab in the "Palette" window; the line can go anywhere as we will refine its position shortly Note that – as is similar to the case for adding the Oval and Environment above – drag the Line from the "Line" from the palette to the area around the origin

iii. Note: the positioning of this line is important. Specifically, notice that you click on the line to select it, it has one side that is labeled with a "+", and another with a dot". That is, close up, it looks like this:



Orient the line you have added so that the "+" end (which is actually the first point drawn on the line) is located at the origin (i.e. at the middle of the oval), the other (i.e. "-") end may located anywhere else, but it's probably best to make it long enough that you can easily select it with the mouse. (When adding this line in AnyLogic 6.2.2, you may find it simpler to simply click at the middle of the circle when drawing the line, and then click elsewhere. The endpoint of the line associated with the first "click" is that labeled "+", and will therefore be located in the middle of the circle.)

The resulting canvas should look similar to the following:



| Description | Location |
|---|---|
| By convention, Java type names… | NDOAd… |

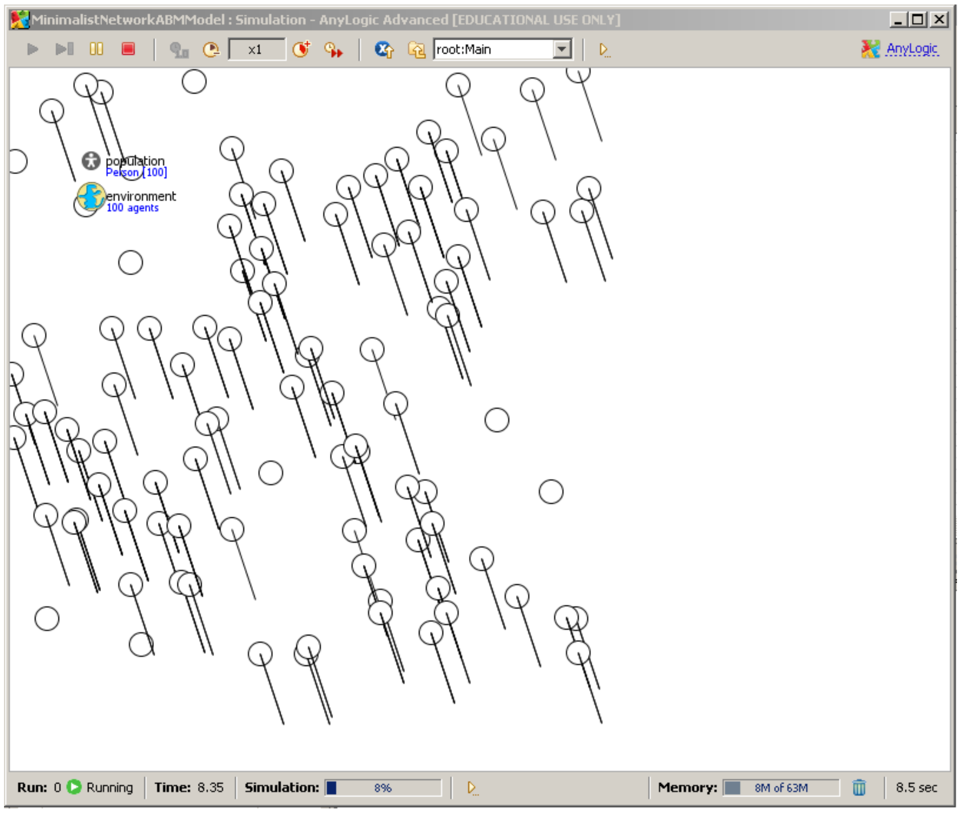
Name: line ☐ Show Name ☑ Ignore ☑ Public ☐ Icon

Line Color: black

Line Width: 1 pt 1 pt

Line Style:

iv. As an experiment, try running the model by right clicking (Control-Command on a Mac) on the experiment labeled (“Simulation: Main”), and selecting “Run”. (For added information on this process, see Step 11). Once you start the simulation using the button on the introductory screen. You should see something like the following. This is somewhat reminiscent of what we are seeking, but is missing some key features – namely, variability in the number and orientation of connections for each node.



