

# Dealing with Data Gradients: “Backing Out” & Calibration

Nathaniel Osgood

Using Modeling to Prepare for Changing  
Healthcare Needs

Duke-NUS

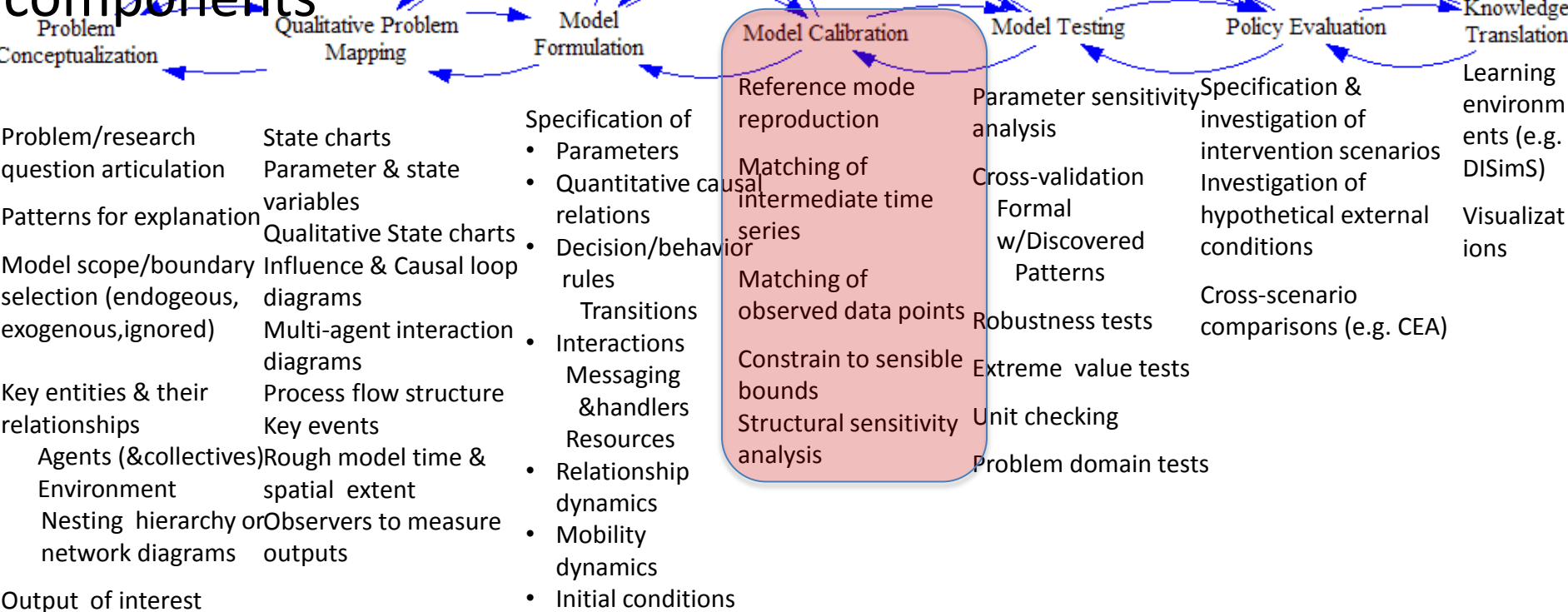
April 16, 2014

# ABM Modeling Process Overview

A Key Deliverable!

ODD:  
Design components  
& details

ODD: Overview  
& high-level design  
components



# Common Sources for Parameter Estimates

- Surveillance data
- Controlled trials
- Outbreak data
- Clinical reports data
- Intervention outcomes studies
- Calibration to historic data
- Expert judgement
- Meta-analyses

