Collecting, Outputting & Inputting Data in AnyLogic

Nathaniel Osgood

Using Modeling to Prepare for Changing Healthcare Needs

Duke-NUS

April 16, 2014

Recording of Results

- A frequent modeler need is to record some components of model state over time
  - State variables (e.g. stocks)
  - States of agents
  - Summaries of model state
  - We informally term this a “trajectory file”
- *Trajectory recording is supported in higher AnyLogic versions*
- All versions of AnyLogic allow for
  - Definition of *DataSets* that record recent values of parameters
  - Statistics summarizing model state
  - Reporting on values of data sets as a graph or table

What to Record?

- Current model state
  - Aggregated (“Stocks”) or disaggregated
- Changes in model state (flow statistics)
- History
- Model version (model structure)
- Assumptions (parameter values)
  - Interventions

Common Ways to Accumulate Flow Statistics

- Have a variable that
  - is zeroed out at the beginning of each time interval of interest
  - Is accumulated across that interval of interest
  - Is stored away at the end of that interval interest
- Option2
  - Have a variable that gives the cumulative number of events that have occurred
  - Subtract off the value of that variable from the beginning of the interval and store away the difference

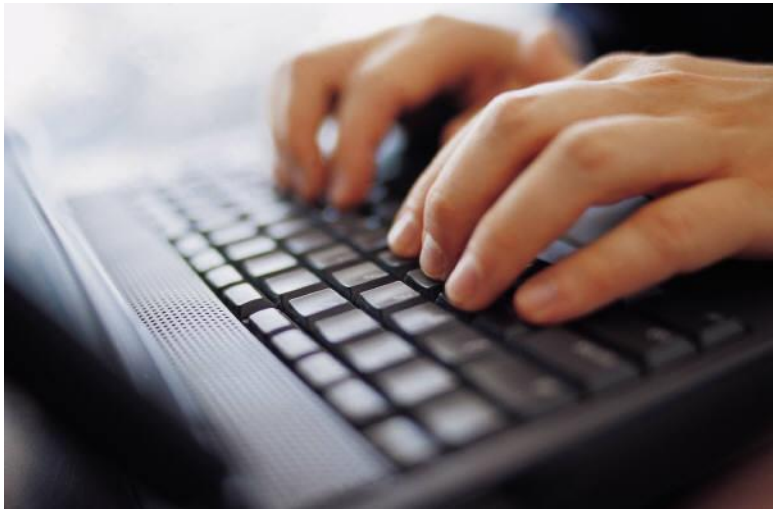
Techniques for Outputting Data

- Ad-Hoc Exports from variables
- Manual copies from visible datasets
- Export to files
- Writing to console
- Export to databases
- [AnyLogic Professional] Dataset archiving
- Capturing images of graphs

Cross-Method Output Tips

- A convenient mechanism is to periodically output data using events (e.g. every time unit)
- Beyond output, be sure to save information on context of run
  - Model version (Use unique id that increment whenever change model)
  - Parameter assumptions
  - Intention
- Think carefully about whether want to save away intermediate data

Hands on Model Use Ahead



Load Sample Model:

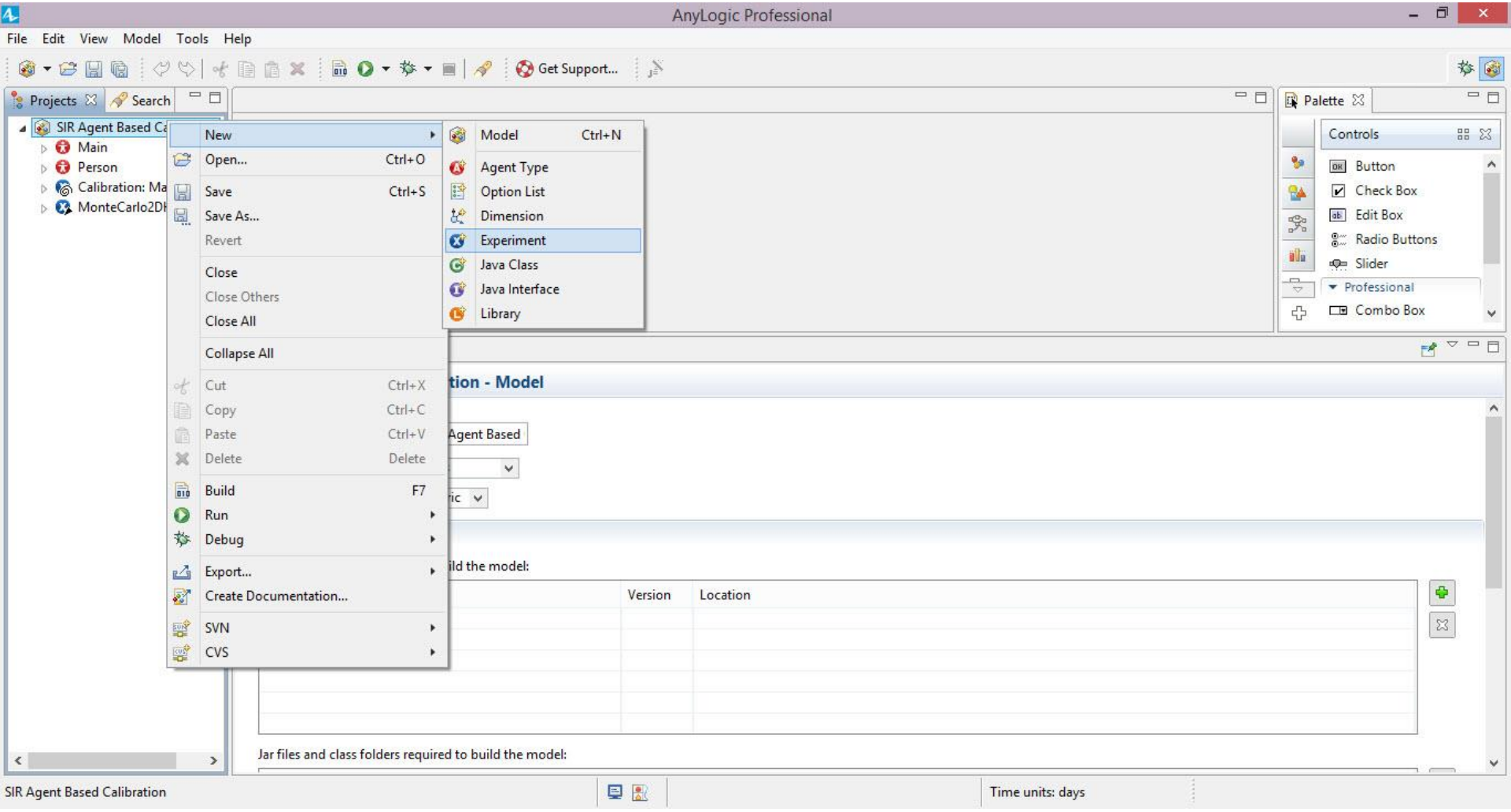
SIR Agent Based Calibration

(Via “Sample Models” under “Help” Menu)

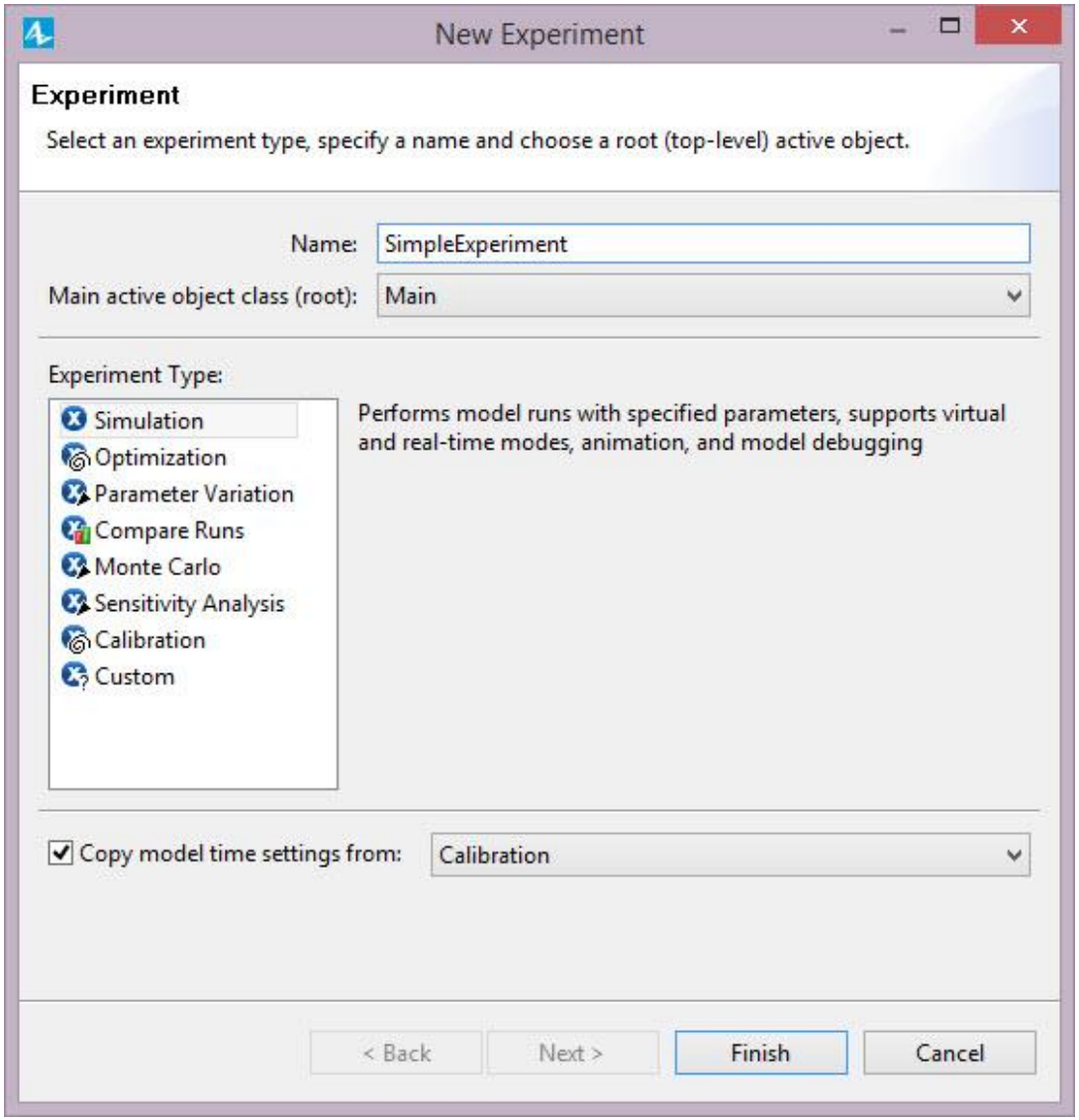
Techniques for Collecting & Outputting Data

- Ad-Hoc Exports from variables
- Pre-Prepared methods
  - Statistics
  - Charts
  - Manual copies from visible datasets
  - Export to files
  - Writing to console
  - Export to databases
  - [AnyLogic Professional] Dataset archiving
  - Capturing images of graphs

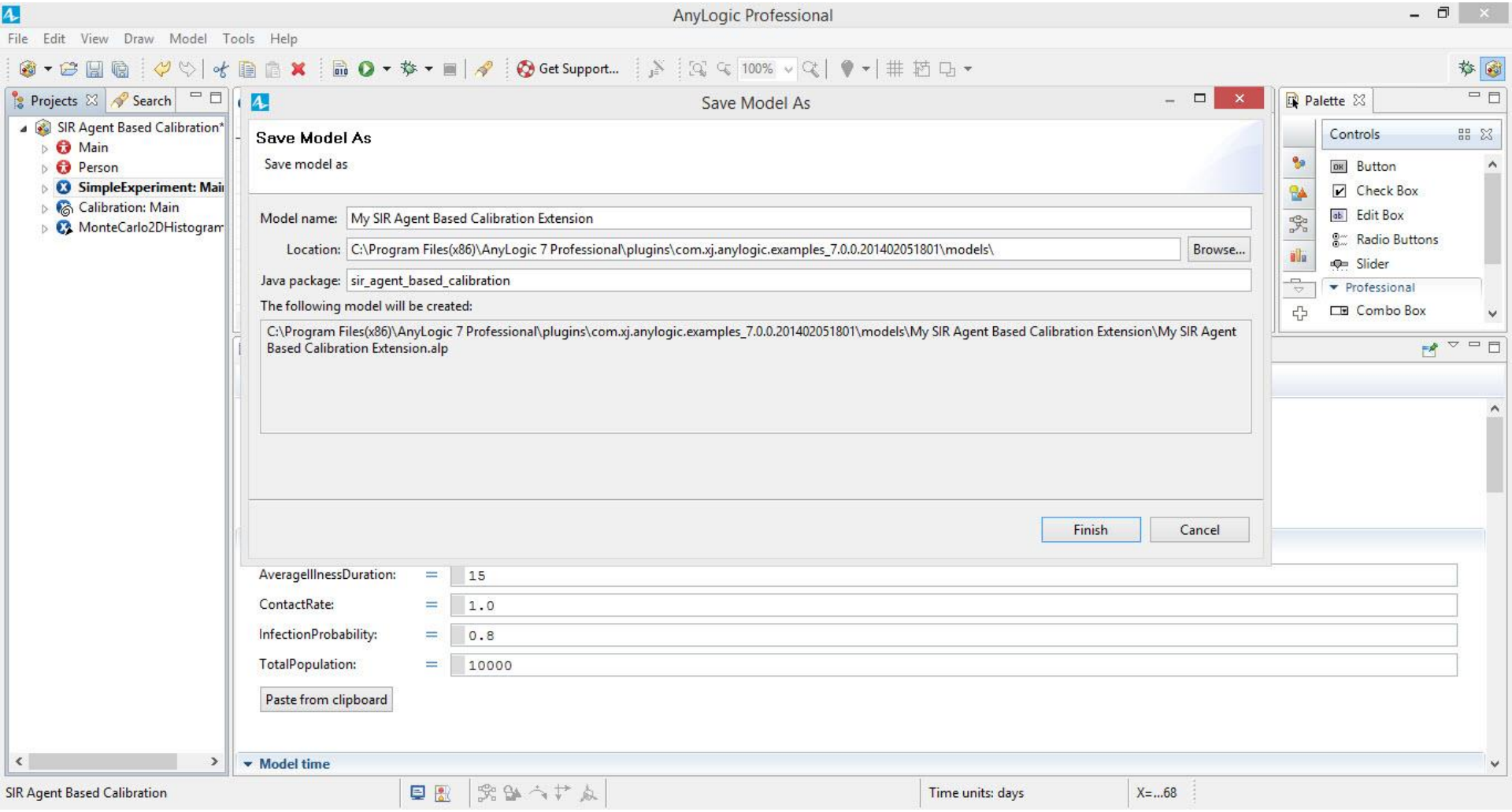
Add an Experiment



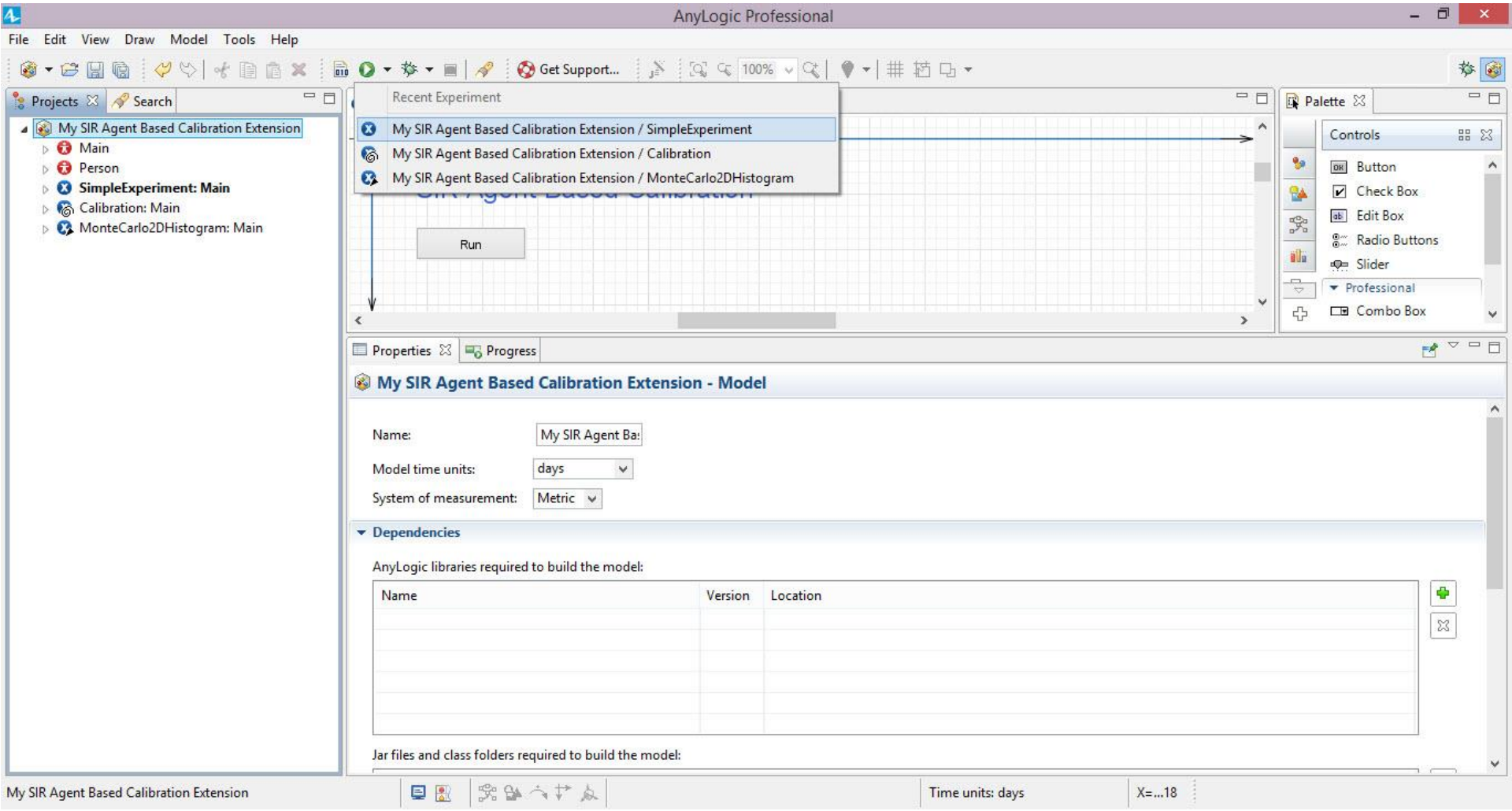
Add an Experiment



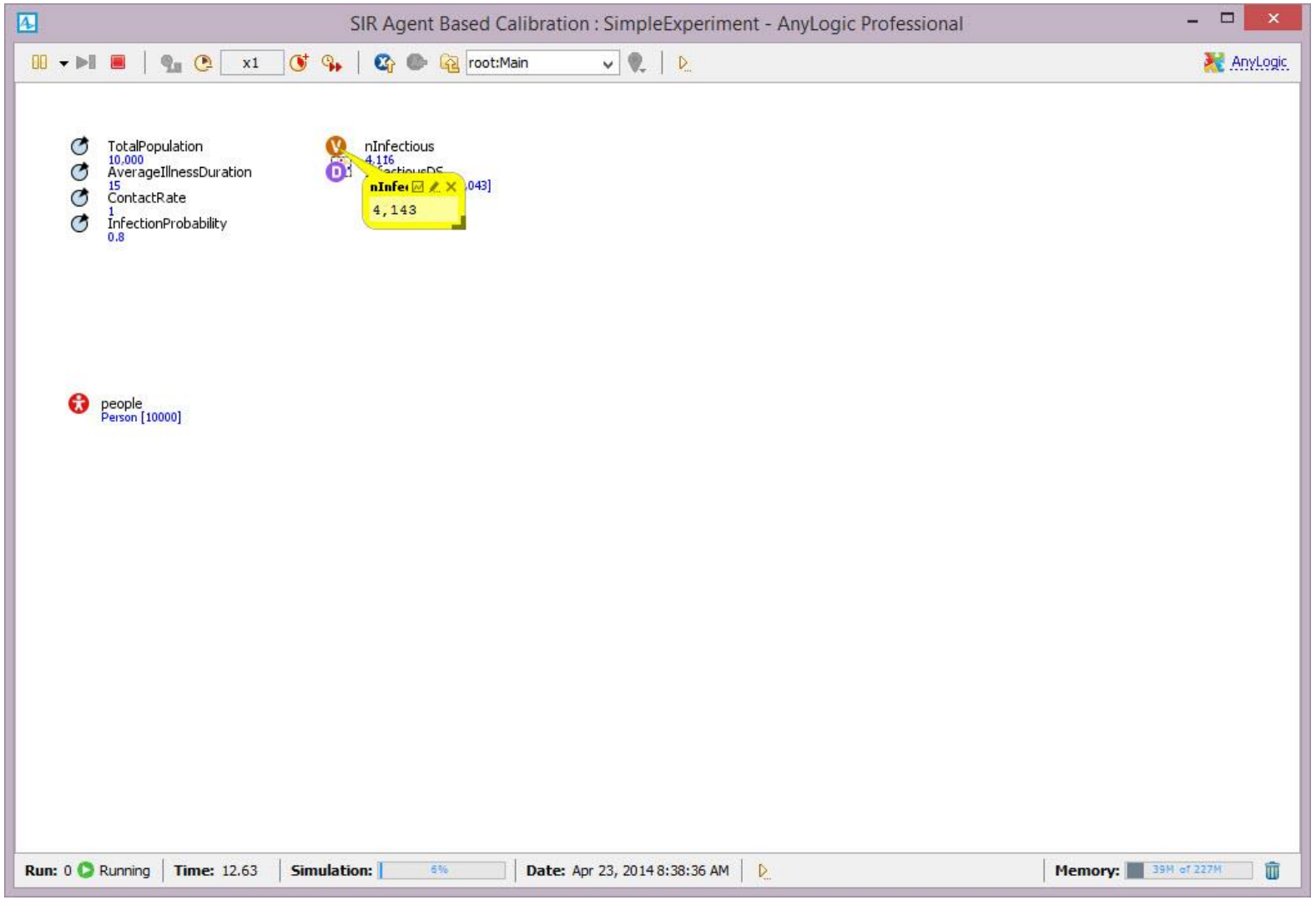
Save the Resulting Model (To Avoid Overwriting the Other Model)