v. Returning to the "Person" canvas, click on the line you added to select it. The properties for the line should now be shown in the "Properties" window. Click on the "Dynamic" tab in the "Properties" window to view optional definitions of the properties for the line. You will now specify 3 of these properties using formulae – technically, called "expressions" in Java – that give their value. At any given time and for a particular individual, a given formula (expression) will evaluate to a particular value. Specifying such a formula for the value of a property will allow the value of that property to vary for different individuals, and to be kept consistent as the characteristics of the agent (Person) vary over time.

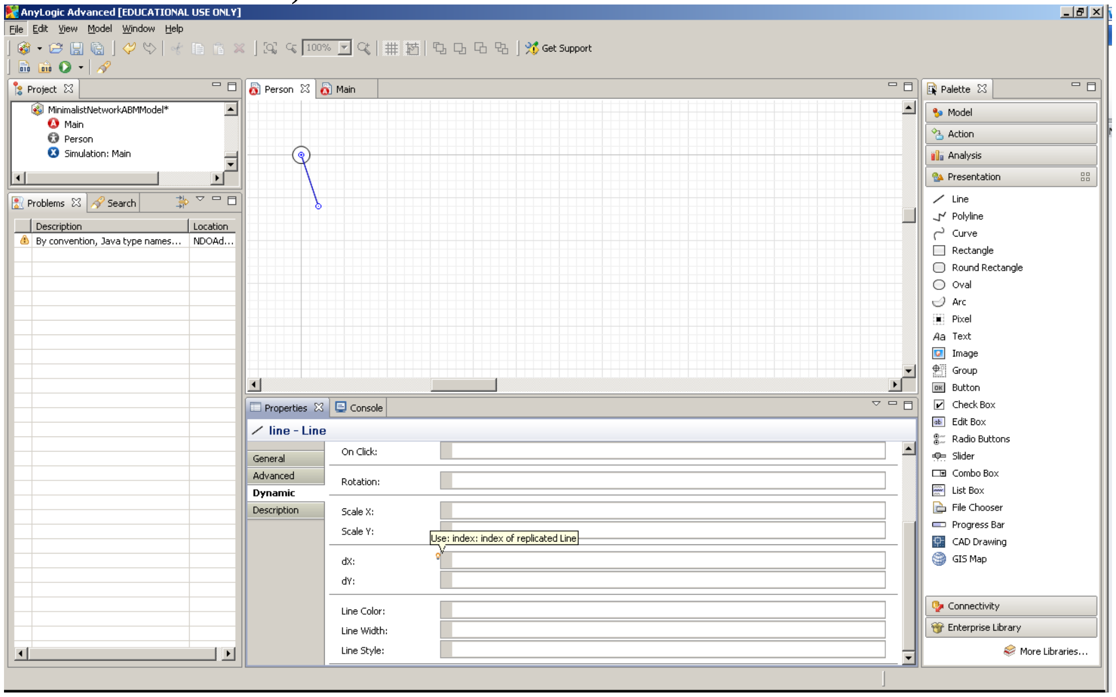
1. Scroll as necessary to find the "Replication" property. The value associated with this property specifies the number of "copies" of the line to create for this particular agent. Enter the following expression: "this.getConnectionsNumber()". Evaluating this expression for a given agent (referred to by "this") will return the number of connections associated with this agent. This reflects the fact that while there was only one line we drew in the canvas for a Person, for a specific person during simulation, there could be multiple copies of that line required – because that person has multiple connections.