Material Adapted from  
External Source

Redacted from Public PDF  
for Copyright Reasons

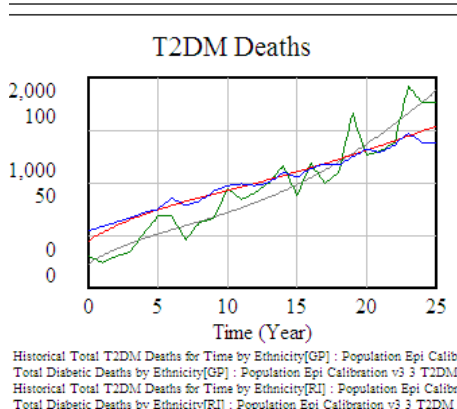
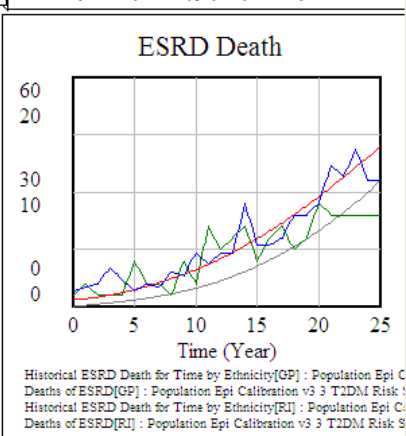
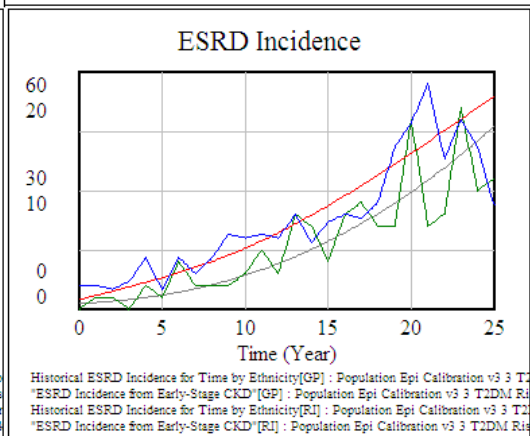
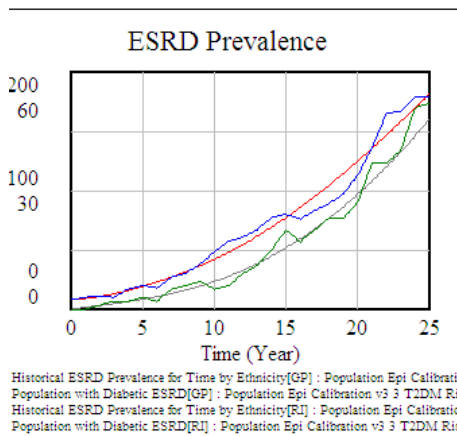
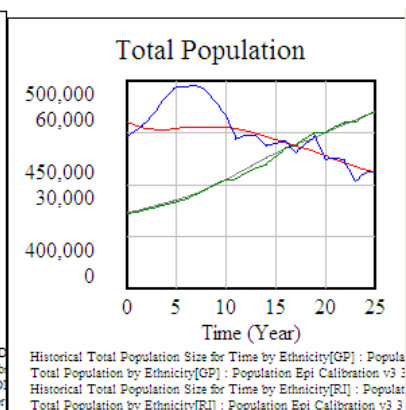
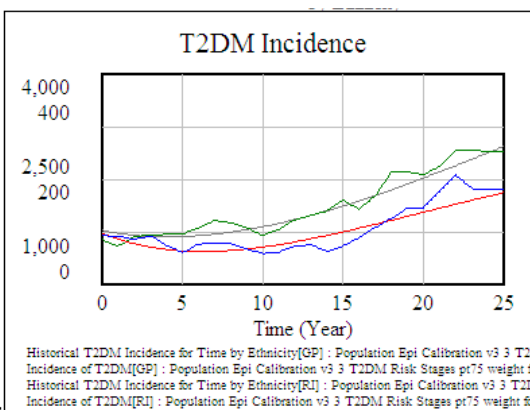
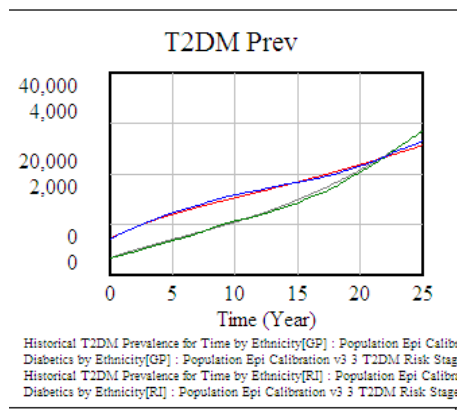
# Sensitivity Analyses

- Same relative or absolute uncertainty in different parameters may have hugely different effect on outcomes or decisions
- Help identify parameters that strongly affect
  - Key model results
  - Choice between policies
- We place more emphasis in parameter estimation into parameters exhibiting high sensitivity

# Dealing with Data Gradients

- Often we don't have reliable information on *some* parameters, but do have other data
  - Often have data on emergent behavior of system – doesn't relate to any one parameter, but a combination influences
  - Some parameters may not be observable, but some closely related observable data is available
  - Sometimes the data doesn't have the detailed breakdown needed to specifically address one parameter
    - Available data could specify sum of a bunch of flows or stocks
    - Available data could specify some function of several quantities in the model (e.g. prevalence)
- Some parameters may implicitly capture a large set of factors not explicitly represented in model
- There are two big ways of dealing with this: manually “backing out”, and automated calibration

# Recall: Single Model Matches Many Data Sources



# “Backing Out”

- Sometimes we can manually take several aggregate pieces of data, and use them to collectively figure out what more detailed data might be
- Frequently this process involves imposing some (sometimes quite strong) assumptions
  - Combining data from different epidemiological contexts (national data used for provincial study)
  - Equilibrium assumptions (e.g. assumes stock is in equilibrium – deriving prevalence from incidence)
  - Independence of factors (e.g. two different risk factors convey independent risks)

# Example

- Suppose we seek to find out the sex-specific prevalence of diabetes in some population
- Suppose we know from published sources
  - The breakdown of the population by sex ( $c_M, c_F$ )
  - The population-wide prevalence of diabetes ( $p_T$ )
  - The prevalence rate ratio of diabetes in women when compared to men ( $rr_F$ )
- We can “back out” the sex-specific prevalence from these aggregate data ( $p_F, p_M$ )
- Here we can do this “backing out” without imposing assumptions