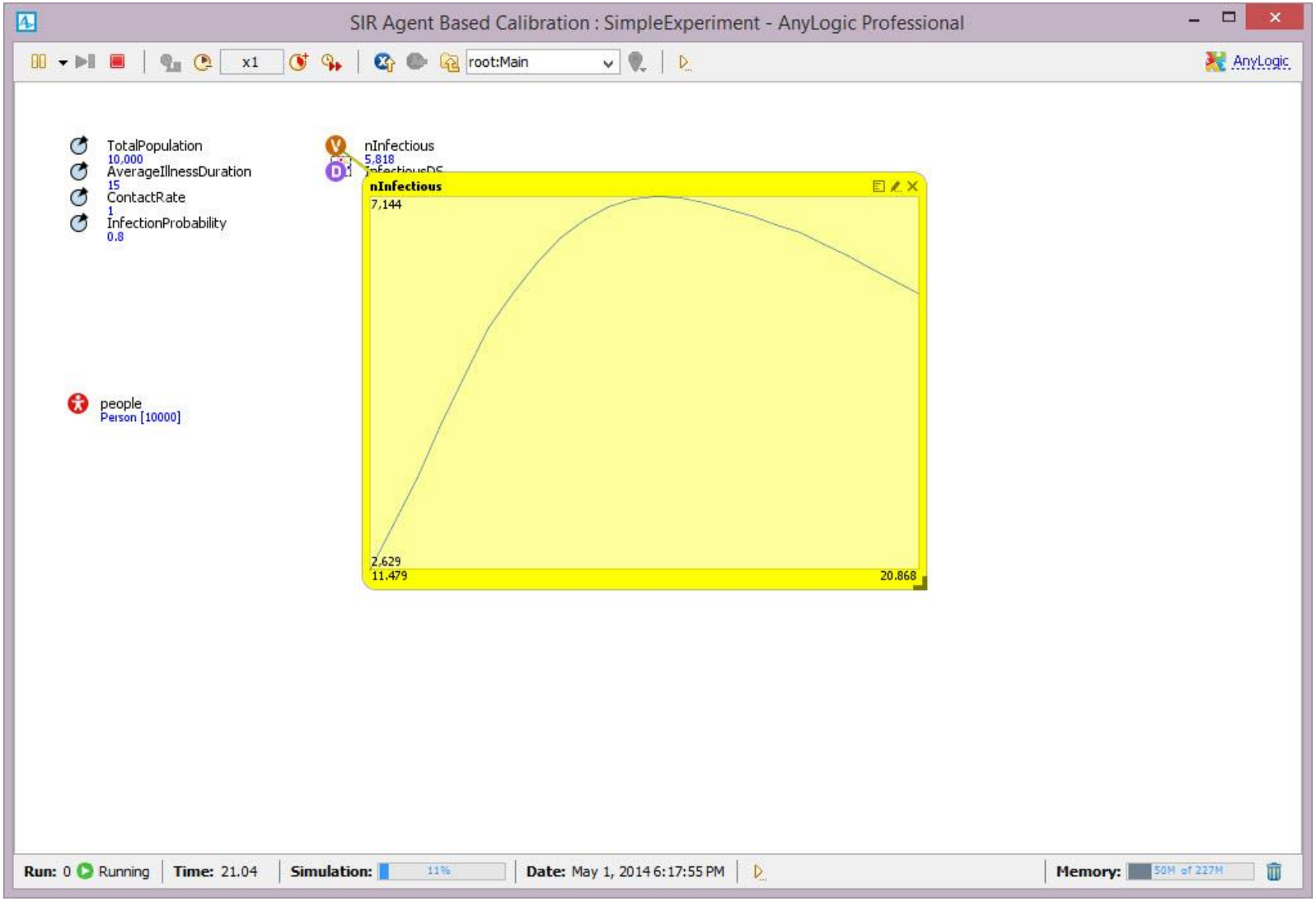
Run the Experiment (To Verify Functionality)



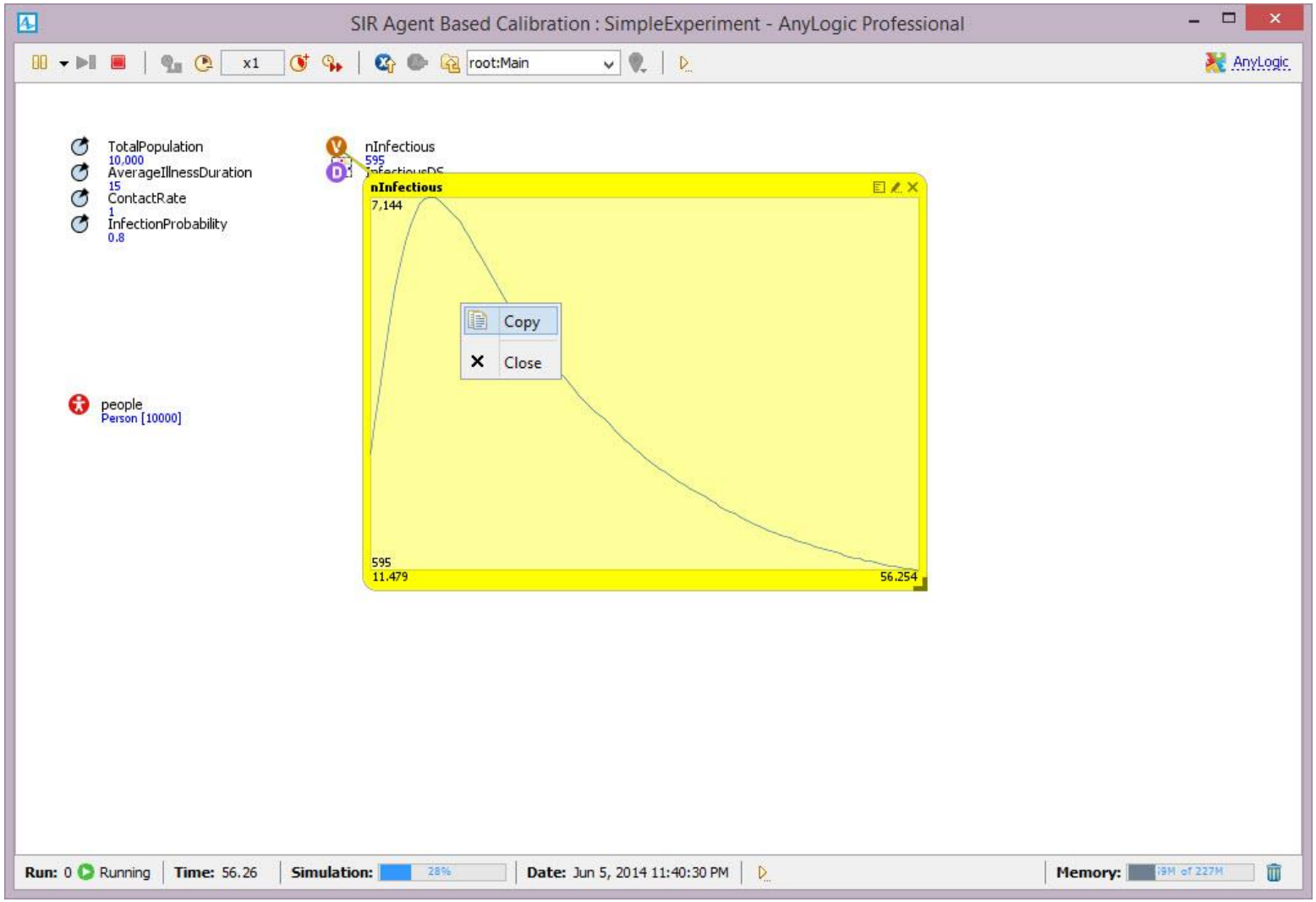
Click on Variable “nInfectious”



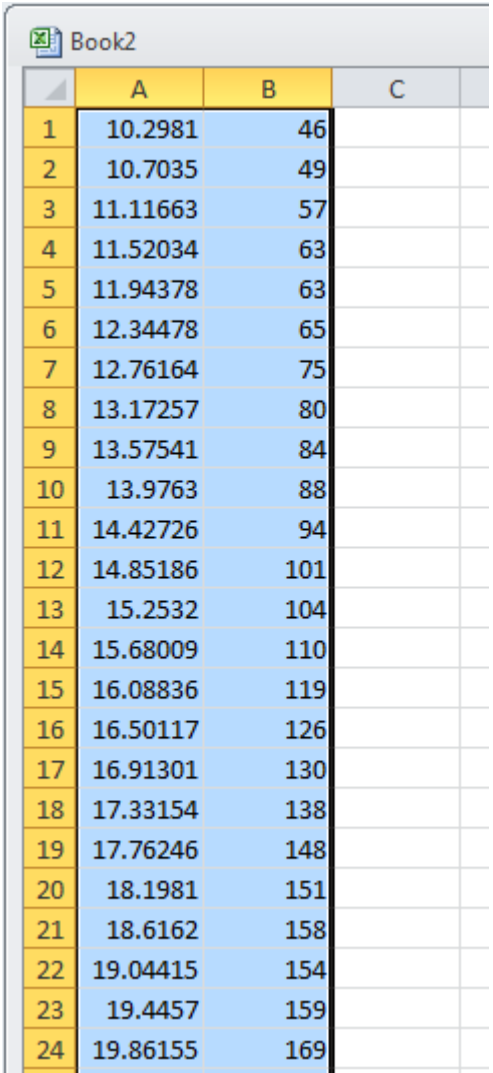
Graph of Variable



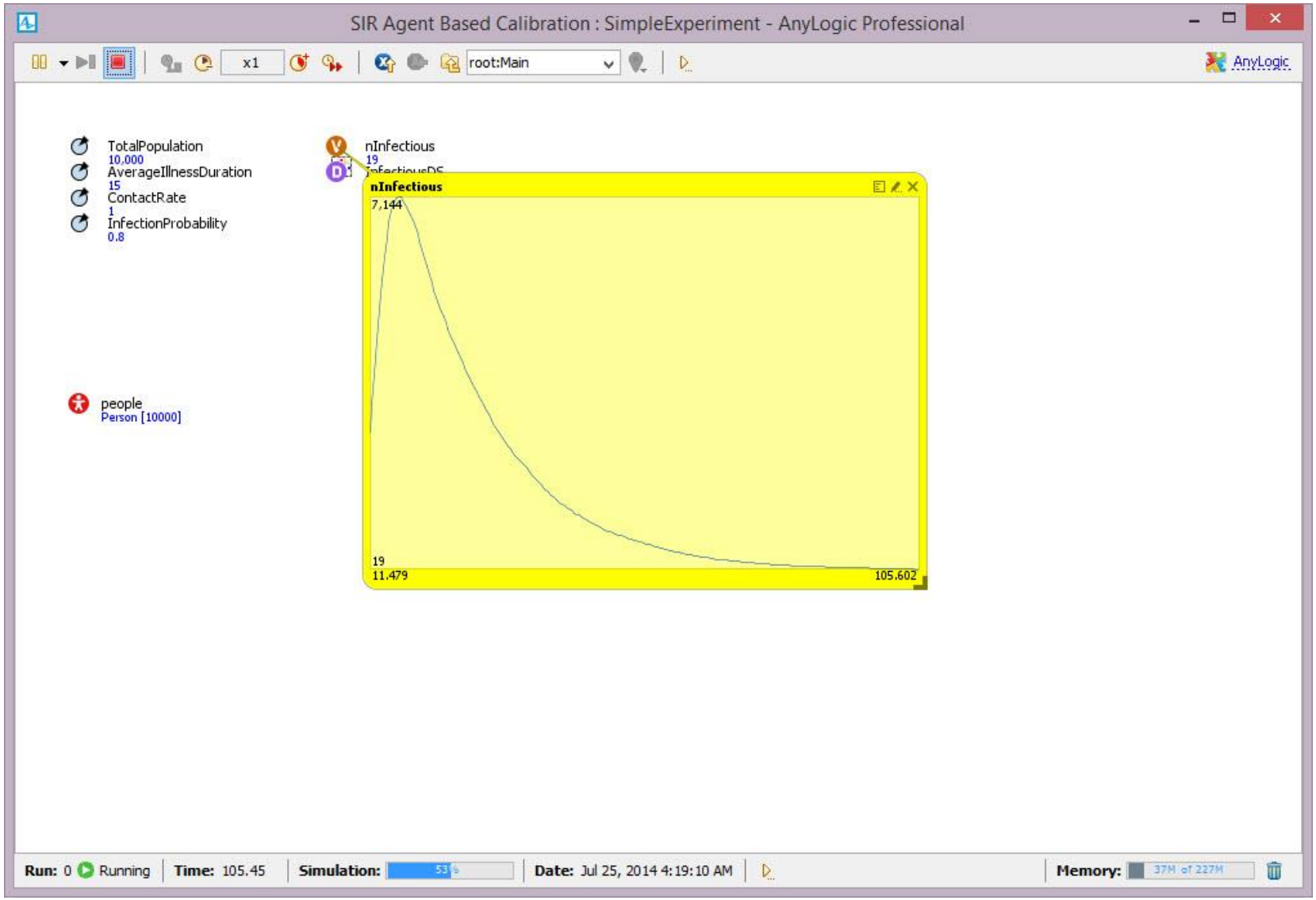
Right-Click to Copy the Numeric Data



Pasting Into Excel



Press Red “Stop” Button to Terminate Execution



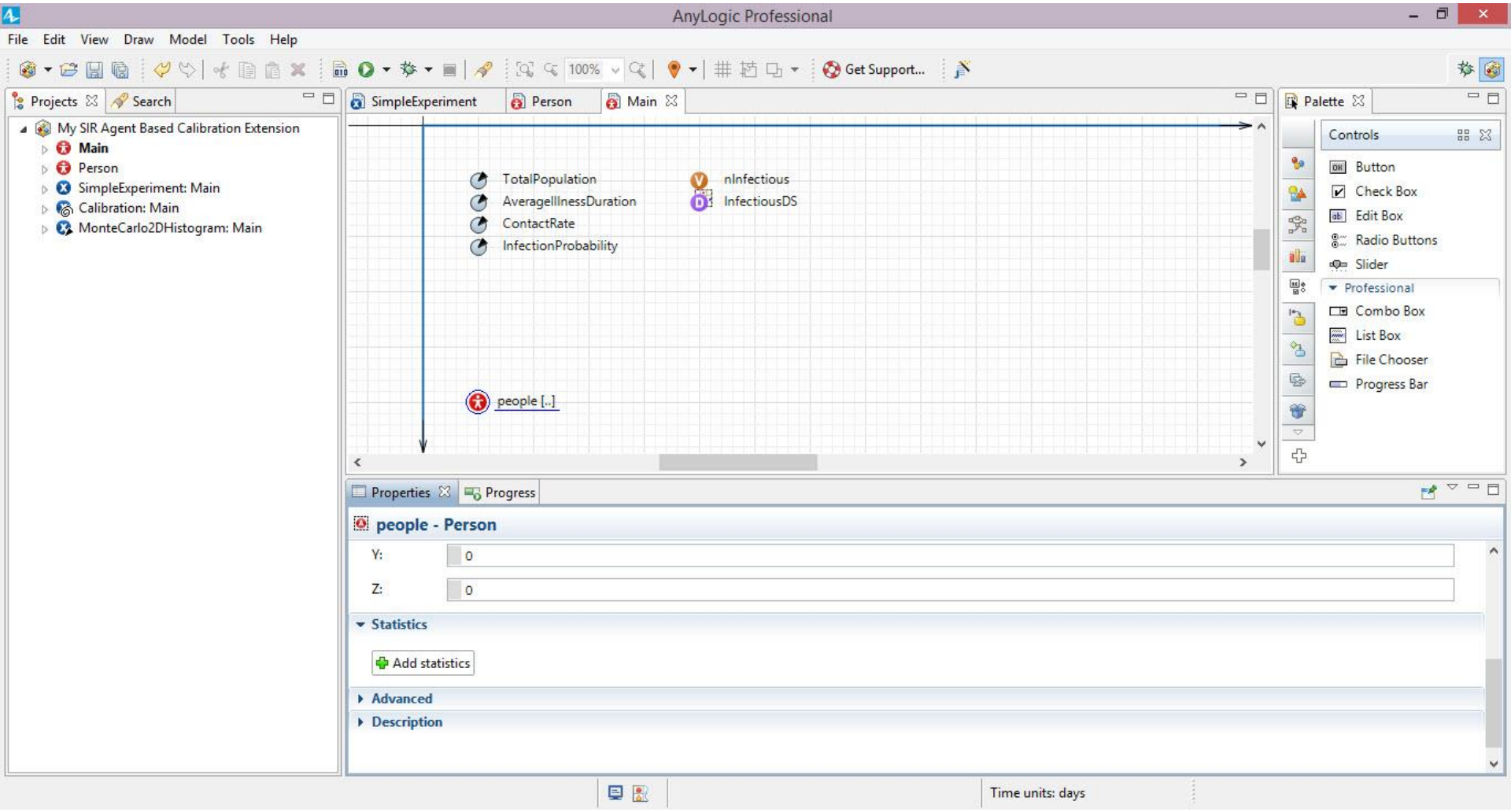
Techniques for Collecting & Outputting Data

- Ad-Hoc Exports from variables
- Pre-Prepared methods
  - Statistics
  - Charts
  - Manual copies from visible datasets
  - Export to files
  - Writing to console
  - Export to databases
  - [AnyLogic Professional] Dataset archiving
  - Capturing images of graphs

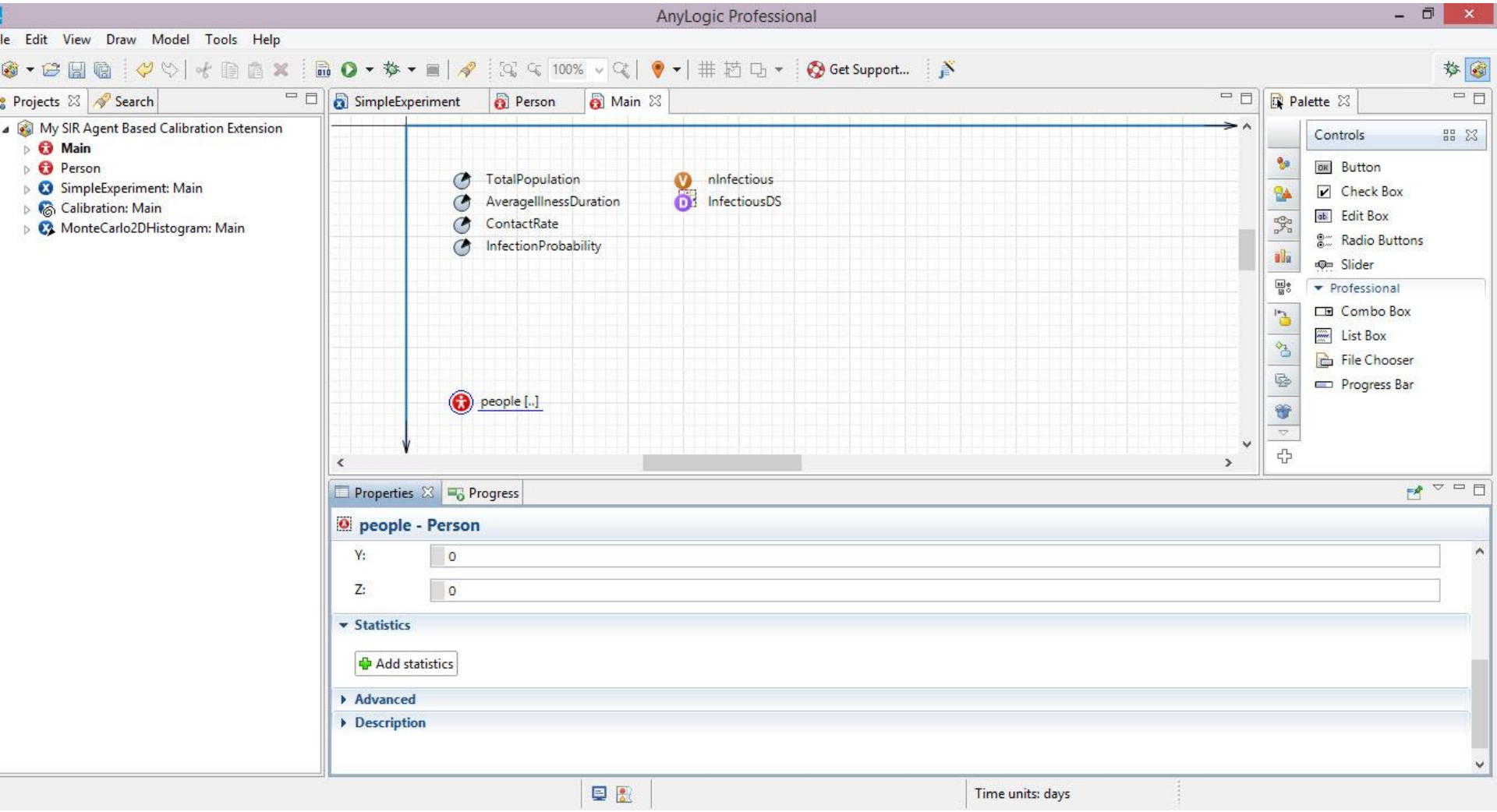
Statistics & Charts

- A population of agents can have associated statistics that calculate values
- Examples of things that can be computed with using AnyLogic's statistics
  - Count of agents in the population for which certain condition ("predicate") evaluates to true
  - Function of the values of some expression over the population
    - Maximum value
    - Minimum value
    - Average value
    - Sum (total) over population
  - Statistics can be defined as properties of the population