13. Scroll as necessary to find the "dX" property. In contrast to the "Replication" property above (which specifies a single value for the line item for the agent), this property is specific to instances (replications) of the line. As we indicated with the "Replications" property, a given person might be associated with many possible

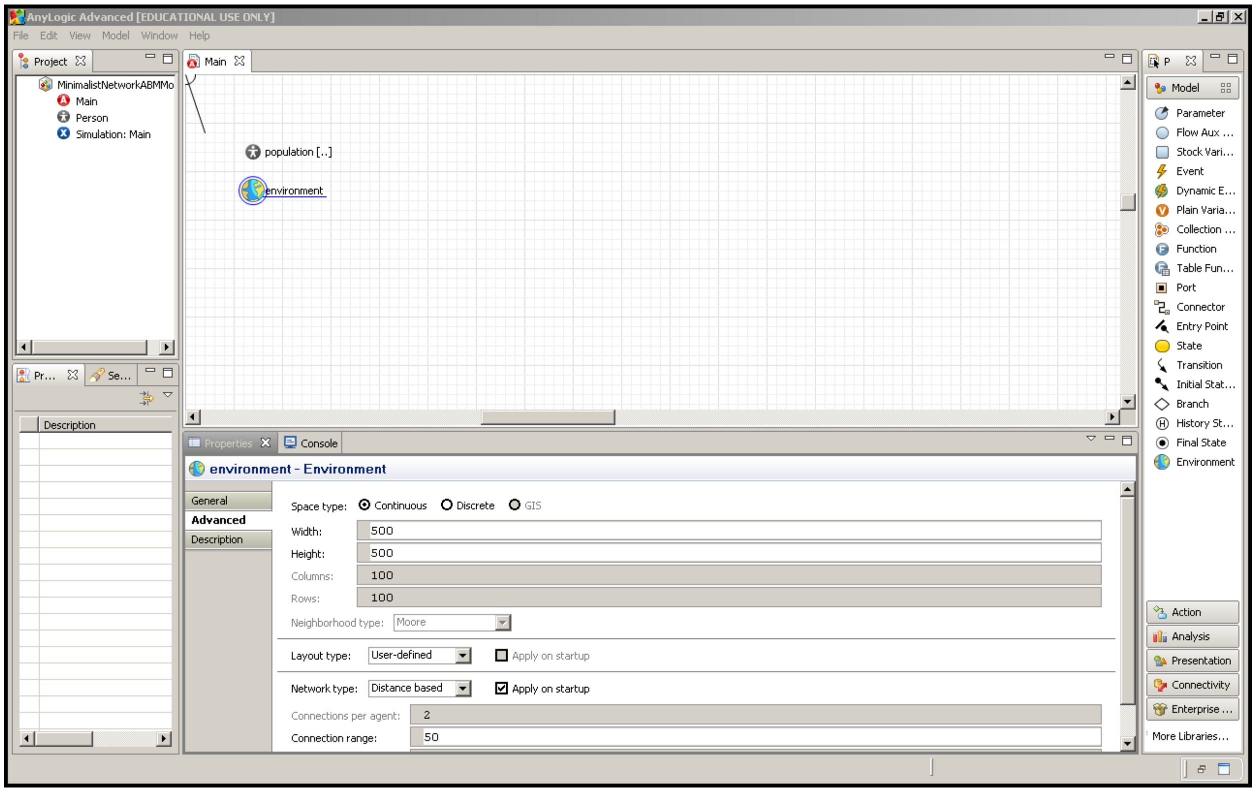
replications of the line – one associated with each connection.
Here, recall that we wish to use a line to visually depict a particular connection between the current agent (associated with the origin of the line) and another agent to which it is logically connected. This visual line we wish to depict will be specific to the particular connection involved – i.e. to a specific "replication" of the visual line we previously drew. The connection number associated with this line is given by the variable "index". As shown in the figure below, this can be determined by hovering the mouse over the "light bulb" symbol that appears just to the left of the text field.
The value for this property specifies the horizontal span over which the line should extend. A positive value indicates that the line travels to the right from its origin; a negative value indicates that the line travels to the left from its origin. Here, based on some reasoning about the relative position of the person at the other end of the connection relative to the current person[2], and knowing that "getConnectedAgent(connectionNumber)" returns a reference to the connectionNumber$^{th}$ agent (here, person) connected to this one, calling the method "get the value for the "dX" property should be "this.getConnectedAgent(index).getX() - this.getX()". (See the footnote below for