# Backing Out

# male diabetics + # female diabetics = # diabetics

$$(p_M * c_M) + (p_F * c_F) = p_T * (c_M + c_F)$$

- Further, we know that  $p_F / p_M = rr_F \Rightarrow p_F = p_M * rr_F$

- Thus

$$(p_M * c_M) + ((p_M * rr_F) * c_F) = p_T * (c_M + c_F)$$

$$p_M * (c_M + rr_F * c_F) = p_T * (c_M + c_F)$$

- Thus

- $p_M = p_T * (c_M + c_F) / (c_M + rr_F * c_F)$

- $p_F = p_M * rr_F = rr_F * p_T * (c_M + c_F) / (c_M + rr_F * c_F)$

# Disadvantages of “Backing Out”

- Backing out often involves questionable assumptions (independence, equilibrium, etc.)
- Sometimes a model is complex, with several related known pieces
  - Even though we may know a lot of pieces of information, it would be extremely complex (or involve too many assumptions) to try to back out several pieces simultaneously

# Another Example: Joint & Marginal Prevalence

|        | Rural    | Urban    |       |
|--------|----------|----------|-------|
| Male   | $p_{MR}$ | $p_{MU}$ | $p_M$ |
| Female | $p_{FR}$ | $p_{FU}$ | $p_F$ |
|        | $p_R$    | $p_U$    |       |

Perhaps we know

- The count of people in each { Sex, Geographic } category
- Each marginal prevalence ( $p_R, p_U, p_M, p_F$ )

We need at least one more constraint (one possibility: assume  $p_{MR} / p_{MU} = p_R / p_U$ )

We can then derive the prevalence in each { Sex, Geographic } category

# Calibration: “Triangulating” from Diverse Data Sources

- Calibration involves “tuning” values of less well known parameters to best match observed data
  - Often try to match against *many* time series or pieces of data at once
  - Idea is trying to get the software to answer the question: “What must these (less known) parameters be in order to explain all these different sources of data I see”
- Observed data can correspond to complex combination of model variables, and exhibit “emergence”
- Frequently we learn from this that our model structure just can’t produce the patterns!

# Calibration

- Calibration helps us find a reasonable (specifics for) “dynamic hypothesis” that explains the observed data
  - Not necessarily the truth, but probably a reasonably good guess – at the least, a consistent guess
- Calibration helps us leverage the large amounts of diffuse information we may have at our disposal, but which cannot be used to directly parameterize the model
- Calibration helps us falsify models

# Calibration: A Bit of the How

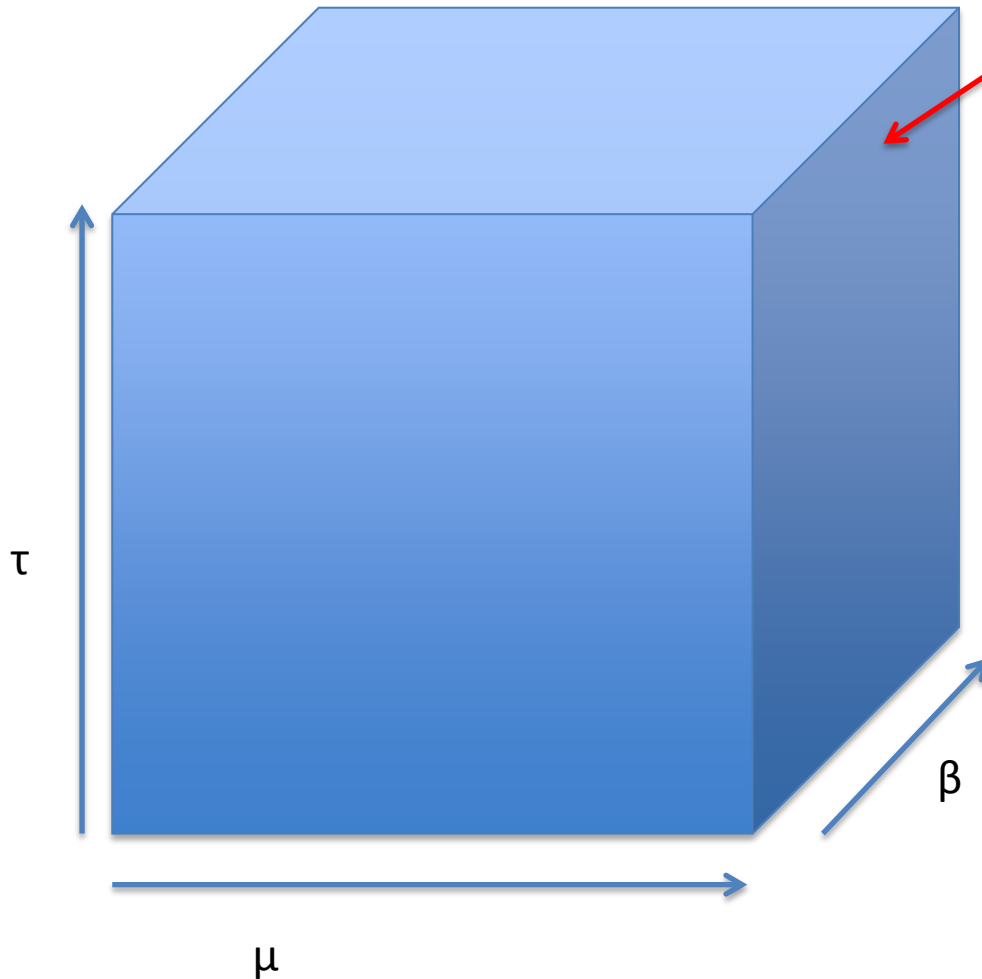
- Calibration uses a (global) optimization algorithm to try to adjust unknown parameters so that it automatically matches an arbitrarily large set of data
- The data (often in the form of time series) informs the objective function of the calibration
- The optimization algorithm will run the model many (thousands or more) times to find the “best” match for all of the data