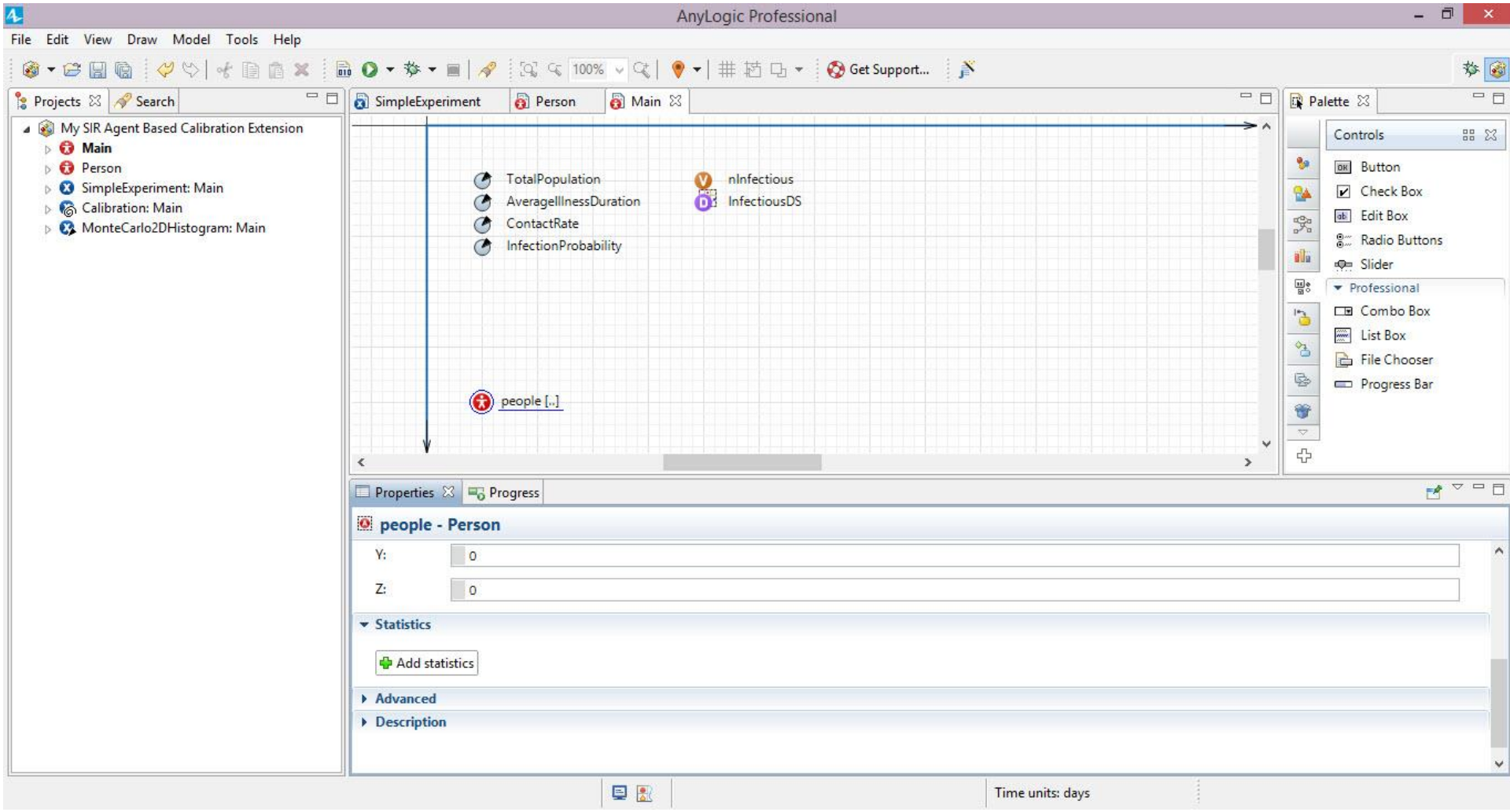
Select “People”, and Choose “Statistics”



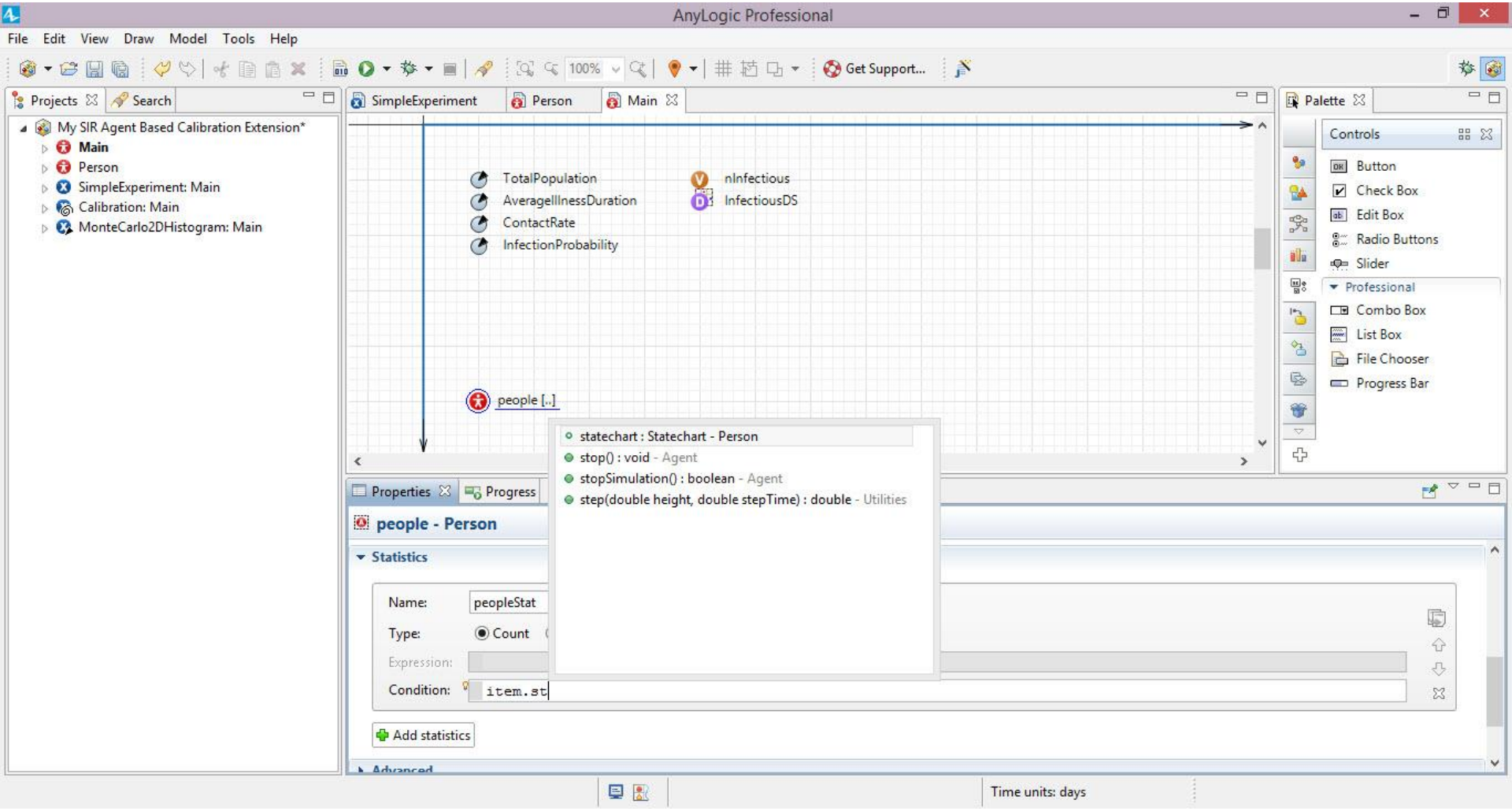
Also, Expand “Statechart” Under “Person”



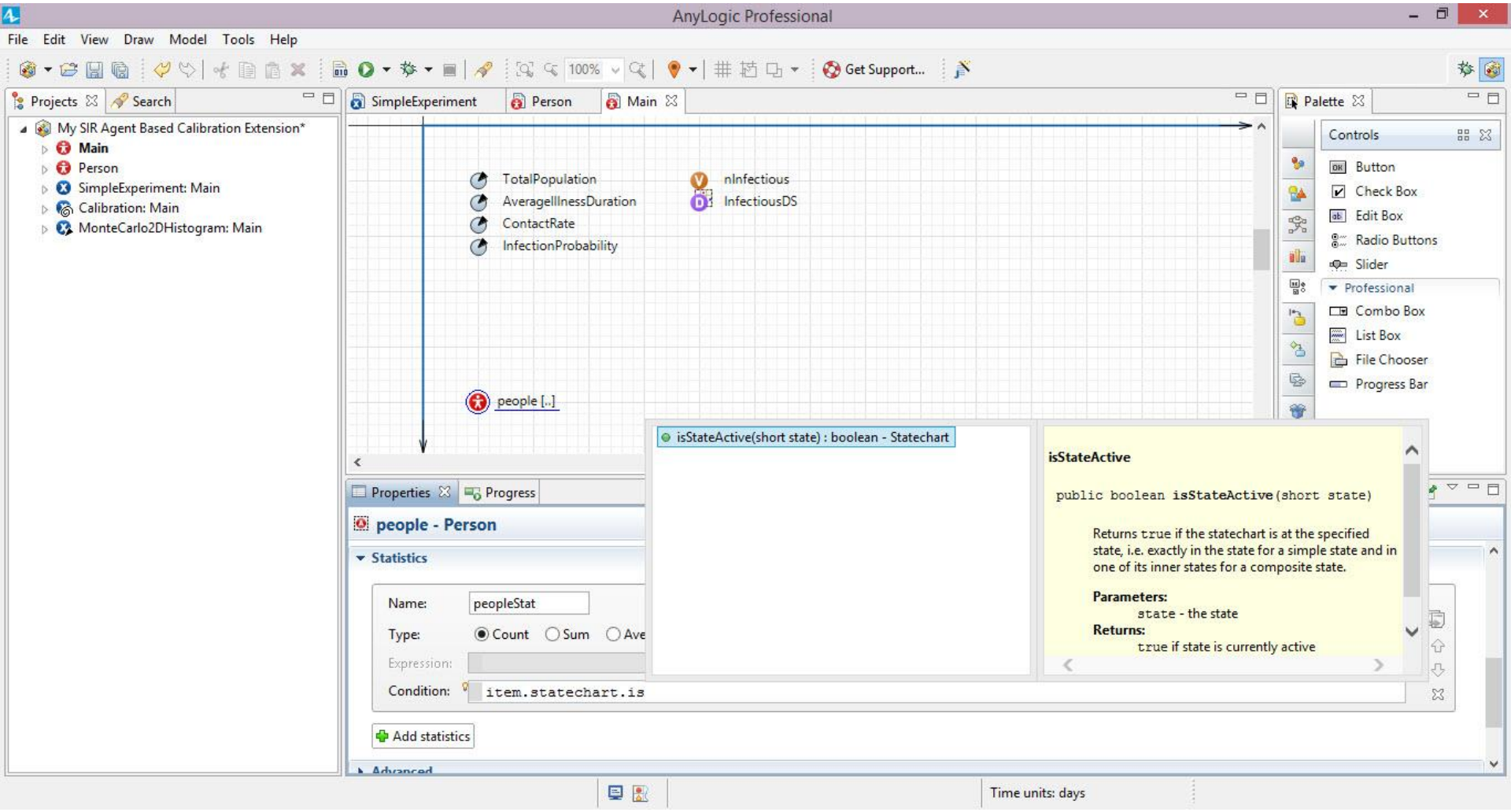
Click “Add Statistics”



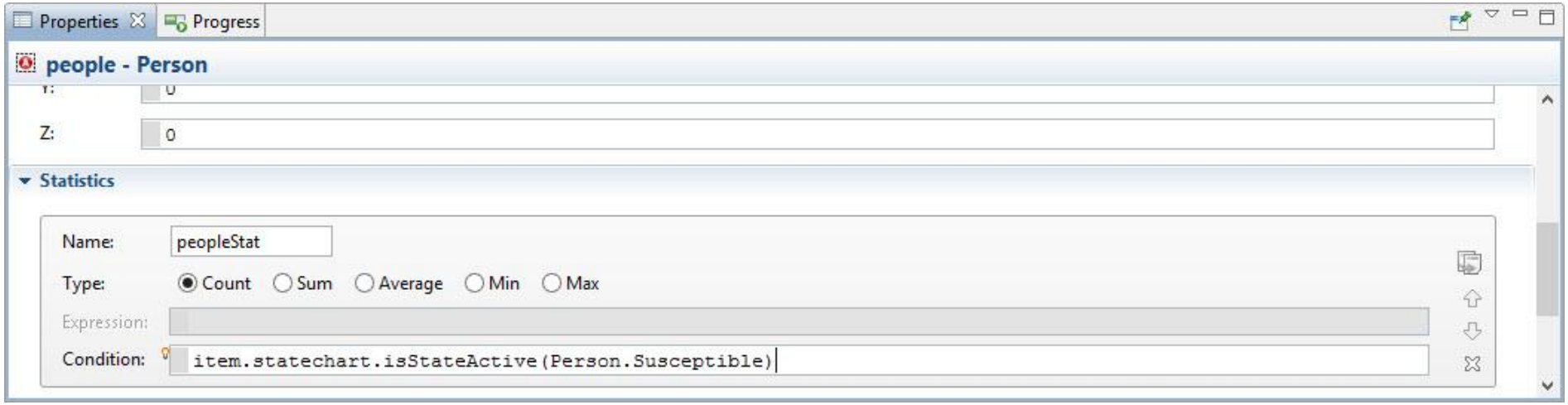
Fill in the “Condition” (Predicate) on Person



Continue Typing



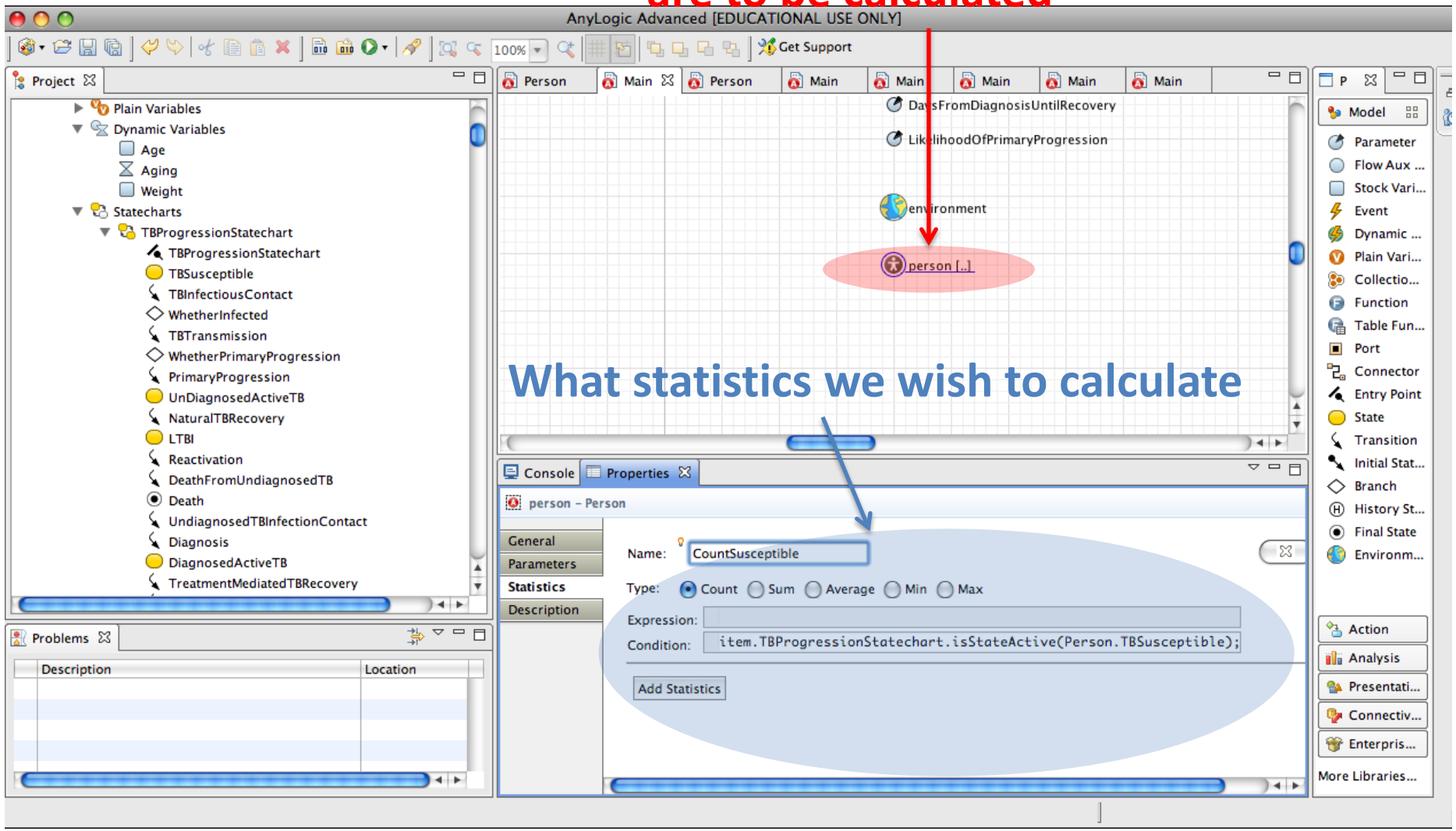
Full Expression



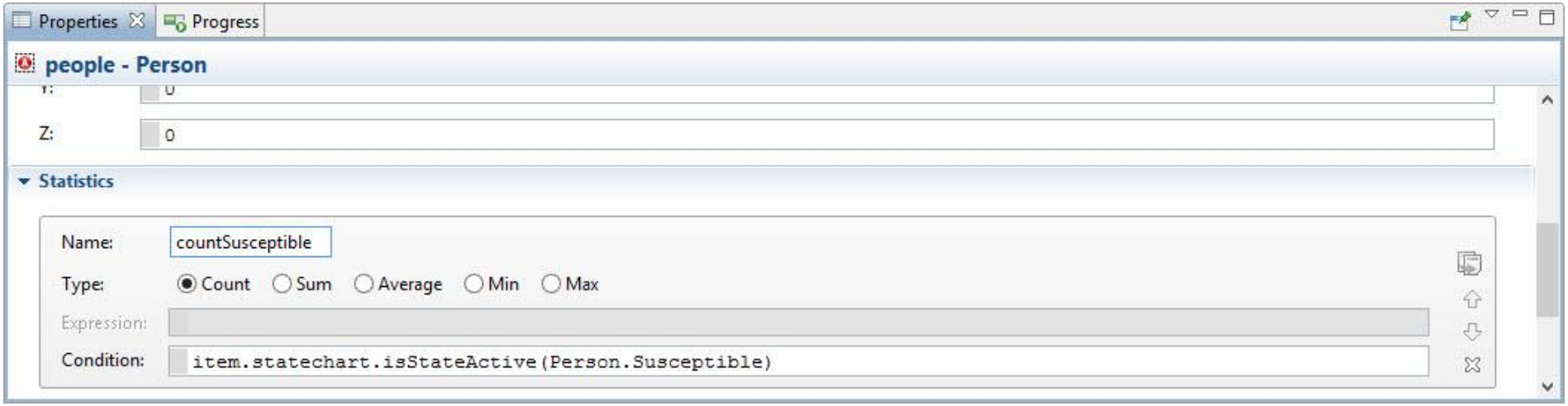
Expression: item.statechart.isStateActive(Person.Susceptible)