[2] Here, we wish to use a line to visually depict a particular connection between the current agent (associated with the origin of the line) and another agent to which it is logically connected. The horizontal span of the line will be dictated by the difference in horizontal position between these two agents. Specifically, the span of the line should be such that if that span is added to the current agent's horizontal position (given by its X coordinate) then it would yield a value equal to the horizontal position (X coordinate) of the other agent. In other words, the length of the line is just X coordinate of the other agent minus the X coordinate of the current agent.
To calculate this value, we need to find out these two X coordinates. If we have a reference to a given agent, we can simply request the X coordinate associated with that agent by calling the "getX()" method on that reference. Since the reference "this" refers to the current agent, to retrieve our own X coordinate we can simply call "this.getX()".
To determine X coordinate for the agent associated with the connection for this line, we first need to get a reference to that agent. To do this, we can use the "getConnectedAgent(connectionNumber)" method on the current agent – just specifying the particular connectionNumber whose reference we wish to retrieve. Calling this using a reference to a given agent will return us a reference to the connectionNumberth agent to which that given agent is connected. The connection number associated with this line is given by the variable "index". As shown in the figure below, this can be determined by hovering the mouse over the "light bulb" symbol that appears just to the left of the text field. To find a reference to the connected, agent, we can thus call "this.getConnectedAgent(index)". The X coordinate for that agent is then given by the expression "this.getConnectedAgent(index).getX()". The value for the "dX" property should therefore just be "`this.getConnectedAgent(index).getX() - this.getX()`"

additional information)