# Required Information for Calibration

- Specification of what to match (and how much to care about each attempted match)
  - Involves an “error function” ( “penalty function”, “energy function”) that specifies “how far off we are” for a given run (how bad the fit is)
  - Alternative: specify “payoff function” (“objective function”)
- A statement of what parameters to vary, and over what range to vary them (the “parameter space”)
- Characteristics of desired optimization (tuning) algorithm
  - e.g. Single starting point of search?

# Envisioning “Parameter Space”

For each point in this space, there will be a certain “goodness of fit” of the model to the collective data





# Assessing Model “Goodness of Fit”

- To improve the “goodness of fit” of the model to observed data, we need to provide some way of quantifying it!
- Within the model, we
  - For each historic data, calculate discrepancy of model
    - Figure out absolute value of discrepancy from comparing
      - Historic Data
      - The model’s calculations
    - Convert the above to a fractional value (dividing by historic data)
  - Sum up these discrepancy

# Characteristics of a Desirable Discrepancy Metric

- **Dimensionless:** We wish to be able to add discrepancies together, regardless of the domain of origin of the data
- **Weighted:** Reflecting different pedigrees of data, we'd like to be able to weigh some matches more highly than others
- **Analytic:** We should be able to differentiate the function one or more times
- **Concave:** Two small discrepancies of size  $a$  should be considered more desirable than having one big discrepancy of size  $2a$  for one, and no discrepancy at all for the other.
- **Symmetric:** Being off by a factor of two should have the same weight regardless of whether we are  $2x$  or  $\frac{1}{2}x$
- **Non-negative:** No discrepancy should cancel out others!
- **Finite:** Finite inputs should yield finite discrepancies

# A Good Discrepancy Function (Assuming non-negative h & m)

Exponent  
>1  $\Rightarrow$  concave with respect to h-m

Taking average in denominator (together w/squaring of result) ensures symmetry with respect to h&m

$$w \cdot \left( \frac{h - m}{\text{average}(h, m)} \right)^2 = w \cdot \left( \frac{h - m}{\frac{h + m}{2}} \right)^2$$

Division  $\Rightarrow$  Dimensionless  
(Judging by *proportional* error, not absolute)

Only zero if h=m=0.

Denominator is only very small if numerator is as well!

# Considerations for Weighting

- **Purpose of model:** If we “care” more about a match with respect to some variables, we can more heavily weight matches for those variables
- **Uncertainty in estimate:** The more uncertain the estimate of the quantity, the lower the weight
- **Whether data exists:** no data => weight should be zero

# Example (Simplistic) Global Optimization Algorithm

- Starts at random position, tries to improve match (minimize error) by
    - Adjusting parameters
    - Running Model
    - Recording error function
  - Keeps on improving until reaches “local minimum” in error of fit
    - May add some randomness to knock out of local minima
- Many more sophisticated “global optimization” algorithms are available and can improve the outcome & speed of optimization (e.g. genetic algorithms, swarm-based methods)**



## Hands on Model Use Ahead



Load Sample Model:

**SIR Agent Based Calibration**

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

# Recall: Optimization Experiment in AnyLogic

AnyLogic Professional

File Edit View Draw Model Tools Help

Projects

- SIR Agent Based Calibration
  - Main
  - Person
  - Calibration: Main
  - MonteCarlo2DHistogram: Main

Calibration

Calibration - Optimization Experiment

Name: Calibration  Ignore

Top-level agent: Main

Objective:  minimize  maximize

`difference( dsInfectiousCurrent, dsInfectiousHistoric )`

Number of iterations: 500

Automatic stop

Maximum available memory: 128 Mb

Create default UI

Parameters

| Parameter          | Type       | Value |     |      |           |
|--------------------|------------|-------|-----|------|-----------|
|                    |            | Min   | Max | Step | Suggested |
| Average...ration   | fixed      | 15    |     |      |           |
| ContactRate        | continuous | 0.5   | 5   | 0    |           |
| Infectio...ability | continuous | 0.1   | 0.8 | 0    |           |
| TotalPopulation    | fixed      | 10000 |     |      |           |

Time units: days X=...70

Stops after best objective ceases to significantly improve  
*Caveat Modelor:*  
May prematurely terminate the optimization

Stops after 500 optimization iterations

Varying these parameters

# An Optimization Experiment in AnyLogic Using Built-in Difference Function (Euclidean distance)

The screenshot displays the AnyLogic Professional interface for configuring an optimization experiment. The main window is titled "Calibration - Optimization Experiment".