Example Statistics

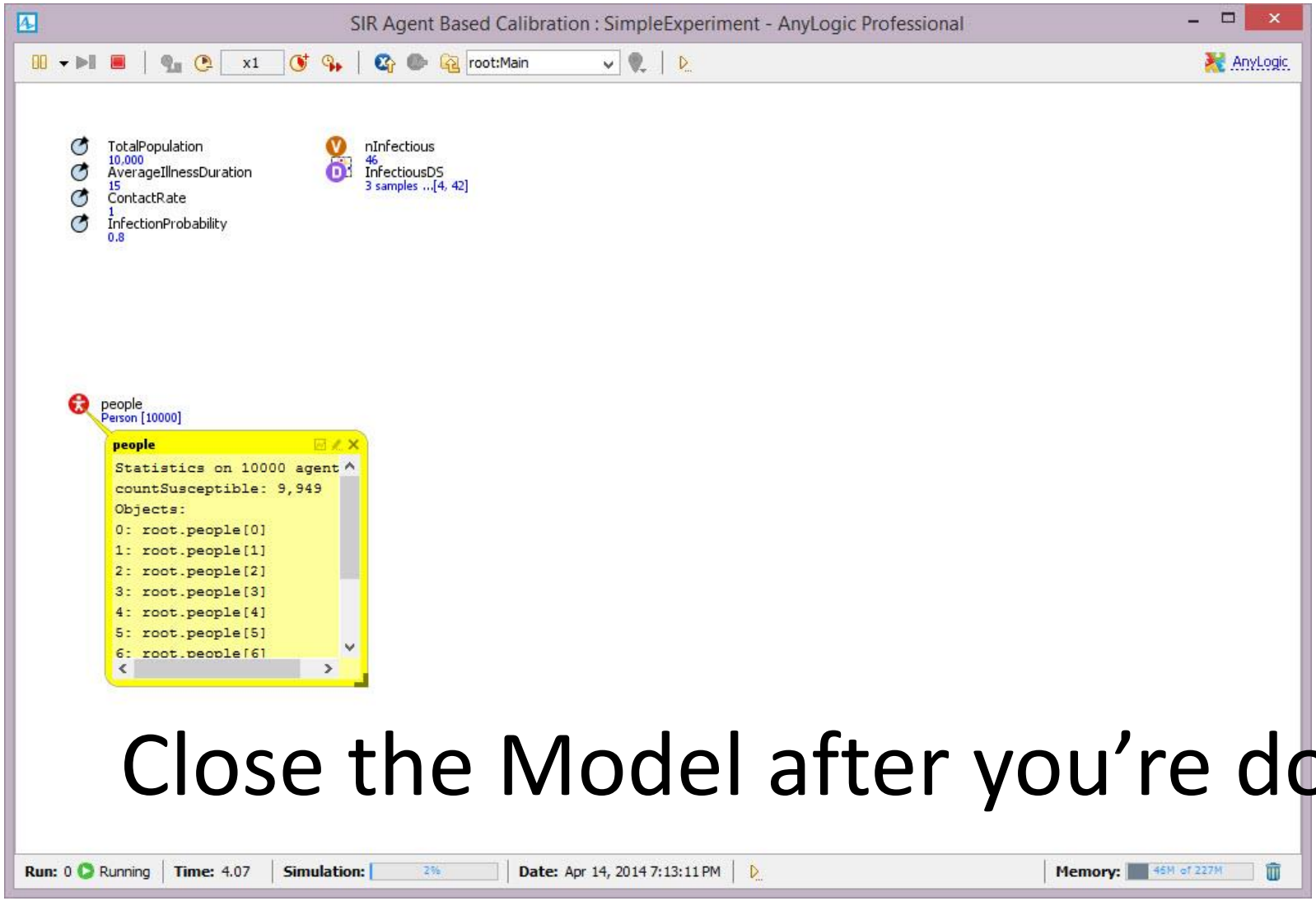
The population in which the statistics are to be calculated



Name the Statistic “countSusceptible”

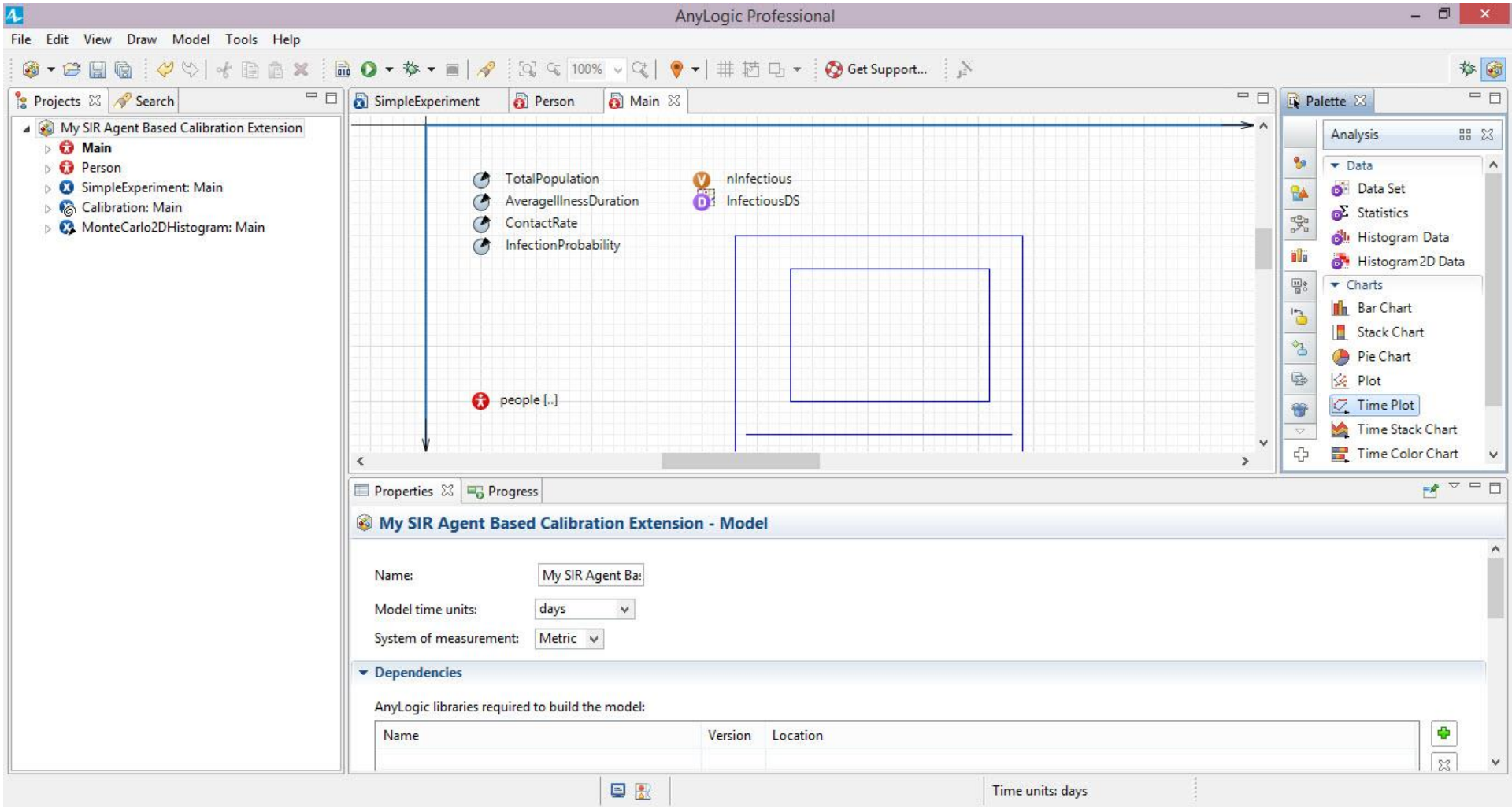


Run the Model, and Click on “people”
The Statistic should be Visible

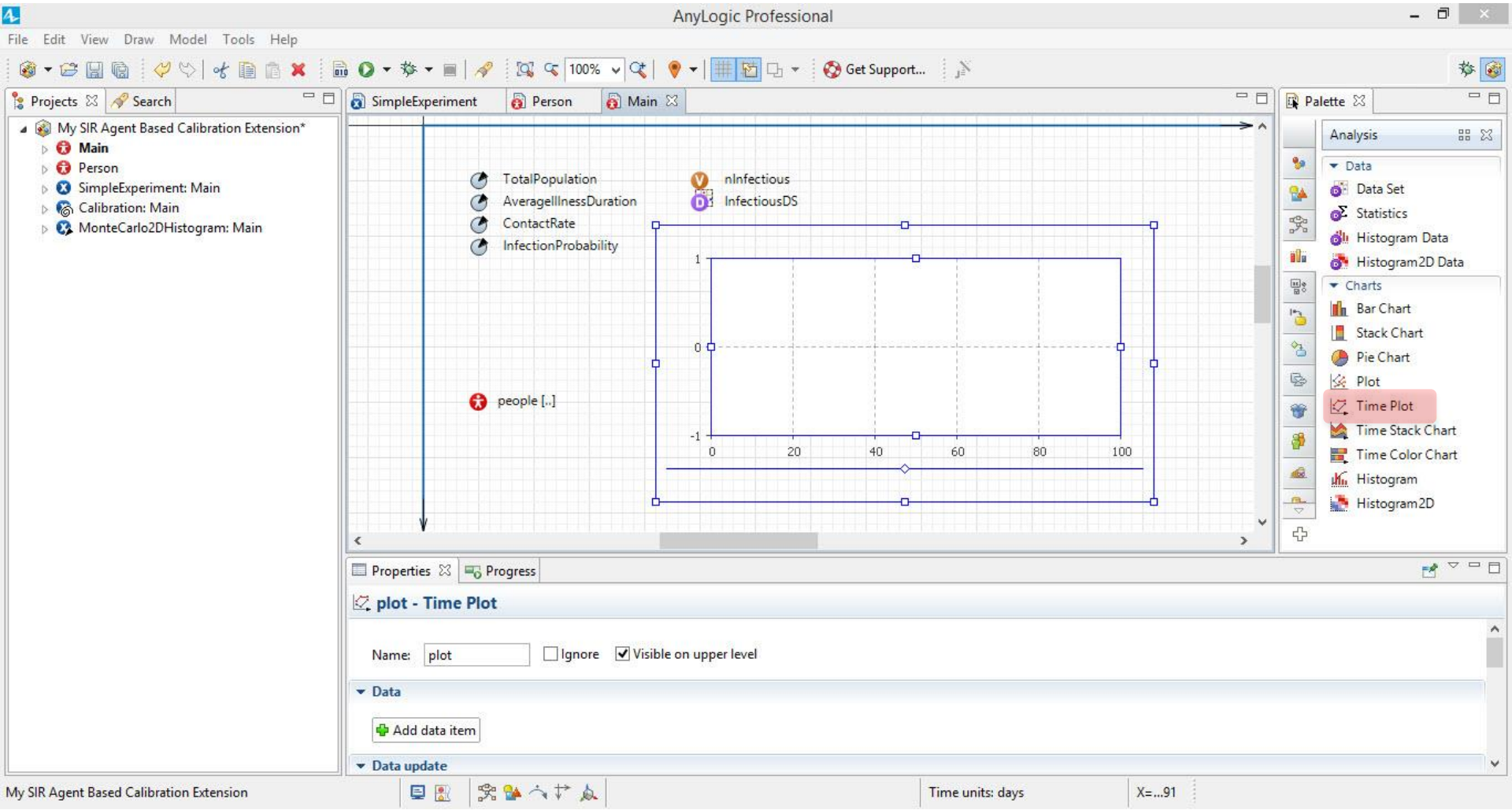


Close the Model after you’re done

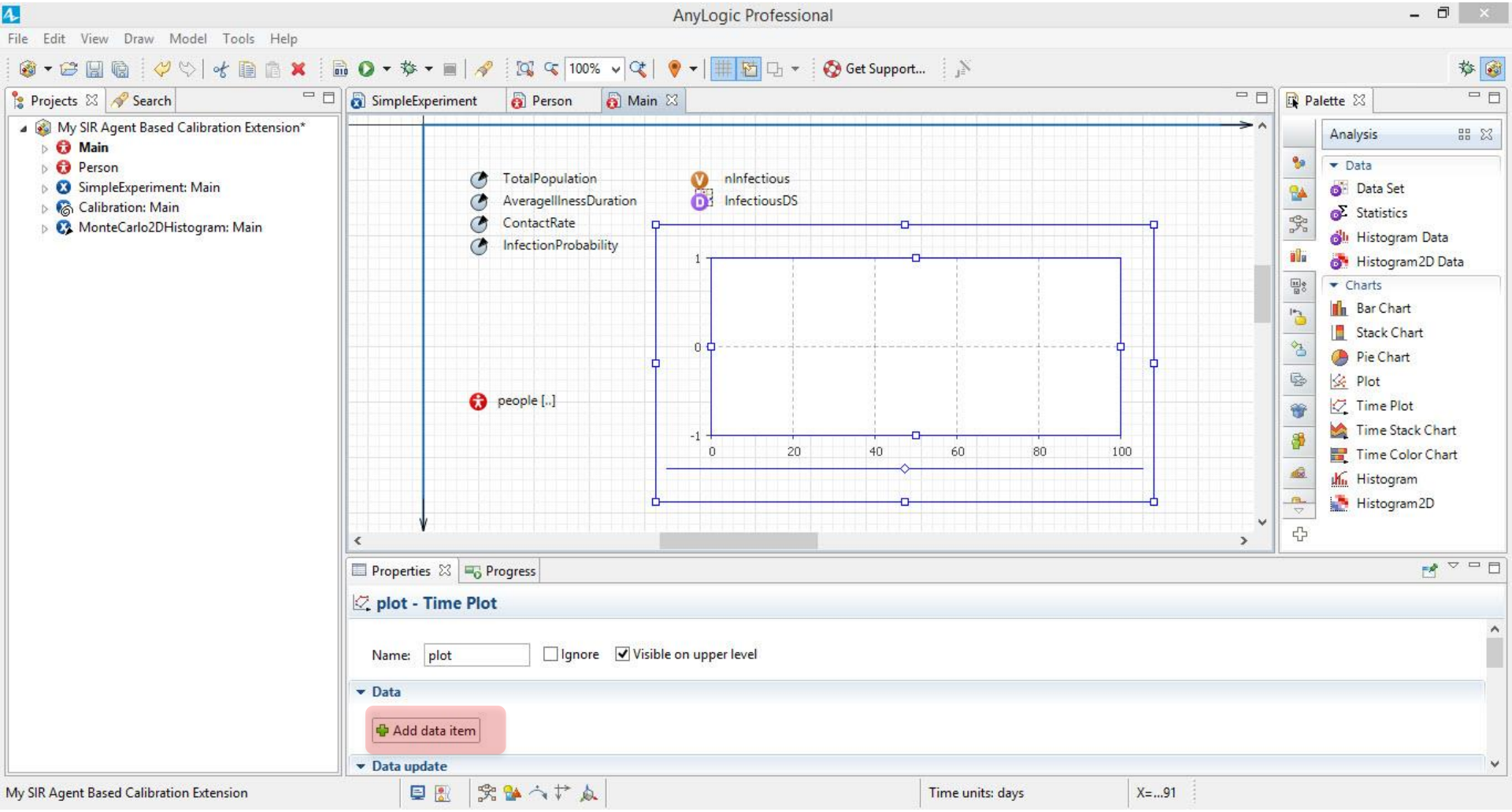
Drag a “Time Plot” from the Palette to the “Main” Canvas



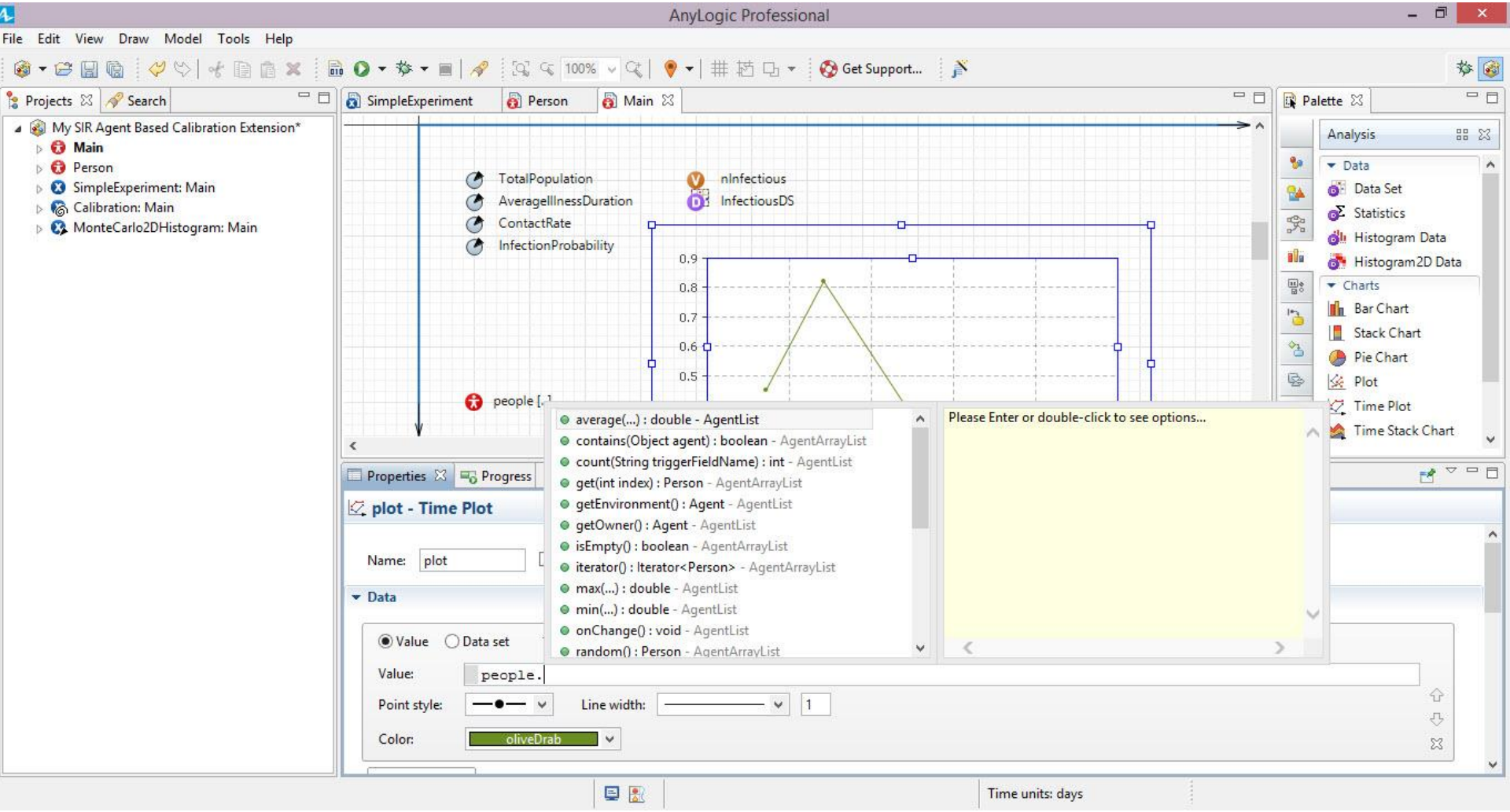
Enlarge the Chart



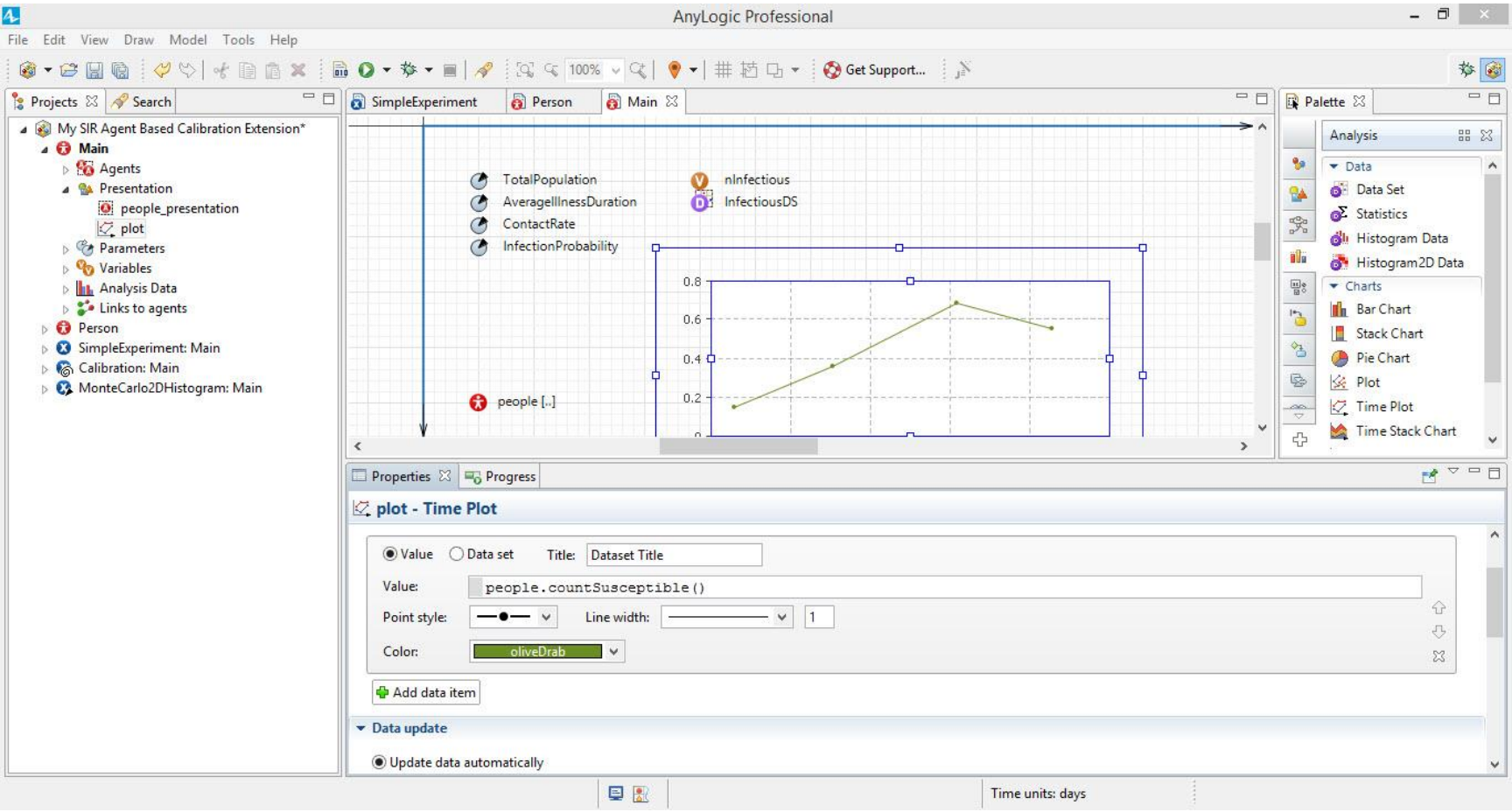
Click “Add Data Item”