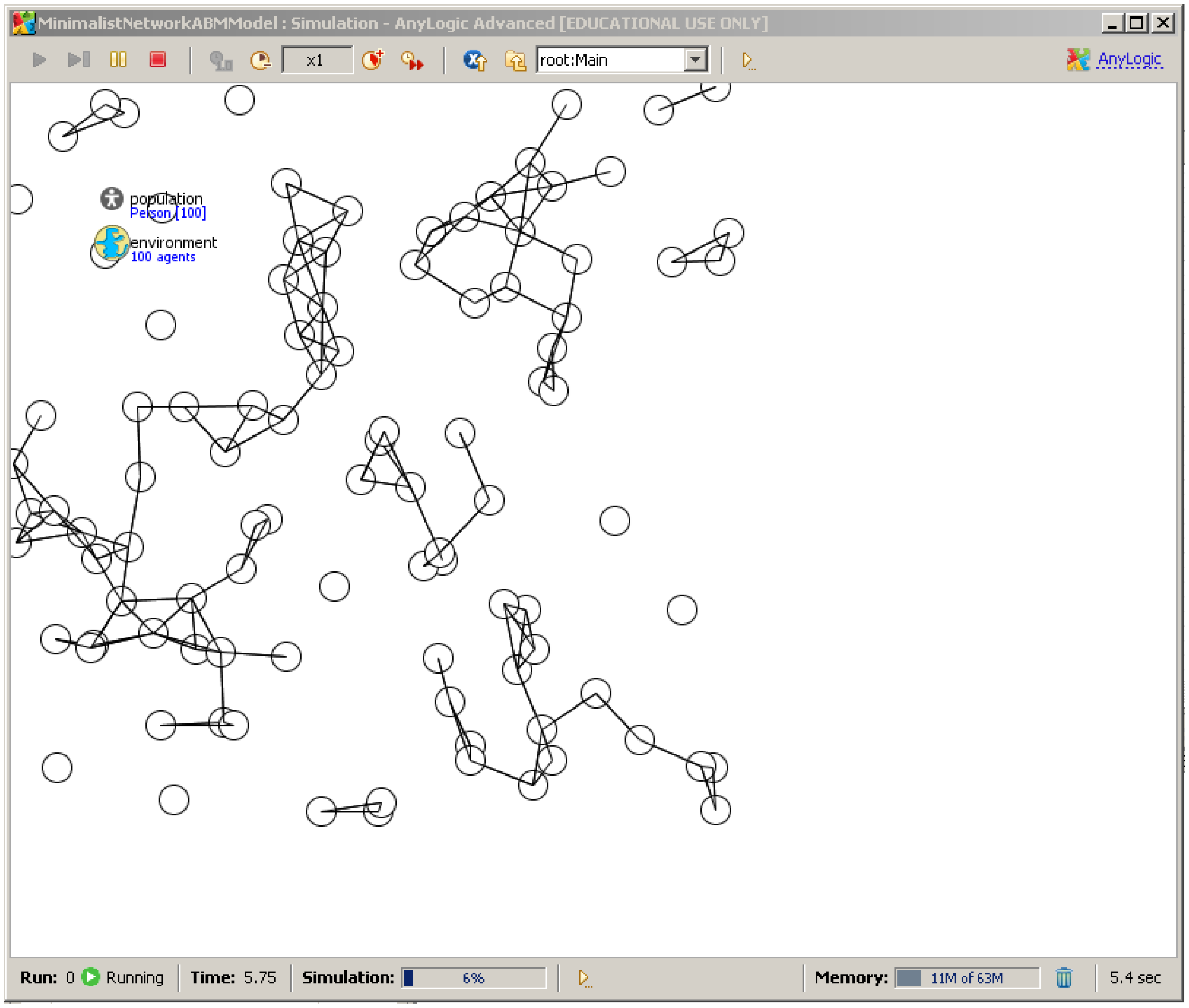
14. Scroll as necessary to find the “dY” property. The situation with this property is entirely analogous to that for the “dX” property handled above – with the exception of the fact that instead of dealing with the horizontal position (as given by the X coordinate), we are now dealing with the vertical position (as given by the Y coordinate). In a manner analogous to the above, the vertical span of the line will be dictated by the difference in vertical position between these two agents. The value to use for the “dY” property is thus “`this.getConnectedAgent(index).getY() - this.getY()`”. We note that this is the same expression as for the dX property, except that the two calls to the “`getX`()” function (method) were replaced by calls to the “`getY`()” function.
15. The final thing that must be done before running the model is to set the network type in which these people will be arranged. We accomplish this with a series of steps.
    a. Double-click on “Main” in the “Properties” window to view the “canvas” for the “Main” class.
    b. Click on “environment” (with a shape like a globe), and view the properties for the environment” in the “Properties” window.
    c. Click on the “Advanced” tab of the “Properties” window.
    d. Using the “Dropdown” menu located to the right of the label “Network type”, set the “Network type” to “Distance based”
    e. For the “Connection range” 2 lines down, use the value “50” (without quotes).
    f. The resulting properties for “Environment” should look as below:



environment - Environment

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: 50

16. Please run the model by right clicking (Control-Command on a Mac) on the experiment labeled ("Simulation: Main"), and selecting "Run". (For added information on this process, see Step 11). You should see something similar to the below:



Santy check: If you were unable to see the above, we suggest you double-checking some items.

If the model won't run, look in the "Problems" window for clues

1. If the "Problems" window complains that "index" is not known, have you left out the definition for "Replications" associated with the line in "Person"? See Step 12.b.v.1 above.
2. Is it possible that you made a mistake in entering the formula one of steps 12.b.v.1, 13, or 14? Look for missing periods – they are easy to miss, but make a big difference. If it's not clear where the problem lies, try an experiment: Try saving away the model under a different name, and replacing each formula in turn by a fixed number (e.g. "10") and seeing if this changes the error message.

If the model does run, but doesn't show the above, suggestions will vary by the circumstances.

**Exercise Rating: Basic**

Check to make sure that the formula is correct, (for example that you changed *both* “getX()” calls to “getY()” calls in adapting the formula from “dX” for “dY”)
As above, try experimenting to see if you notice changes that could give a hint as to the nature of the problem.

**Exercise Rating: Basic**