**Objective:** `difference( dsInfectiousCurrent, dsInfectiousHistoric )`

**Parameters:**

| Parameter          |            |       |     |   |
|--------------------|------------|-------|-----|---|
| Average...ration   |            |       |     |   |
| ContactRate        | continuous | 0.5   | 5   | 0 |
| Infectio...ability | continuous | 0.1   | 0.8 | 0 |
| TotalPopulation    | fixed      | 10000 |     |   |

**difference**  
public static double difference(DataSet ds, TableFunction f)  
Difference function which is always not-negative and reflects difference between given data set and table function in their common arguments range  
Parameters:  
ds - data set  
f - table function with linear interpolation



# Finding the Definition

The screenshot shows the AnyLogic University help interface. At the top, the search bar contains the text "difference dataset dataset" and the scope is set to "All topics". The search results are displayed in a list on the left, with the top result being "Collects data (PDF, CDF, etc.) of an array of histograms, each having a certain range of base (x) values and a range of data - y values. When an item (x,y) is added to Hist...".

The main content area is divided into two panes. The left pane, titled "All Classes", lists various classes and packages, including [com.xi.anylogic.engine](#), [com.xi.anylogic.engine.analysis](#), [com.xi.anylogic.engine.connector](#), and [com.xi.anylogic.engine.presentation](#). The right pane displays the definition for the `difference` method, which is a public static double method that takes two `DataSet` objects as arguments and returns a double value representing the square root of the average of the square of the difference between the two data sets.

The search results list includes the following items:

- Collects data (PDF, CDF, etc.) of an array of histograms, each having a certain range of base (x) values and a range of data - y values. When an item (x,y) is added to Hist...**
- Compare Runs Experiment**  
This is an interactive experiment that allows you to input the model parameters, run simulation, and add the simulation output to the charts where they can be compared with...
- Calibration Experiment**  
When you have your model structure in place, you may wish to tune some parameters of the model so that its behavior in particular conditions matches a known (historical) pa...
- Sensitivity Analysis Experiment**  
This experiment helps you to explore how sensitive are the simulation results to changes of the model parameters. The experiment runs the model multiple times varying one o...
- Monte Carlo Experiment**  
Monte Carlo experiment obtains and displays a collection of simulation outputs for a stochastic model or for a model with stochastically varied parameter(s). You can find t...
- AnyLogic Professional**  
edition is the ultimate solution for development of large and complex simulation models and sophisticated animations, embedding models into various IT environments, and cre...
- Statistics**  
The Statistics object calculates statistical information (mean value, minimum, maximum, etc.) on a series of data samples of type double. The object works differently depen...
- AnyLogic 6.5 New Features**  
3D animation Easy access to MS Excel files on all platforms "How to..." models and other materials to support learning Model documentation in one click New objects and improv...
- Parameter Variation**  
AnyLogic affords an opportunity to run model with different model parameters and analyze how some certain parameters affect the model behavior. You don't need to run your m...
- Optimization Experiment**  
If you need to run a simulation and observe system behavior under certain conditions, as well as improve system performance, for example, by making decisions about system p...

The "All Classes" list includes:

- [AbstractShapeGISMap](#)
- [ActiveObject](#)
- [ActiveObjectArrayList](#)
- [ActiveObjectCollection](#)
- [ActiveObjectIntegrationMan](#)
- [ActiveObjectInkedHashSet](#)
- [ActiveObjectList](#)
- [Agent](#)
- [AgentContinuous](#)
- [AgentContinuous2D](#)
- [AgentContinuous3D](#)
- [AgentContinuousGIS](#)
- [AgentDiscrete2D](#)
- [Area2D](#)
- [Area3D](#)
- [BarChart](#)
- [Camera3D](#)
- [Chart](#)
- [Chart.Properties](#)
- [Chart1D](#)
- [Chart1DSum](#)
- [Chart2D](#)
- [Chart2DPlot](#)
- [Chart2DPlot.Appearance](#)
- [ChartItem](#)
- [Configuration3D](#)
- [CustomDistribution](#)
- [Database](#)
- [DataItem](#)
- [DataSet](#)

The `difference` method definition is as follows:

```
public static double difference(DataSet ds1,
                              DataSet ds2)
```

Difference function which is always not-negative and reflects difference between 2 given data sets in their common arguments range

**Parameters:**

- ds1 - data set
- ds2 - data set

**Returns:**

- square root of the average of square of difference between linearly interpolated data sets
- The integration range is the intersection of argument ranges of data sets

The `millisecond` method definition is as follows:

```
public double millisecond()
```

Returns a time value equal to one millisecond according to the current time unit setting.

**Returns:**

- a time value equal to one millisecond

The `second` method definition is as follows:

```
public double second()
```

Returns a time value equal to one second according to the current time unit setting.

**Returns:**

- a time value equal to one second

The `minute` method definition is as follows:

```
public double minute()
```

Returns a time value equal to one minute according to the current time unit setting.