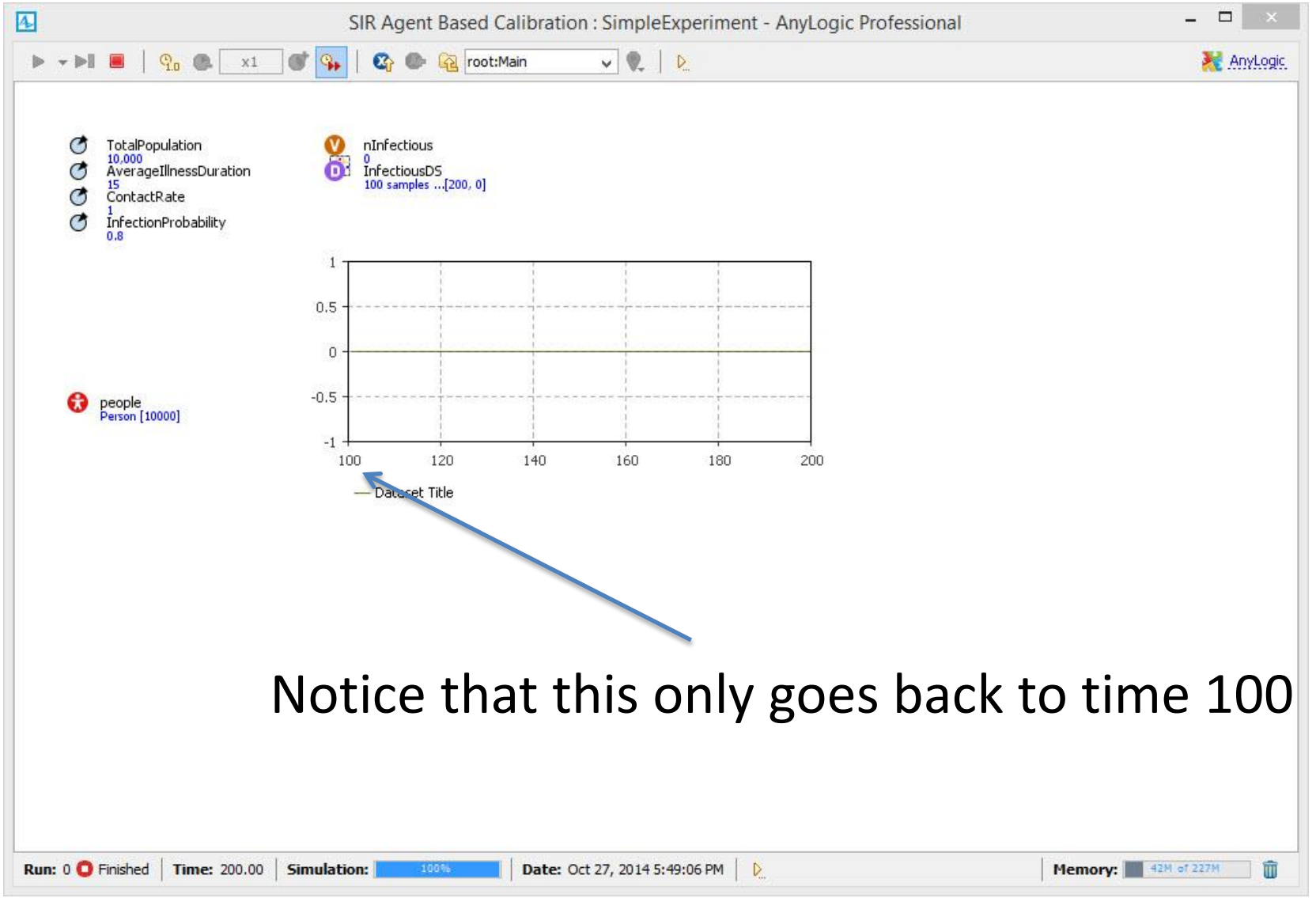
Put in “people.” and Press Ctrl-Space



Choose “Count Susceptible”

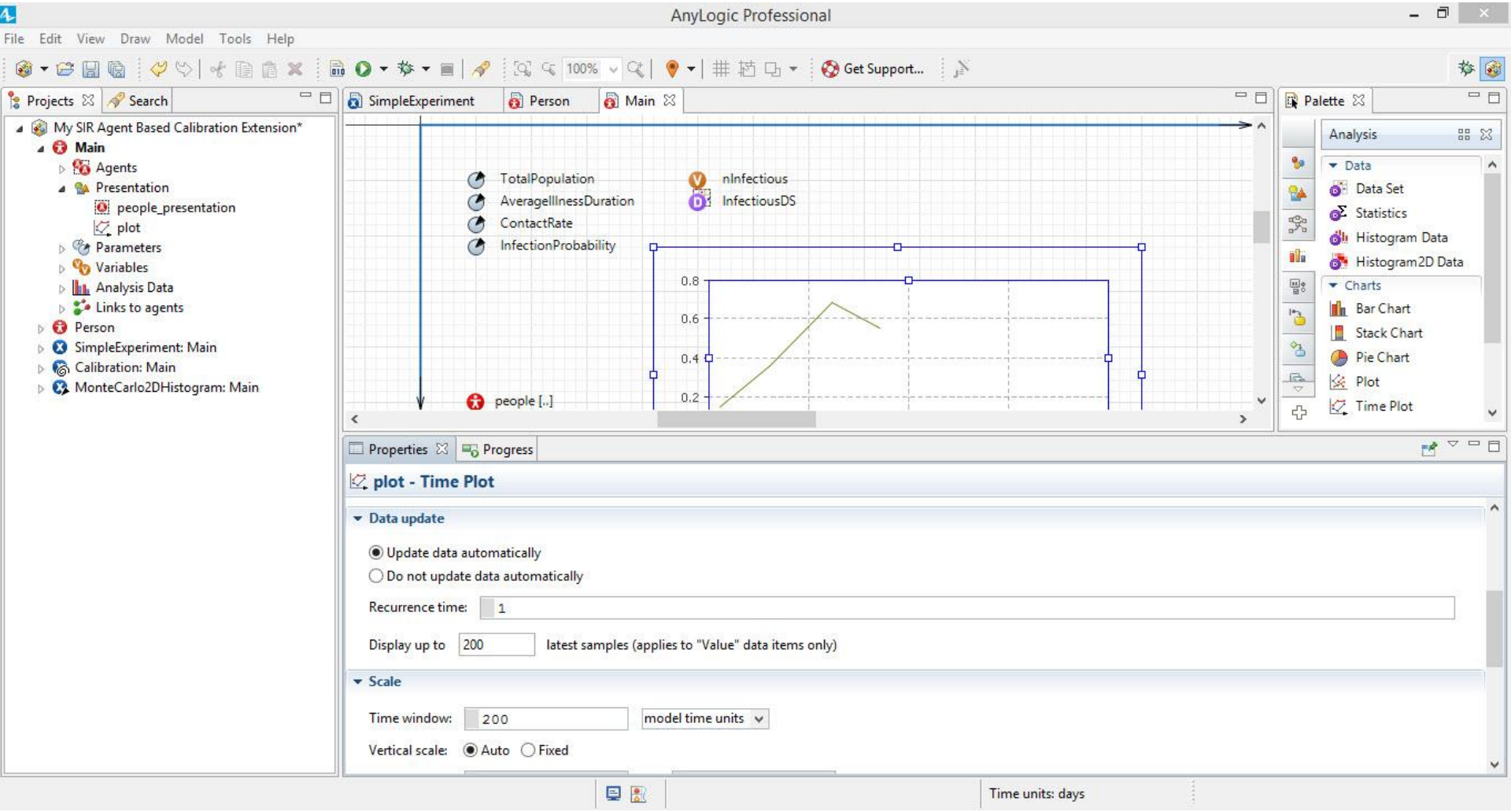


Now Run the Model

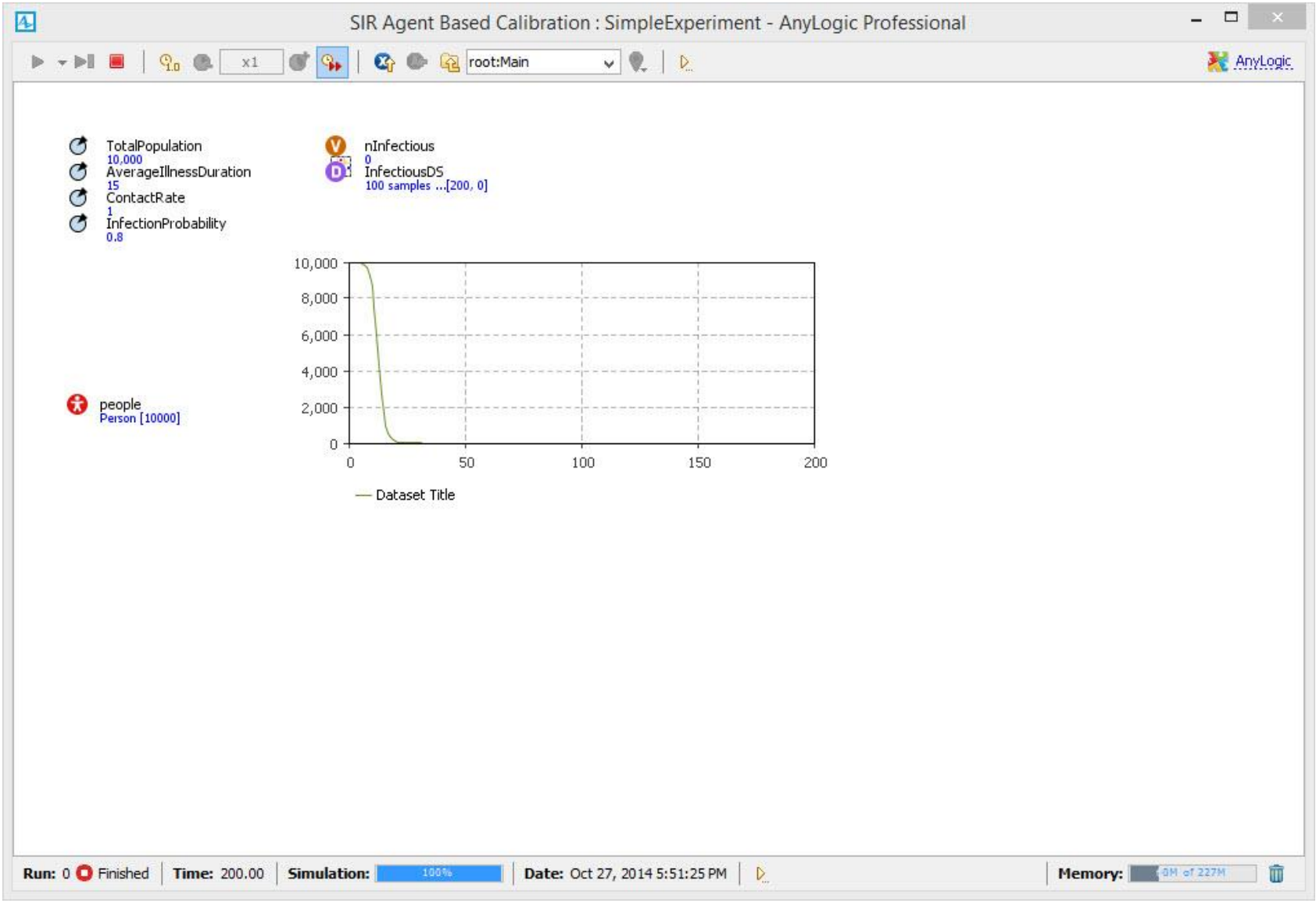


Notice that this only goes back to time 100

Stop the Simulation, and Click on the Plot. Change Time Window & Display Size to 200



This Captures the Full Time Range



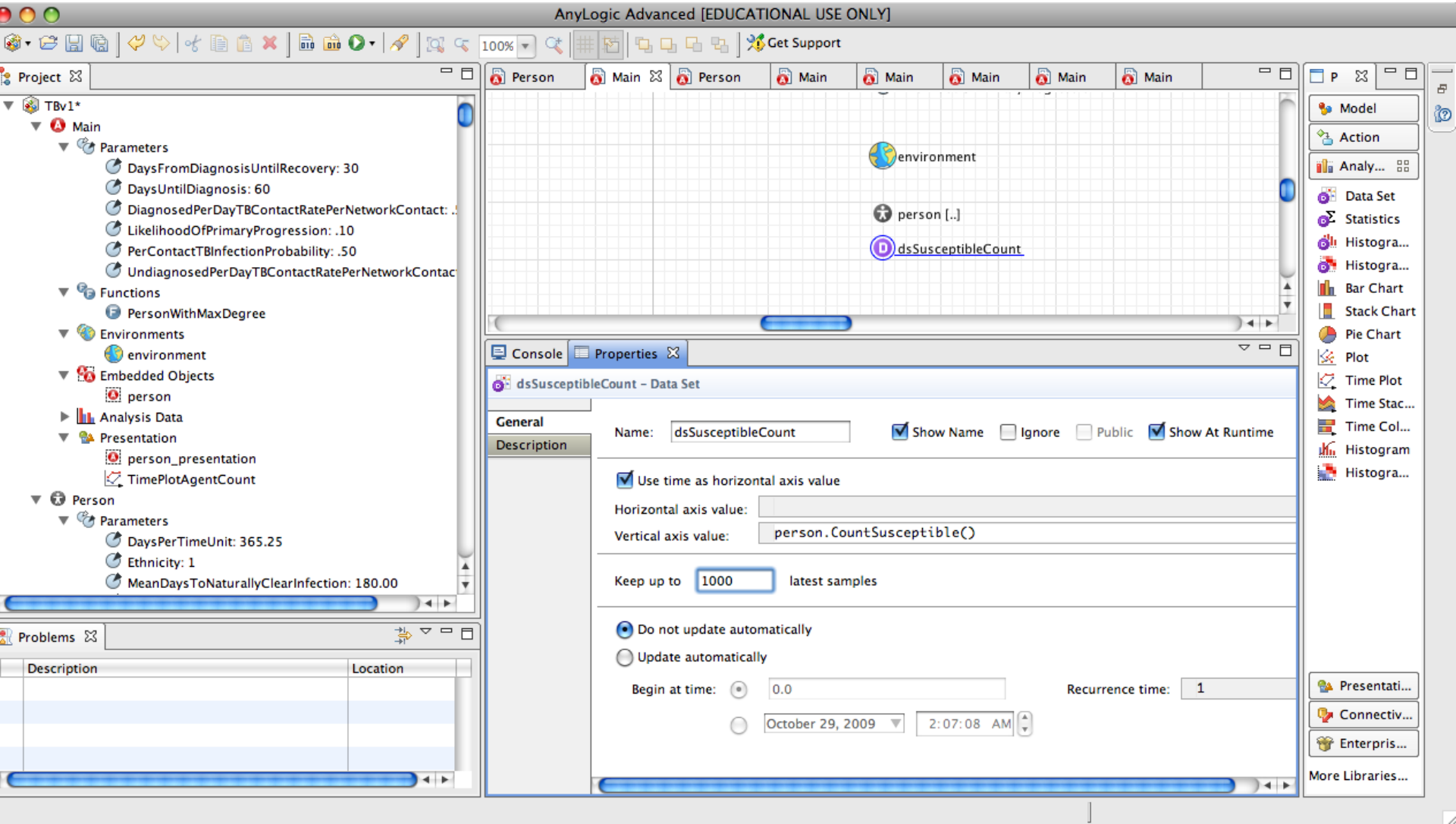
Techniques for Collecting & Outputting Data

- Ad-Hoc Exports from variables
- Pre-Prepared methods
  - Statistics
  - Charts
  - Manual copies from visible datasets
  - Export to files
  - Writing to console
  - Export to databases
  - [AnyLogic Professional] Dataset archiving
  - Capturing images of graphs

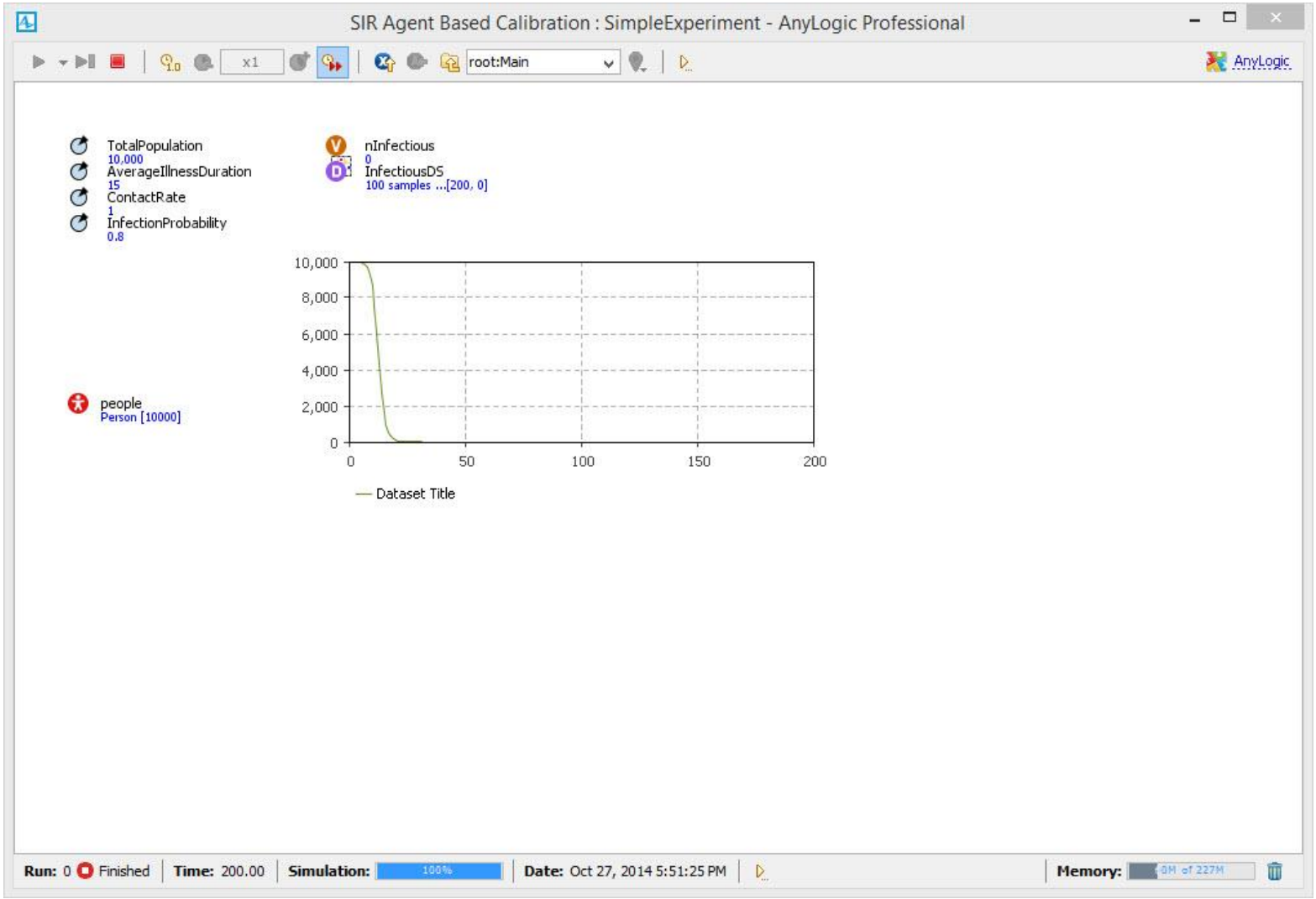
Datasets

- Datasets store recent values of some quantities from the model
- Datasets can be exported easily using custom code
  - This can simply call the dataset's toString method

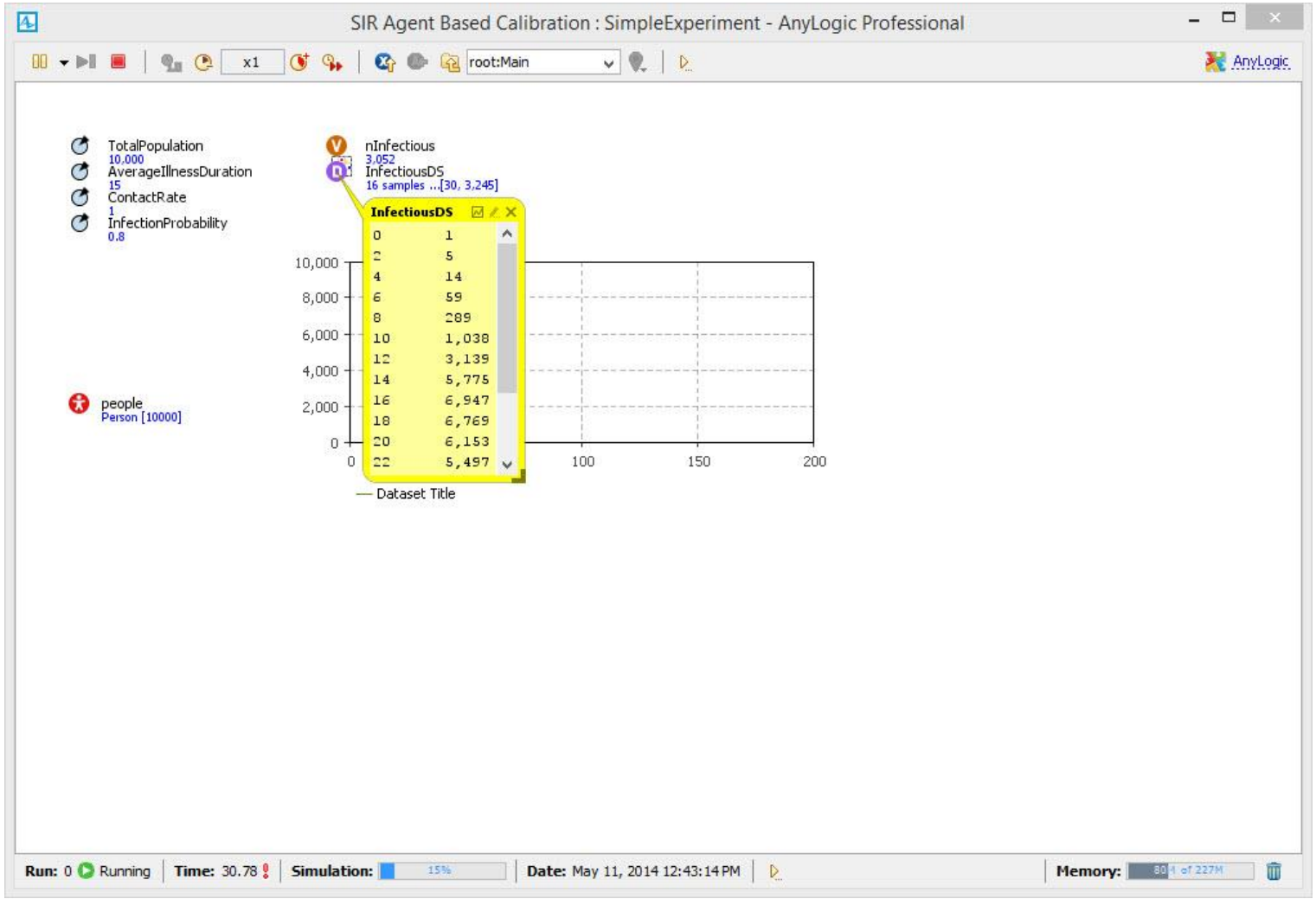
Output: Datasets



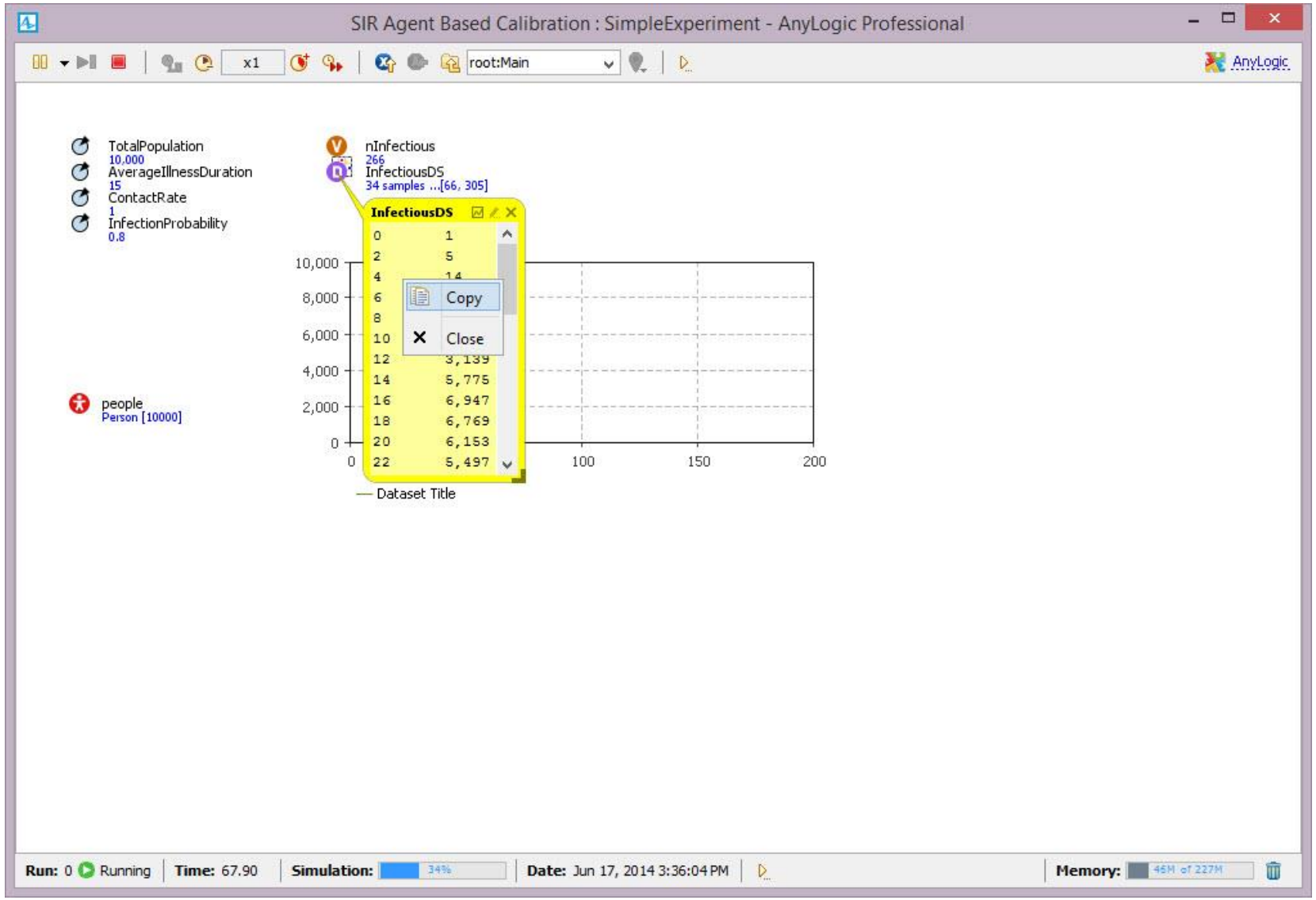
Run the Experiment & Click on "Infectious DS"



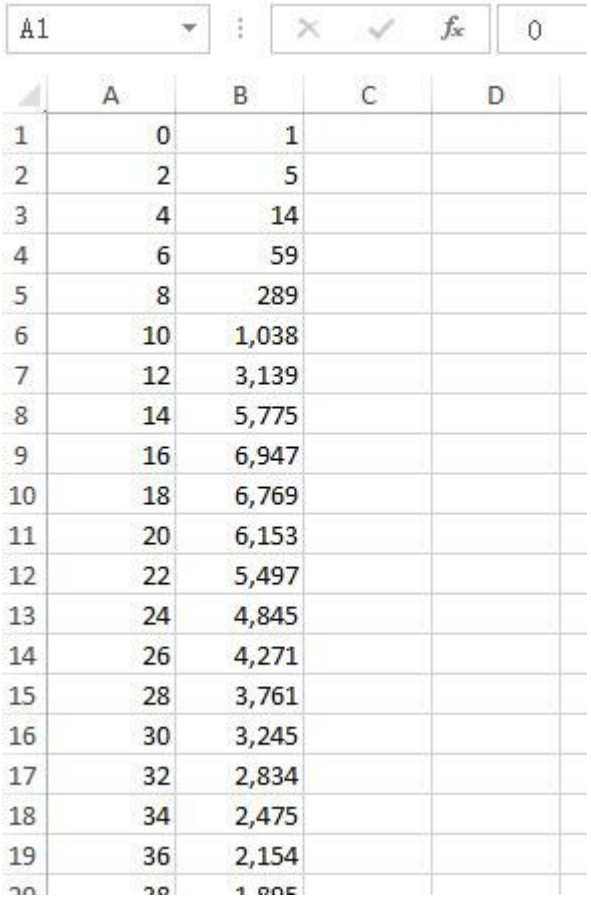
Click on “InfectiousDS” to See Data in Dataset



Right Click and Select “Copy”



Call Up Excel and Paste into It



	A	B	C	D
1	0	1		
2	2	5		
3	4	14		
4	6	59		
5	8	289		
6	10	1,038		
7	12	3,139		
8	14	5,775		
9	16	6,947		
10	18	6,769		
11	20	6,153		
12	22	5,497		
13	24	4,845		
14	26	4,271		
15	28	3,761		
16	30	3,245		
17	32	2,834		
18	34	2,475		
19	36	2,154		

Dataset Properties

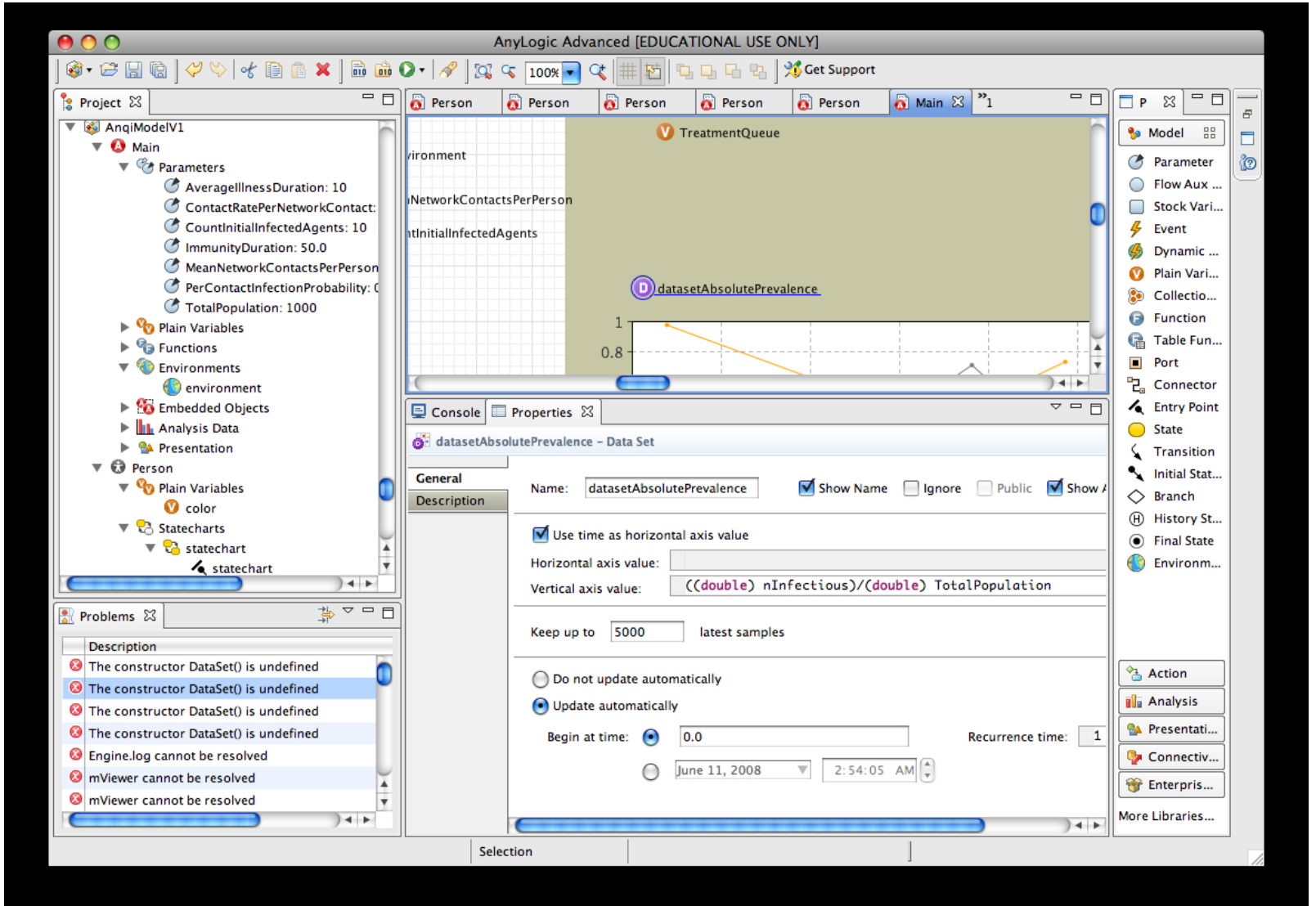
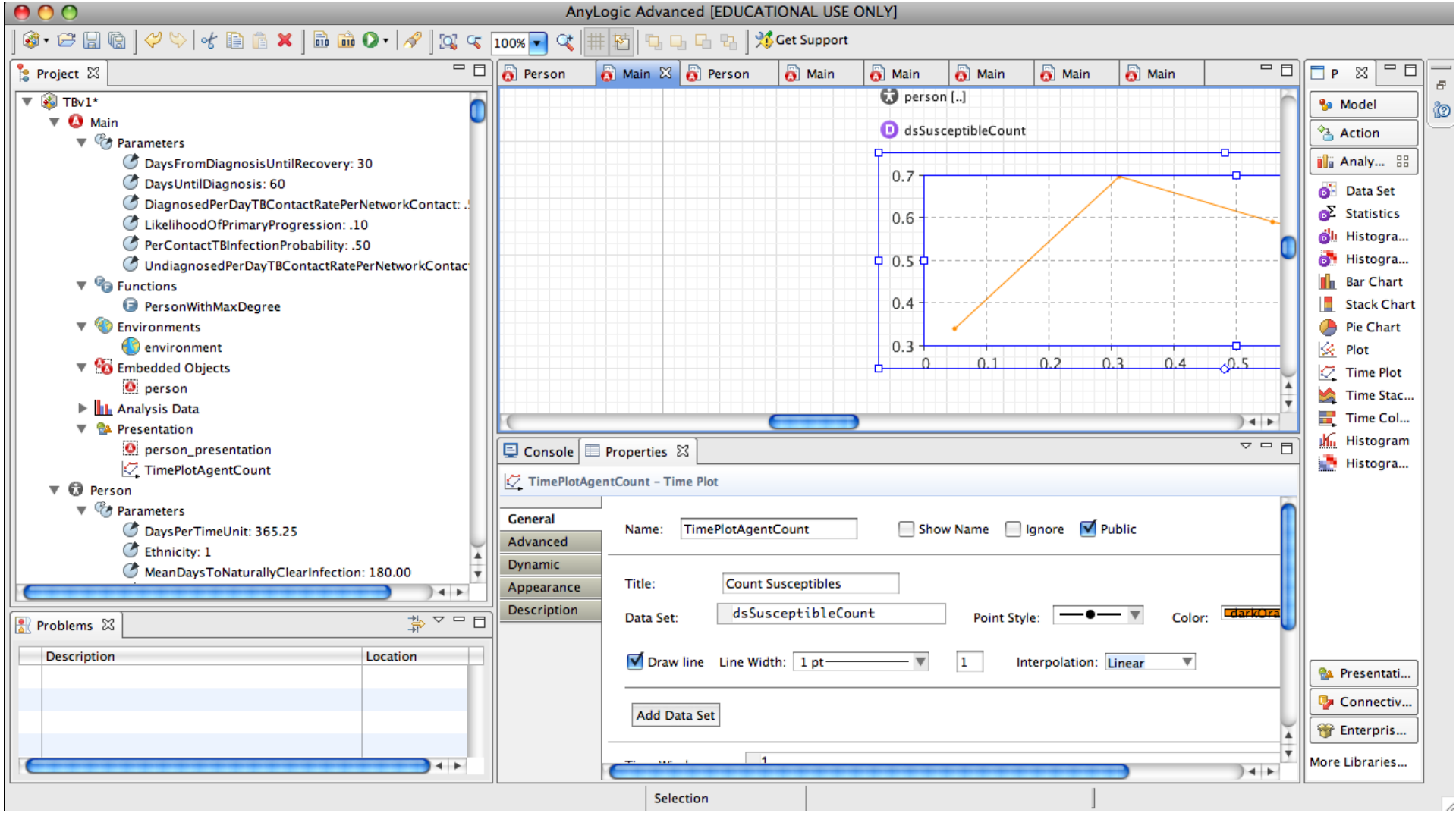
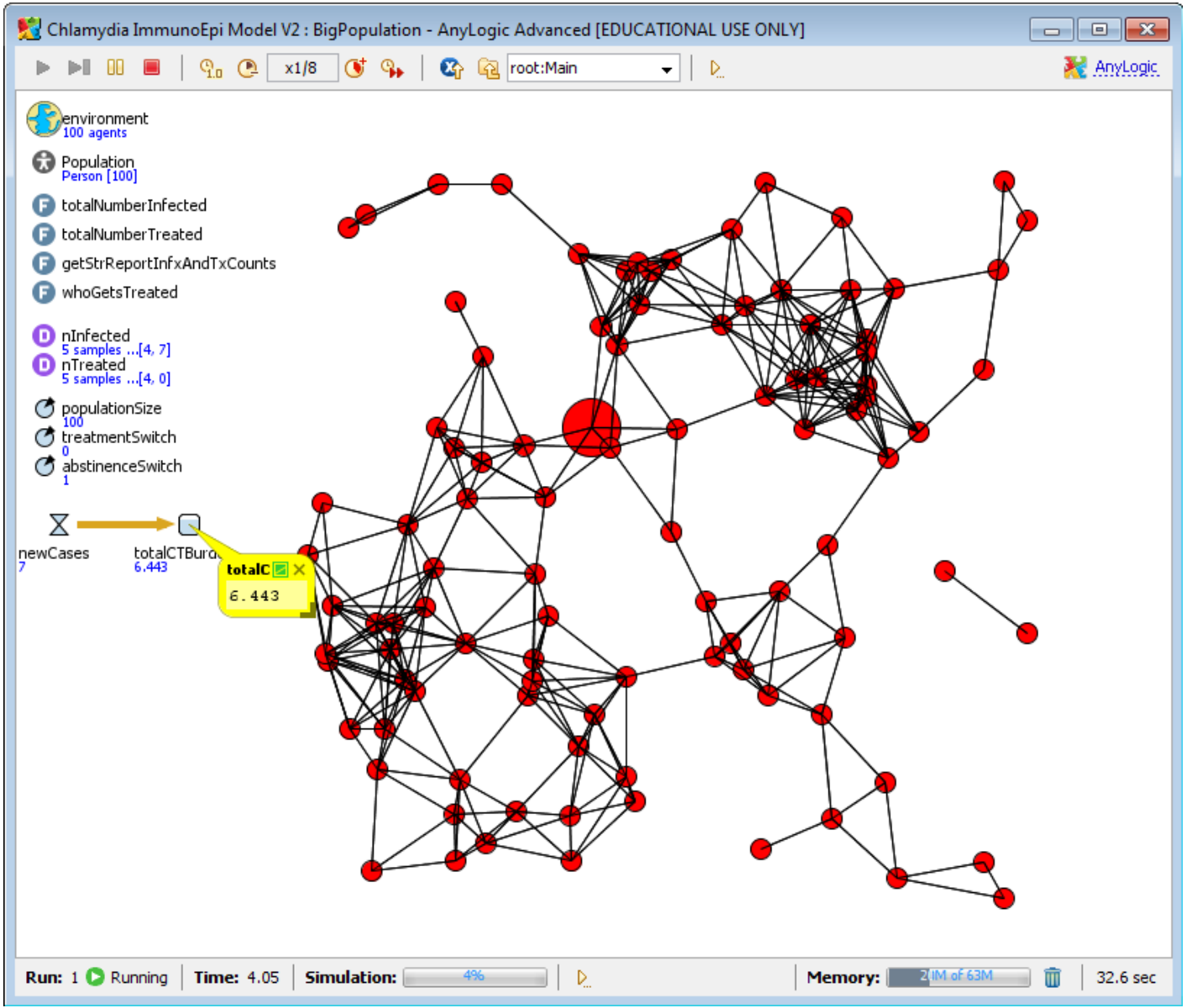