# An Optimization Experiment in AnyLogic with a custom difference function

The screenshot displays the AnyLogic Professional interface for a calibration experiment. The main workspace shows a chart titled "Calibration of Agent Based SIR Model" with a "Run calibration" button and a "Calibration progress" indicator. The properties panel is configured for a "Calibration - Optimization Experiment" with the objective set to "difference ()". The number of iterations is set to 500, and the maximum available memory is 128 Mb. The parameters table below shows the settings for the experiment.

| Parameter          | Type       | Min   | Max | Step | Suggested |
|--------------------|------------|-------|-----|------|-----------|
| Average...ration   | fixed      | 15    |     |      |           |
| ContactRate        | continuous | 0.5   | 3   | 0    |           |
| Infectio...ability | continuous | 0.1   | 0.8 | 0    |           |
| TotalPopulation    | fixed      | 10000 |     |      |           |

Annotations in the image include:

- A blue arrow pointing to the "difference ()" objective function in the properties panel, labeled "Custom distance function".
- A yellow arrow pointing to the "ContactRate" parameter in the parameters table, labeled "Varying these parameters".
- A red arrow pointing to the "InfectiousHistoric difference" function in the project tree.

# Defining a Payoff Function

Caveat: Here, Non-Analytic, Non-Concave

AnyLogic Professional

File Edit View Draw Model Tools Help

Projects

- SIR Agent Based Calibration\*
- Main
- Person
- Calibration: Main
  - Presentation
  - Analysis Data
  - Functions
    - InfectiousHistoric
    - difference
  - MonteCarlo2DHistogram: Main

Calibration

datasetBestFeasibleObjective

difference

These data correspond to  
ContactRate = 1.5  
InfectionProbability = 0.4

InfectiousHistoric

dsInfectiousHistoric

Run callit

Iteration:  
Replication:  
Objective: ↓

Parameters

ContactRate  
InfectionProbal

Properties

Progress

difference - Function

Function body

```
int diff = 0;
for(int i=0; i<dsInfectiousCurrent.size(); i++)
    diff += abs ( dsInfectiousCurrent.getY(i) - dsInfectiousHistoric.getY(i));
return diff;
```

Advanced

Description

Build completed successfully. Time: 0.156 s.

Time units: daily

**Computing absolute discrepancy  
Between historic & model values at  
specific point (index i) during realization**

# Historic Data Captured via Table Function

The screenshot displays the AnyLogic Professional interface. The main workspace shows a calibration model with a grid and several components: 'datasetBestFeasibleObjective', 'difference', 'InfectiousHistoric', and 'dInfectiousHistoric'. A green callout box points to the 'InfectiousHistoric' component with the text: 'These data correspond to ContactRate = 1.5, InfectionProbability = 0.4'. The 'Properties' panel at the bottom is open to the 'InfectiousHistoric - Table Function' section. The 'Name' field is 'InfectiousHistoric', 'Visible' is checked, and 'Interpolation' is set to 'Linear'. The 'Table Data' section shows a table with the following data:

| Argument | Value |
|----------|-------|
| 2        | 3     |
| 4        | 8     |
| 6        | 24    |
| 8        | 71    |
| 10       | 202   |
| 12       | 558   |

The 'Time units: days' is indicated at the bottom right of the interface.

How to interpolate (“fill in”) between data points

# Populating a Dataset with Historic Data

AnyLogic Professional

File Edit View Draw Model Tools Help

Projects

- SIR Agent Based Calibration\*
- Main
- Person
- Calibration: Main
- Presentation
- Analysis Data
- Functions
- InfectiousHistoric
- difference
- MonteCarlo2DHistogram: Main

Calibration

Run callit

Iteration:

Replication:

Objective: ↓

Parameters

General

- Agent
- Parameter
- Event
- Dynamic Event
- Variable
- Collection
- Function
- Table Function

Properties Progress

Calibration - Optimization Experiment

Java actions

Initial experiment setup:

```
dsInfectiousHistoric.fillFrom( InfectiousHistoric );
```

Before each experiment run:

```
datasetCurrentObjective.reset();  
datasetBestFeasibleObjective.reset();
```

Before simulation run:

After simulation run:

```
dsInfectiousCurrent.fillFrom( root.InfectiousDS );
```

After iteration:

```
if ( getCurrentIteration() == getBestIteration() )  
  dsInfectiousBest.fillFrom( dsInfectiousCurrent );
```

Time units: days

Populating the dataset from the previously defined table function

# Stochastics in Agent-Based Models

- Recall that ABMs typically exhibit significant stochastics
  - Event timing within & outside of agents
  - Inter-agent interactions
- When calibrating an ABM, we wish to avoid attributing a good match to a particular set of parameter values simply due to chance
- To reliably assess fit of a given set of parameters, we need to repeatedly run model realizations
  - We can take the mean fit of these realizations

# Recall: Important Distinction (Declining Order of Aggregation)

- Experiment
  - Collection of simulations
- Simulation (i.e., Scenario)
  - Collection of replications that can yield findings across set of replications (e.g. mean value)
- Replication (i.e., Realization)
  - A Single realization (“run”) of the model, with a unique random number seed