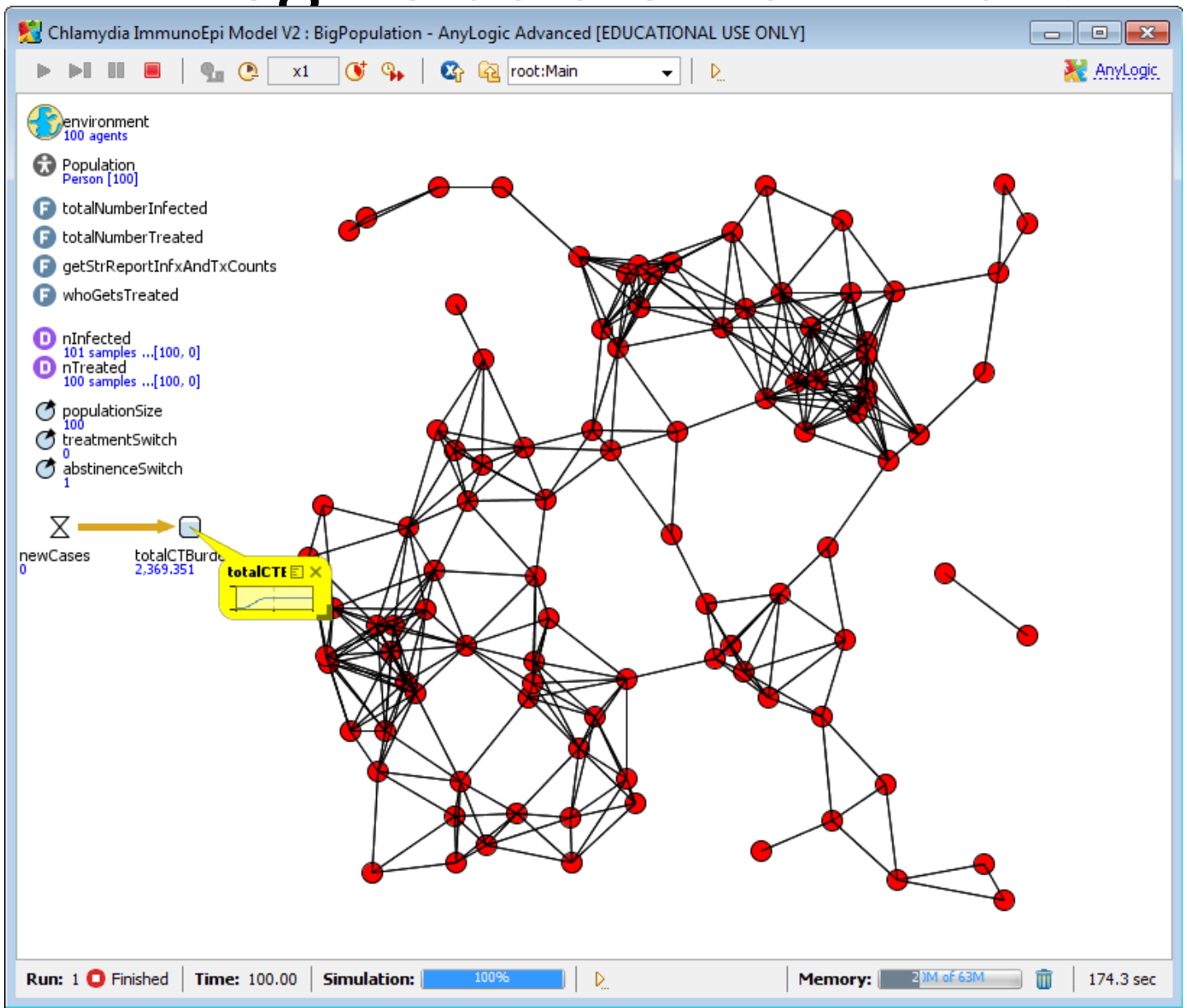
Chart Use of Datasets



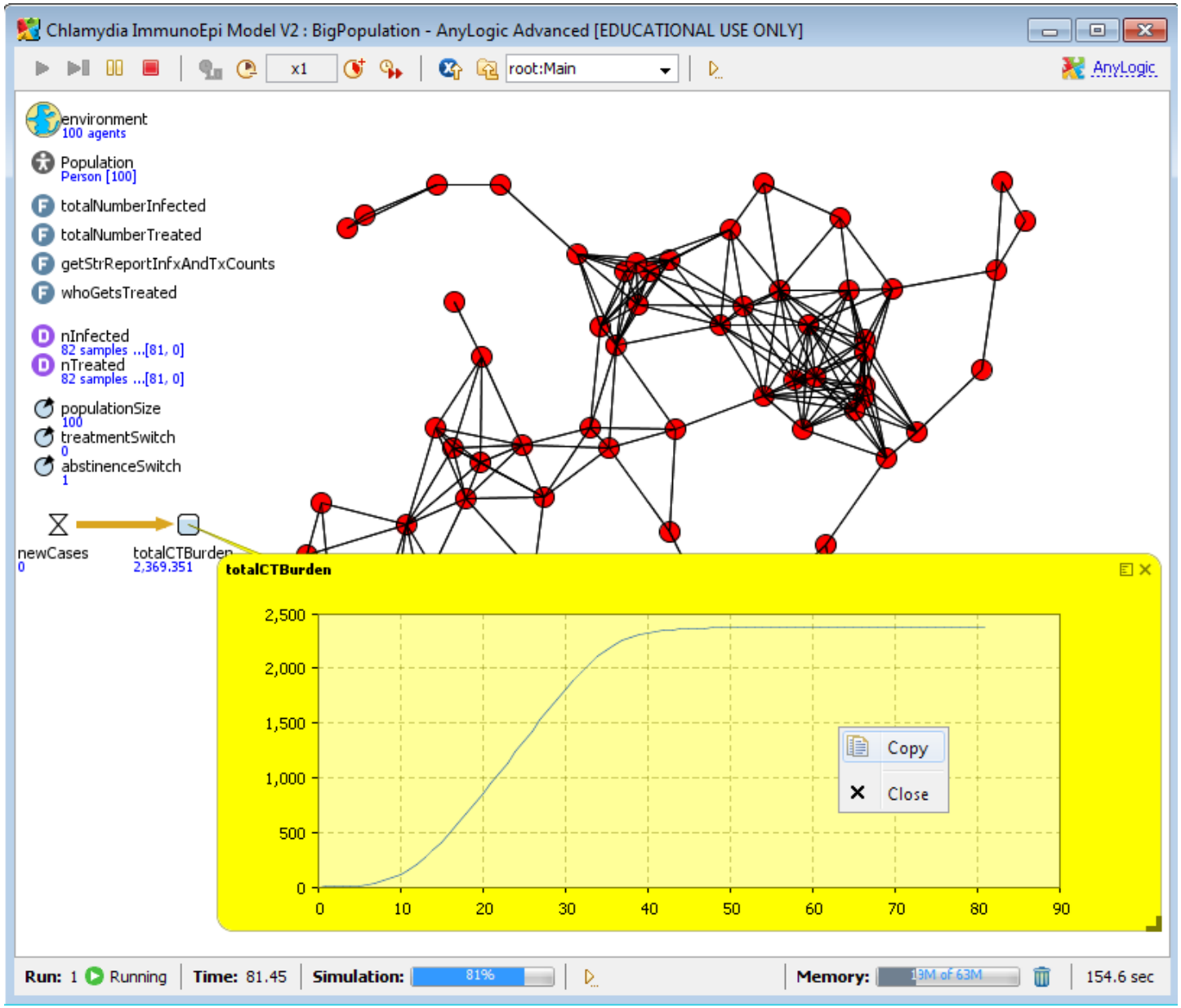
Ad-hoc Export



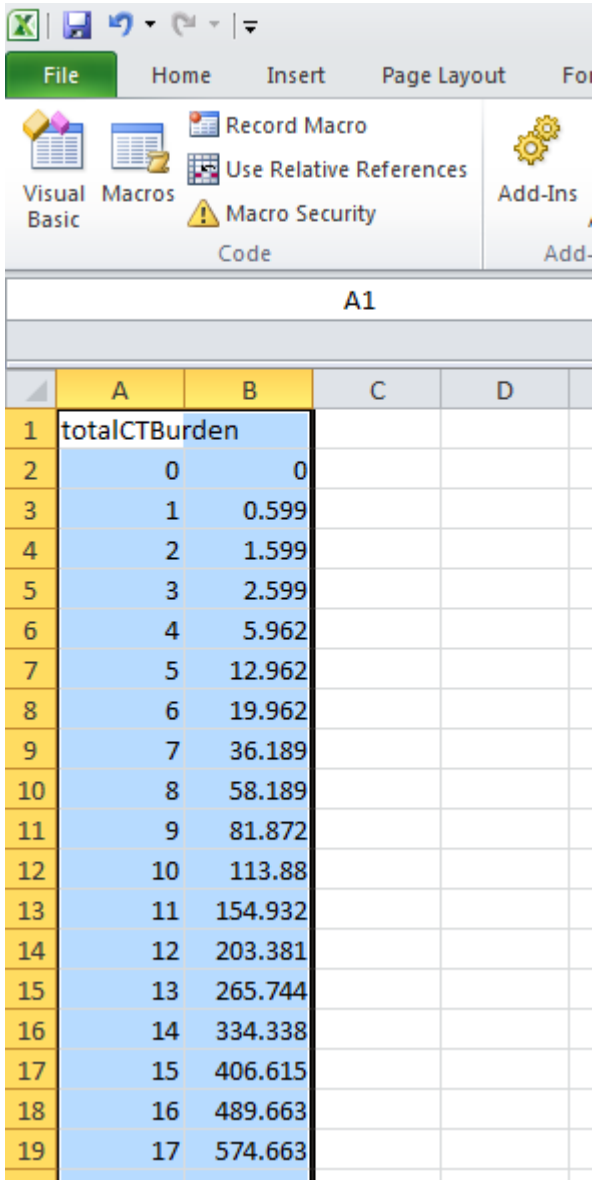
Begins as a Small Chart



Copying Data



Data Exported from Ad-Hoc Chart

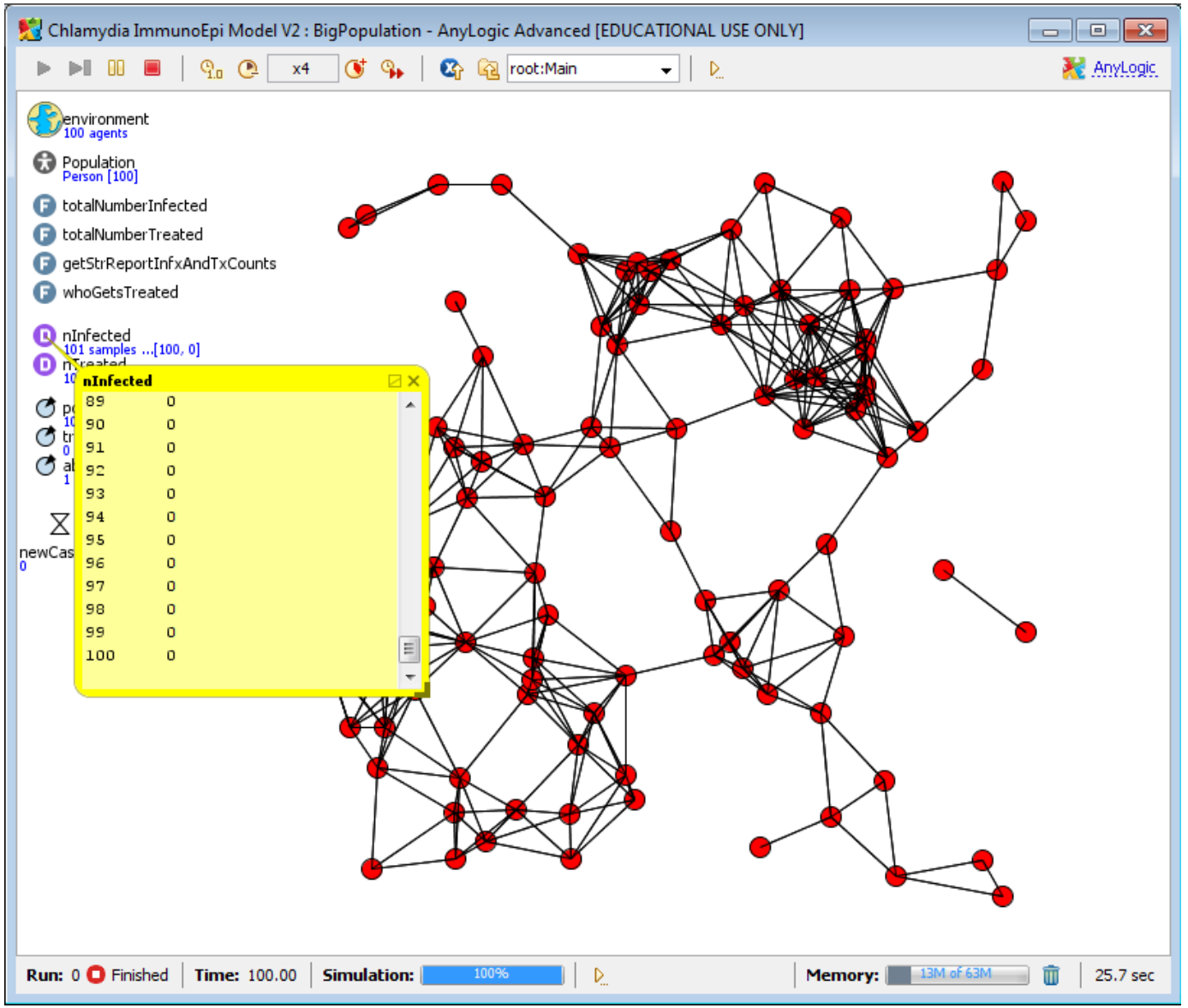


	A	B	C	D
1	totalCTBurden			
2	0	0		
3	1	0.599		
4	2	1.599		
5	3	2.599		
6	4	5.962		
7	5	12.962		
8	6	19.962		
9	7	36.189		
10	8	58.189		
11	9	81.872		
12	10	113.88		
13	11	154.932		
14	12	203.381		
15	13	265.744		
16	14	334.338		
17	15	406.615		
18	16	489.663		
19	17	574.663		

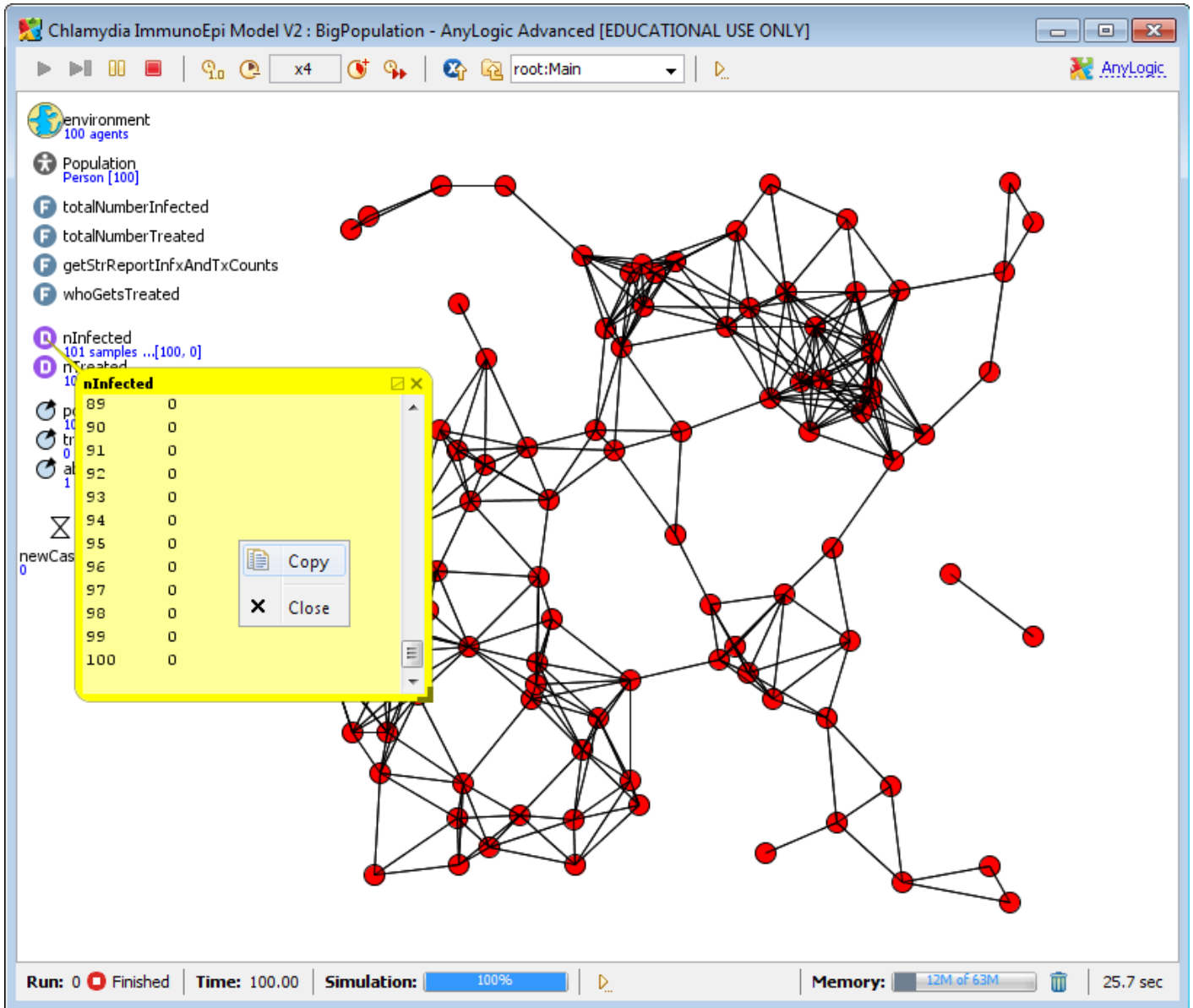
Techniques for Outputting Data

- Ad-Hoc Exports from variables
- Manual copies from visible datasets
- Capturing images of graphs
- Export to files
- Writing to console
- [AnyLogic Professional] Dataset archiving
- Export to databases

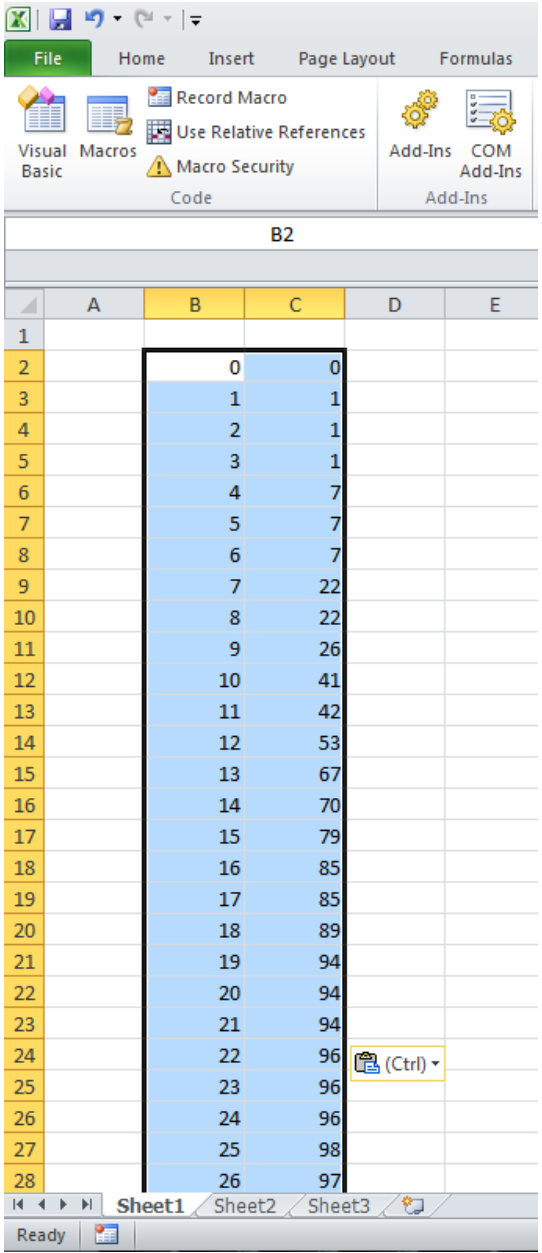
Manual Output from Datasets



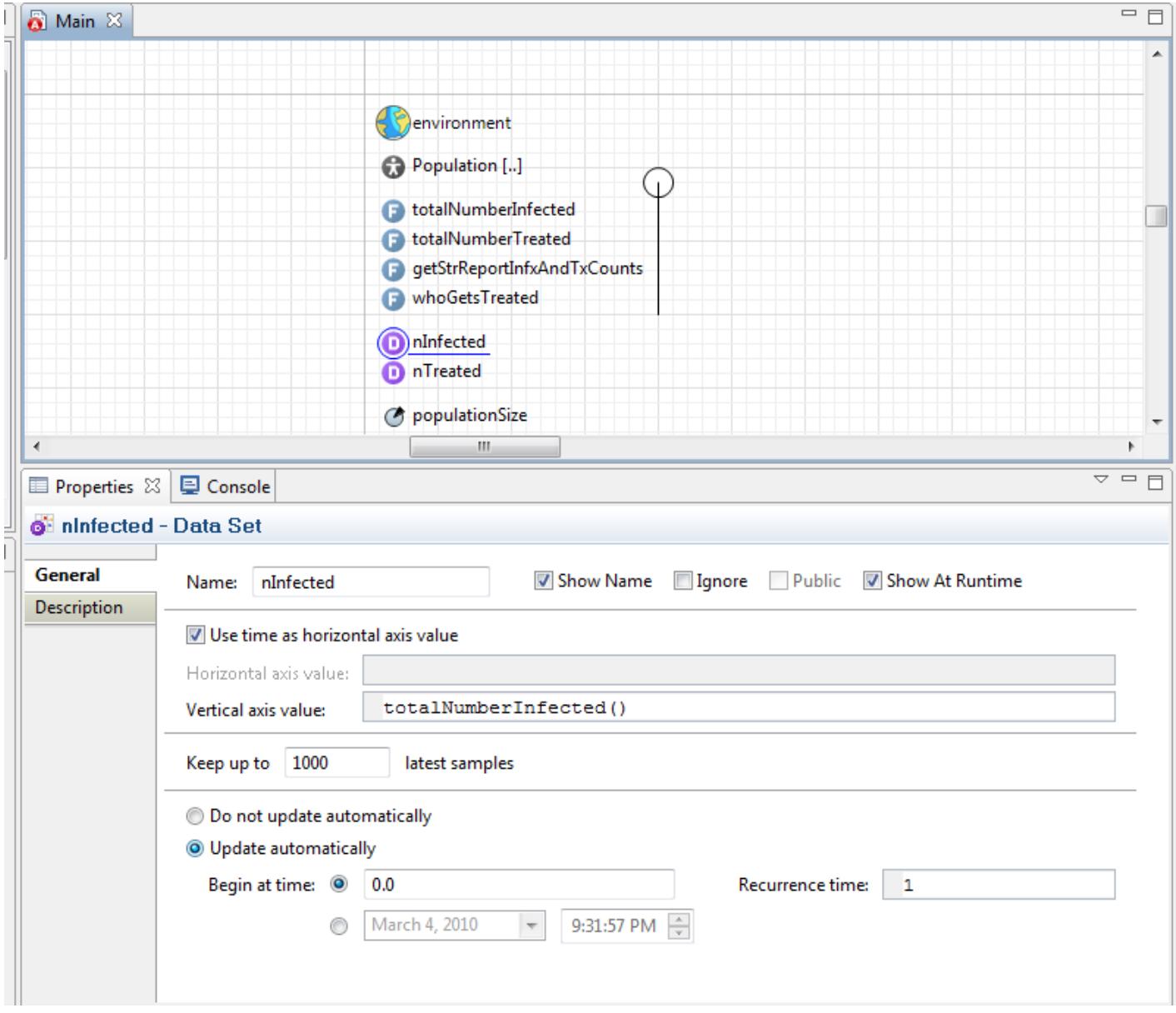
Right Clicking Gives Context Menu



Copied Data Can be Pasted into Excel



Declaratively Specifying Datasets



Supported Dataset Types

- Simple
  - holds values only -- no timestamps
- Timed
  - holds values and timestamps
- Phase
  - holds pairs of values but no timestamps
- Histogram
  - can define bins for data set
  - data set will record # falling in each bin

Techniques for Outputting Data

- Ad-Hoc Exports from variables
- Manual copies from visible datasets
- Capturing images of graphs
- Output to console
- Export to files
- [AnyLogic Professional] Dataset archiving
- Export to databases

Output to Console

- Pros
  - Easy to program
    - ActiveObject.traceln(String str) outputs string to console
    - System.out.println(String str) (Black)
    - System.err.println(String str) (Red)
  - Readily visible
  - Copy & Paste to another document
- Cons
  - May be mixed with other output (easy to miss other output)
  - Limited length
  - Depends on memory to copy
  - Less structured

Techniques for Outputting Data

- Ad-Hoc Exports from variables
- Manual copies from visible datasets
- Capturing images of graphs
- Writing to console
- Export to files
- [AnyLogic Professional] Dataset archiving
- Export to databases

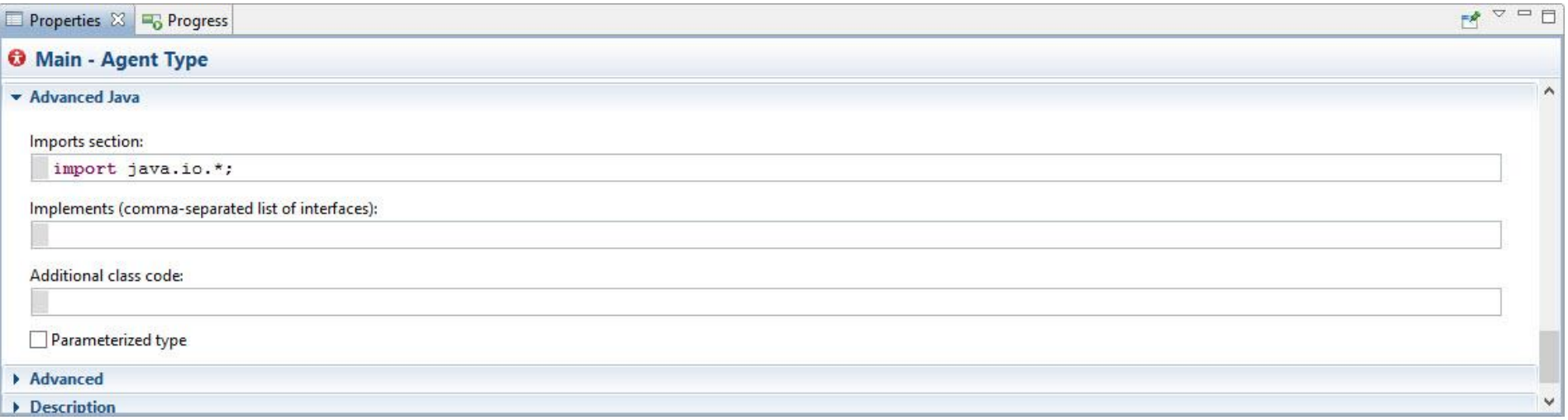
Data Output to File

- Pros
  - Simple to perform
  - Relatively easy to import into e.g. Excel, R, etc.
  - Files can be readily archived
- Cons
  - Awkward to have multiple data sets output to the same file
  - Awkward to draw combine from multiple files
  - Denormalization: Requires either
    - Duplication of scenario-wide information (e.g. parameter values) on each row
    - Separate header section & later section

Outputting a Dataset to a File Requires 2 Steps

- “Importing” (specifying how to find) the necessary Java code
- Defining the code

Step 1: Importing the Necessary Java Libraries



Step 2: Code to Export Dataset to File

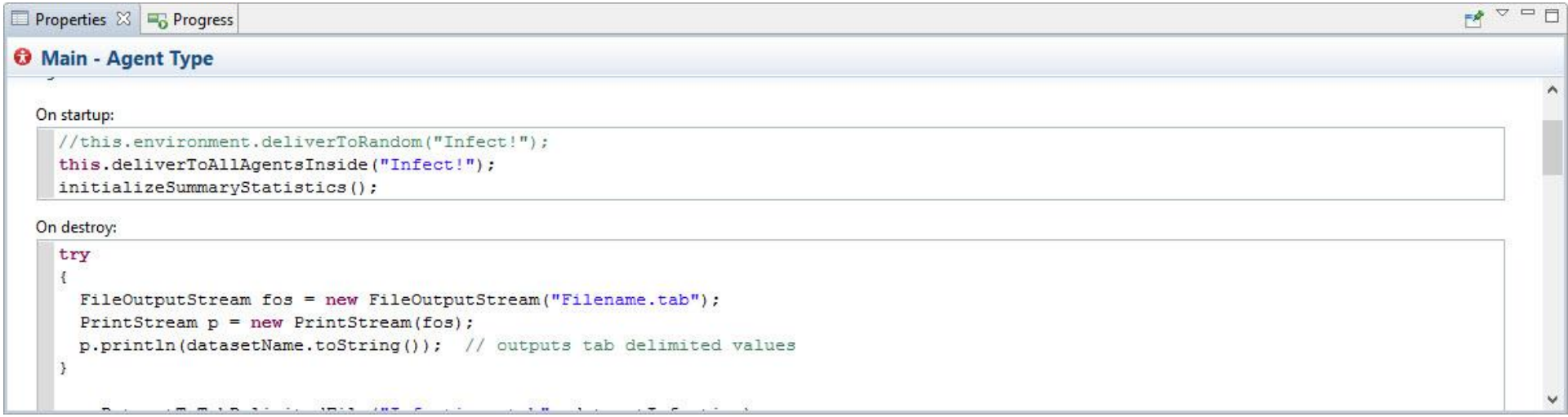
```
try
{
  FileOutputStream fos = new
  FileOutputStream("Filename.tab");
  PrintStream p = new PrintStream(fos);
  p.println(datasetName.toString());  // outputs
  tab delimited values
}
catch (Exception e)
{
   traceln("Could not write to file.");
}
```

Substitute whatever file name you wish to use
You may wish to put a "path" in front of this

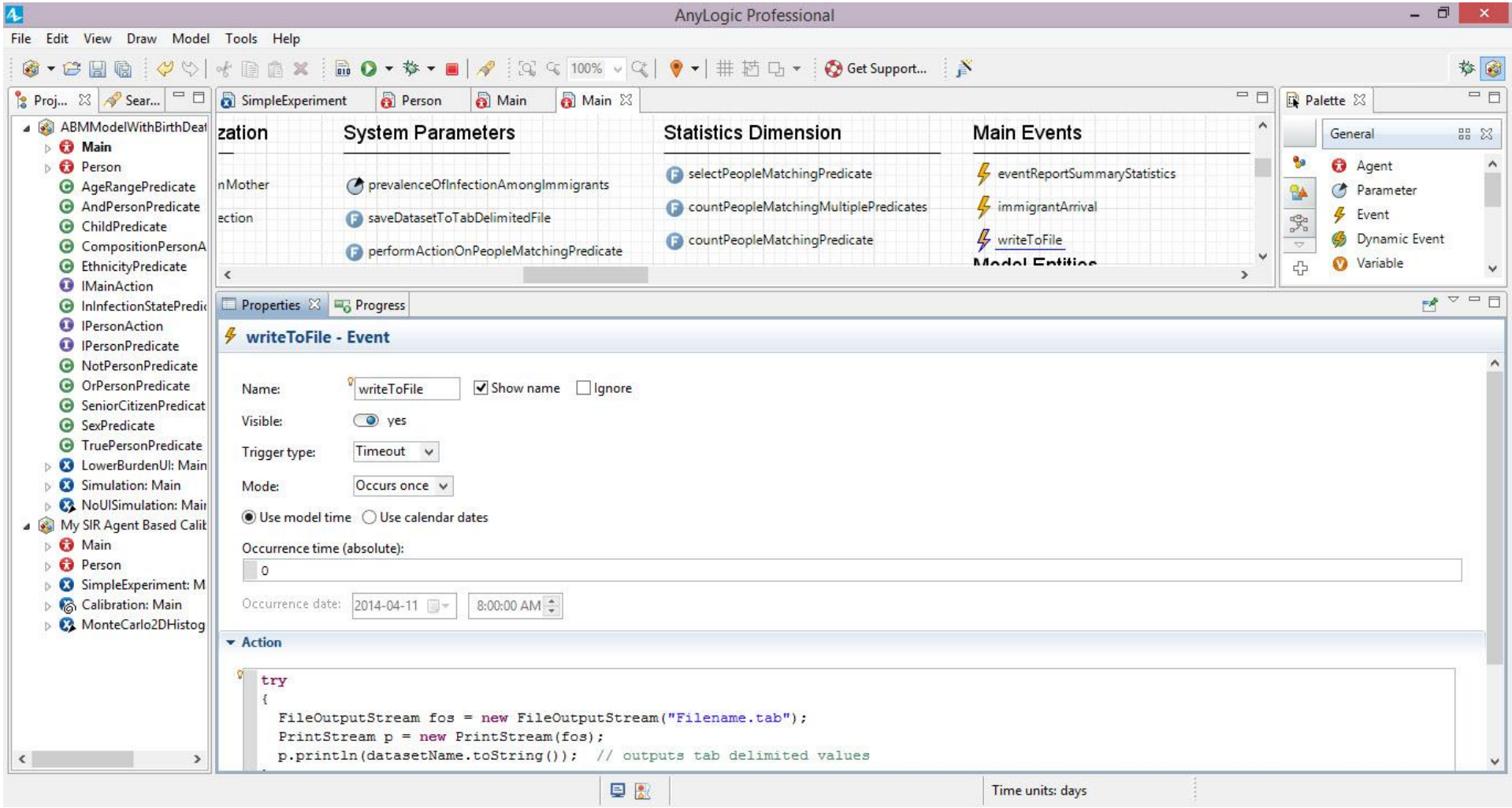
Substitute the name of the dataset
You wish to output

Suggestion: For greater versatility, place this in a function that takes the file name as a parameter.

Where to Put the Code to Output the Dataset
Option 1: In “Destroy Code” for Main



Where to Put the Code to Output the Dataset
Option 2: In "Action" for an Event Triggered at times



Techniques for Outputting Data

- Ad-Hoc Exports from variables
- Manual copies from visible datasets
- Capturing images of graphs
- Writing to console
- Export to files
- [AnyLogic Professional] Dataset archiving
- Export to databases

Output to Databases: Tradeoffs

- Pros
  - More flexible than string output to file
  - Can query from diverse tools (e.g. excel, R, SPSS, SAS, etc.)
  - Can easily clean up
  - For larger databases
    - Transactional (either writes entirely or not at all)
    - Can query from remote machines
- Cons
  - More programming
  - Need to set up a database

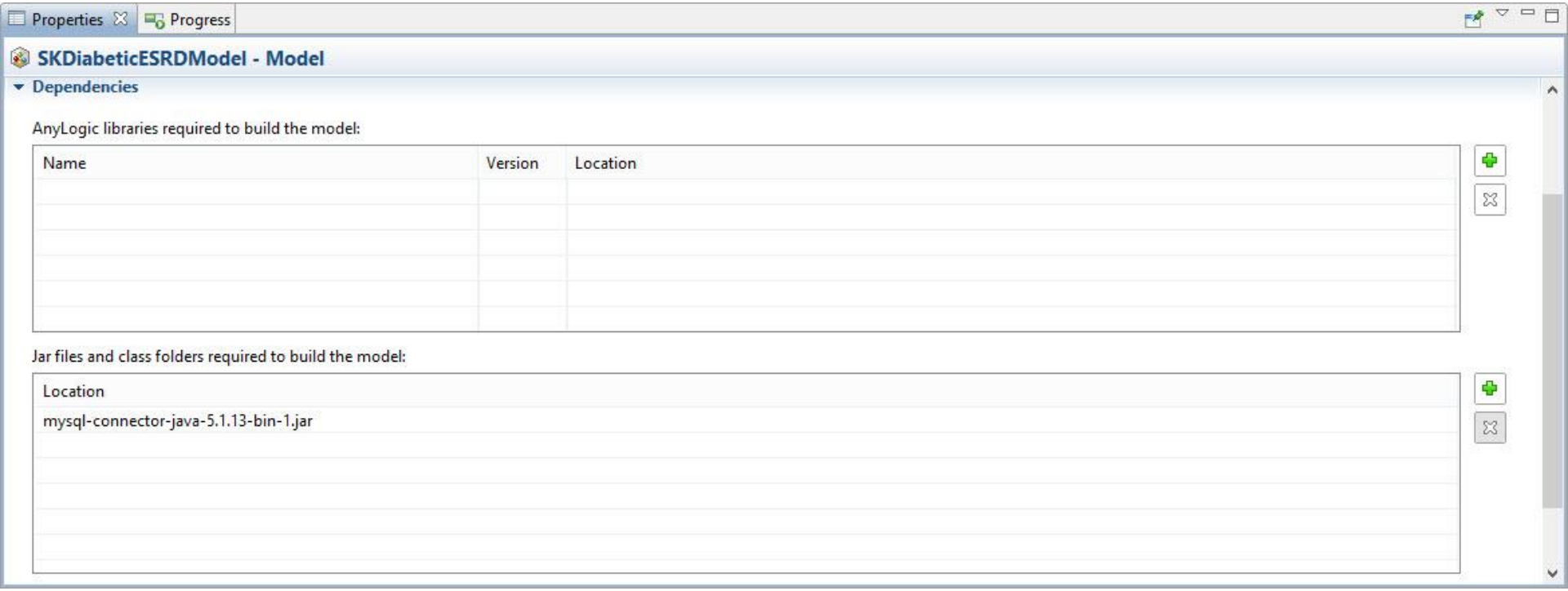
Output to Databases: Steps

- One Time:
  - Install database on computer
  - Setting up the database schema
  - Add reference to database libraries
- Each scenario during simulation
  - Open database connection at start of model
  - Optionally, “insert” model version & parameter information into the database
  - Periodically during simulation
    - “insert” values into databases
  - At end of model execution, close database connection

Relevant Databases

- Databases most oriented towards single users & single computers
  - MS Access
  - H2
  - These databases less robust => risk of corruption
  - These are often quite fast
- Databases oriented towards multiple users & multiple computers
  - Oracle
  - DB2
  - MS SQL Server
  - Open source
    - Postgres
    - Derby
    - MySQL
- More robust
- Support remote access

Database Dependencies (MySQL database)



Options for Database Access

- AnyLogic Professional: Built-in visual database classes
  - Simplify the composition of database operations
- Direct calling of database operations in Java's "Java DataBase Connectivity" (JDBC) Library
  - Note ODBC "bridge" for windows database driver support
- Custom database classes
  - We would be happy to share our simple interface
  - More refined interfaces possible

Example Simple Database Class for SQL Relational Database Systems

A Method is associated with each of

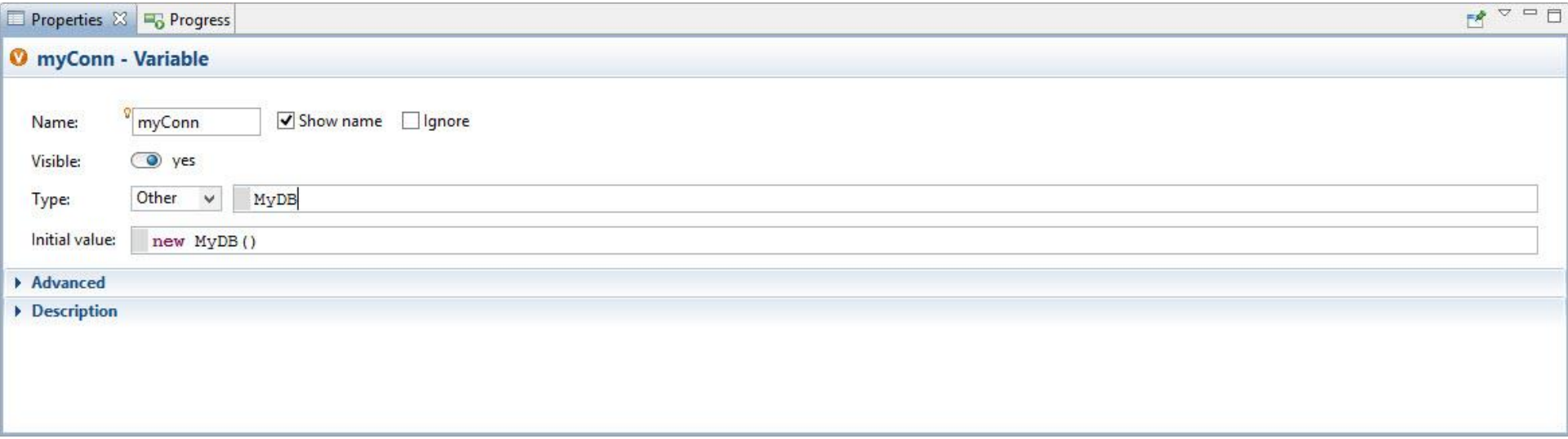
- Execute Query
- Insert

```
public class MyDB {
    private static String DriverName = "com.mysql.jdbc.Driver";
    private String dbURL = "jdbc:mysql://localhost:3306/mydb";
    private String dbuser = "root";
    private String dbpassword = "2005051146";
//This is due to consideration of conflicts between database of AnyLogic and java.sql.* package.
    private java.sql.Connection conn = null;
    private java.sql.Statement stmt = null;
    private java.sql.ResultSet rs = null;
    /**
     * Default constructor
     */
    public MyDB(){
     try{
      Class.forName(DriverName);
     }catch(java.lang.ClassNotFoundException e){
            System.err.println(e.getMessage());
            System.out.println("Error with constructor!");
     }
}
/*************************************************
     *method name: executeQuery()
     *Query
     *return value: ResultSet
*************************************************/
     public java.sql.ResultSet executeQuery(String sql){
            try{
                    conn = DriverManager.getConnection(dbURL,dbuser,dbpassword);
                    stmt = conn.createStatement();
                    rs=stmt.executeQuery(sql);
            }catch(SQLException ex){
                    System.err.println(ex.getMessage());
                    System.out.println("Error with executeQuery() method!");
                    }
            return rs;
     }

/*************************************************
      *method name: executeUpdate()
      *udpate, delete, and insert
      *return value: int
*************************************************/
     public int executeUpdate(String sql){
            int result=0;
            try{
                    conn = DriverManager.getConnection(dbURL,dbuser,dbpassword);
                    stmt=conn.createStatement();
                    result=stmt.executeUpdate(sql);
            }catch(SQLException ex){
                    result=0;
                    System.err.println(ex.getMessage());
                    }
            return result;
     }


 @Override
 public String toString() {
  return super.toString();
 }
}
```

Example: Execute Query

```
/*******************************************************
     *method name: executeQuery()
     *Query
     *return value: ResultSet
*******************************************************/
     public java.sql.ResultSet executeQuery(String sql){
               try{
                         conn = DriverManager.getConnection(dbURL,dbuser,dbpassword);
                         stmt = conn.createStatement();
                         rs=stmt.executeQuery(sql);
               }catch(SQLException ex){
                         System.err.println(ex.getMessage());
                         System.out.println("Error with executeQuery() method!");
                         }
               return rs;
     }
```

Setup for Database Class



Example Database Output Code

A database query language (SQL) statement

```
double simulated_time = time()+1975;
for(int k=0;k<Cube[0][0].length;k++){
        String sql ="INSERT INTO dataset(agegroup,ethnicity,state,modeltime,amount,simulation_id) VALUES (
                     "+1+","+0+","+k+","+simulated_time+","+Cube[1][0][k]+","+simulation_id+")";
        int ret = myConn.executeUpdate(sql);
        if(ret == 0){
           traceln("Adding new record"+1+" "+0+" "+k+" in "+Cube[1][0][k]+" dataset TABLE failed!");
           }
}
```

Requesting that the database class execute the SQL statemen

Checking to make sure that the insert worked properly

Database Output: Suggestions

- Maintain metadata
  - Purpose of run
  - Parameter settings
  - Model version (& possibly .alp file)
- Be mindful of performance & space burdens
  - Try to batch up data inserts
  - Be selective in what data to store, balancing pros & cons of storing more material
    - Pros: Analytic flexibility, greater understanding, less risk of having to re-run simulation
    - Cons: Mammoth database size, Impaired performance
  - Use a local database if possible

Database Input

- Database input can be desirable when “feeding in” certain data to model
  - Connection choreography
  - Agent movement patterns
  - Count of incident cases of a condition
  - Count of vaccinations over time
- Frequently this data is “quantized” into time units
  - In those cases, Dynamic Events can be helpful