# Populating the Appropriate Datasets

The screenshot shows the AnyLogic Professional interface for a 'Calibration - Optimization Experiment'. The 'Java actions' section is divided into four categories: 'Initial experiment setup', 'Before each experiment run', 'Before simulation run', and 'After simulation run'. The 'After simulation run' section contains a code block that populates datasets based on the current iteration and the best iteration found.

```
Initial experiment setup:  
dsInfectiousHistoric.fillFrom( InfectiousHistoric );  
  
Before each experiment run:  
datasetCurrentObjective.reset();  
datasetBestFeasibleObjective.reset();  
  
Before simulation run:  
  
After simulation run:  
dsInfectiousCurrent.fillFrom( root.InfectiousDS );  
After iteration:  
if( getCurrentIteration() == getBestIteration() )  
    dsInfectiousBest.fillFrom( dsInfectiousCurrent );
```

Annotations on the screenshot include:

- Populates historic data up front from table fn** (Red text, pointing to the initial setup code)
- These datasets are within the experiment Persist beyond the simulation** (Yellow text, pointing to the 'Analysis Data' folder in the Projects pane)
- If this is the best iteration, saves away the results** (Green text, pointing to the 'After iteration' code block)
- Retaining the Current value After the realization (Simulation run)** (Blue text, pointing to the 'After simulation run' code block)

A green callout box in the diagram area contains the text: "These data correspond to ContactRate = 1.5, Infection Probability = 0.1, InfectiousHistoric".



# Running Calibration in AnyLogic

**Calibration of Agent Based SIR Model**

Run calibration

|              | Current | Best     |
|--------------|---------|----------|
| Iteration:   | 15      | 2        |
| Replication: | 4       |          |
| Objective: ↓ | 79,534  | 74,970.4 |

**Parameters**

ContactRate: 2.735  
InfectionProbability: 0.139

Copy the best solution to the clipboard

The built-in OptQuest optimizer is used to calibrate an agent based model of contagious disease diffusion. In the model each person has 3 possible states: Susceptible, Infectious and Recovered (SIR). Initially all but few people are susceptible, and few - infectious. A person can contact another person, and in case one is susceptible and another - infectious, the first may get infected with a certain probability. The objective is to find the parameters of the agents (contact frequencies and infection probabilities) so that the output of the simulation model fits best with the historical data - the dynamics of infectious population (disease prevalence). As the model is stochastic, the calibration is done under uncertainty, and simulation replications are used. A separate 1st order Monte Carlo experiment is included to demonstrate the output of the agent based model.

**Calibration progress**

Legend: ■ Current objective, — Best feasible objective

**Historic data, best fitting and current simulation output**

Legend: — Historic data, — Current output, — Best output

Run: 51 Error Experiment: 100% Simulations: Stop time not set 7.4 sec

Best payoff (objective) yet reached (lower is better)

Values of parameters being calibrated at best calibration thus far

# Optimization Constraints – Tests on Legitimacy of Parameter Values

The screenshot displays the AnyLogic Professional interface for a calibration experiment. The main window shows a 'Run calibration' button and a table comparing 'Current' and 'Best' values for 'Iteration', 'Replication', and 'Objective'. The 'Current' values are marked as 'infeasible'. A 'Calibration progress' graph shows a red line fluctuating between 0.6 and 1.0. The 'Properties' panel is open to the 'Calibration - Optimization Experiment' section, which includes 'Constraints' and 'Requirements' tables. The 'Constraints' table has columns for 'Enabled', 'Expression', 'Type', and 'Bound'. The 'Requirements' table also has columns for 'Enabled', 'Expression', 'Type', and 'Bound'. The 'Randomness' section shows 'Random number generation' options: 'Random seed (unique simulation runs)' (selected) and 'Fixed seed (reproducible simulation runs)' (with a 'Seed value' of 1). The 'Time units' are set to 'days'.

AnyLogic Professional

File Edit View Draw Model Tools Help

Projects

- SIR Agent Based Calibration\*
- Main
- Person
- Calibration: Main
  - Presentation
  - Analysis Data
    - datasetBestFeasibleObjective
    - datasetCurrentObjective
    - dsInfectiousBest
    - dsInfectiousCurrent
    - dsInfectiousHistoric
  - Functions
    - InfectiousHistoric
    - difference
  - MonteCarlo2DHistogram: Main

Run calibration

AnyLogic  
This model is © The AnyLogic Company. www.anylogic.com

|              | Current      | Best         |
|--------------|--------------|--------------|
| Iteration:   | infeasible ? | infeasible ? |
| Replication: | infeasible ? | infeasible ? |
| Objective: ↓ | ?            | ?            |

Calibration progress

Parameters

Properties Progress

Calibration - Optimization Experiment

Constraints

Constraints on simulation parameters (are tested before a simulation run):

| Enabled | Expression | Type | Bound |
|---------|------------|------|-------|
|         |            |      |       |

Requirements

Requirements (are tested after a simulation run to determine whether the solution is feasible):

| Enabled | Expression | Type | Bound |
|---------|------------|------|-------|
|         |            |      |       |

Randomness

Random number generation:

Random seed (unique simulation runs)

Fixed seed (reproducible simulation runs) Seed value: 1

Time units: days

# Optimization Requirements – Tests to Sense Validity of Emergent Results

The screenshot displays the AnyLogic Professional interface for a calibration experiment. The main workspace shows a 'Run calibration' button and a 'Calibration progress' graph. The graph shows a red line representing the progress, which is currently at 0.6. The 'Parameters' table below the graph shows the current and best values for various parameters.

|              | Current      | Best         |
|--------------|--------------|--------------|
| Iteration:   | infeasible ? | infeasible ? |
| Replication: | ?            | ?            |
| Objective: ↓ | ?            | ?            |

The 'Calibration - Optimization Experiment' panel is expanded, showing the following sections:

- Constraints:** Constraints on simulation parameters (are tested before a simulation run):
- Requirements:** Requirements (are tested after a simulation run to determine whether the solution is feasible):
- Randomness:** Random number generation:  Random seed (unique simulation runs)  Fixed seed (reproducible simulation runs) Seed value: 1

The bottom status bar shows 'Time units: days'.

# Enabling Multiple Realizations ("Replications", "Runs") per Iteration

The screenshot displays the AnyLogic Professional software interface. The main workspace shows a calibration experiment titled "Calibration - Optimization Experiment". The experiment is configured with the following parameters:

|              | Current      | Best         |
|--------------|--------------|--------------|
| Iteration:   | infeasible ? | infeasible ? |
| Replication: | ?            | ?            |
| Objective:   | ↓ ?          | ?            |

The "Parameters" section is highlighted in yellow. A "Calibration progress" graph is visible, showing a red line representing the progress of the calibration process. The graph has a y-axis ranging from 0 to 1.0 and an x-axis representing iterations. The red line starts at approximately 0.2, rises to 0.6, then drops to 0.4, and finally rises to 0.8.

The "Properties" panel on the right shows the following settings for the "Calibration - Optimization Experiment":

- Constraints
- Requirements
- Randomness
- Replications
  - Use replications
- Window
  - Title: Calibration of SIR Agent Based Mod
  - Width: 800
  - Height: 600
  - Enable panning
  - Maximized size
  - Enable zoom
  - Close confirmation

The "Show Toolbar sections:" option is also visible at the bottom of the Properties panel.

The bottom status bar shows "Time units: days".

# Fixed Number of Replications per Iteration

**Specifies stopping Condition once minimum replications have been run. Indicates that the x% confidence interval around the mean is within “Error percent” of the iteration mean obtained as of the most recent replication**

Properties Progress

Calibration - Optimization Experiment

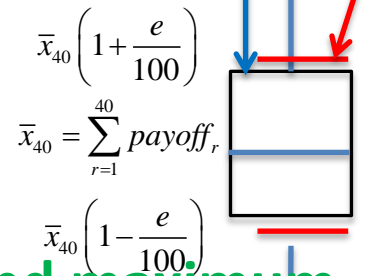
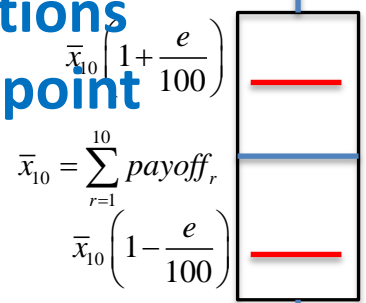
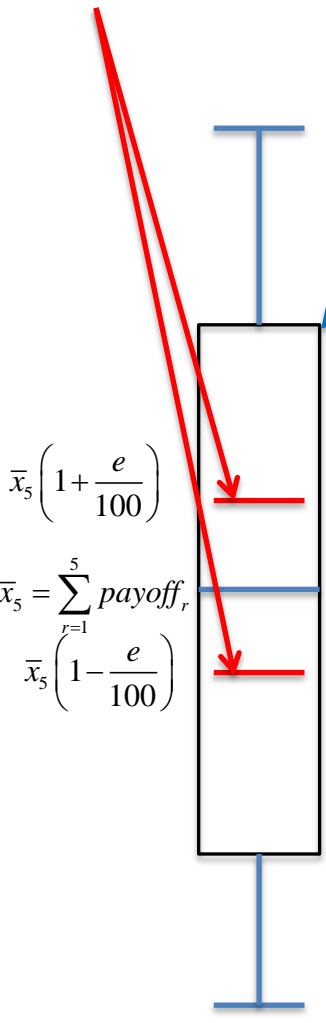
- Constraints
- Requirements
- Randomness
- Replications
  - Use replications
    - Fixed number of replications
      - Replications per iteration: 5
    - Varying number of replications (Stop after minimum replications, when confidence level is reached)
      - Minimum replications: 10
      - Maximum replications: 10
      - Confidence level: 80%
      - Error percent: 0.5
- Window

# Example

Bars showing that delineating values within errorPercent% of mean

Terminates because confidence interval falls within errorPercent% bars

x% (e.g. 80%) confidence interval for sample mean (average) of replications to this point



Minimum and maximum Observed values from replications

After 5 replications

After 10 replications

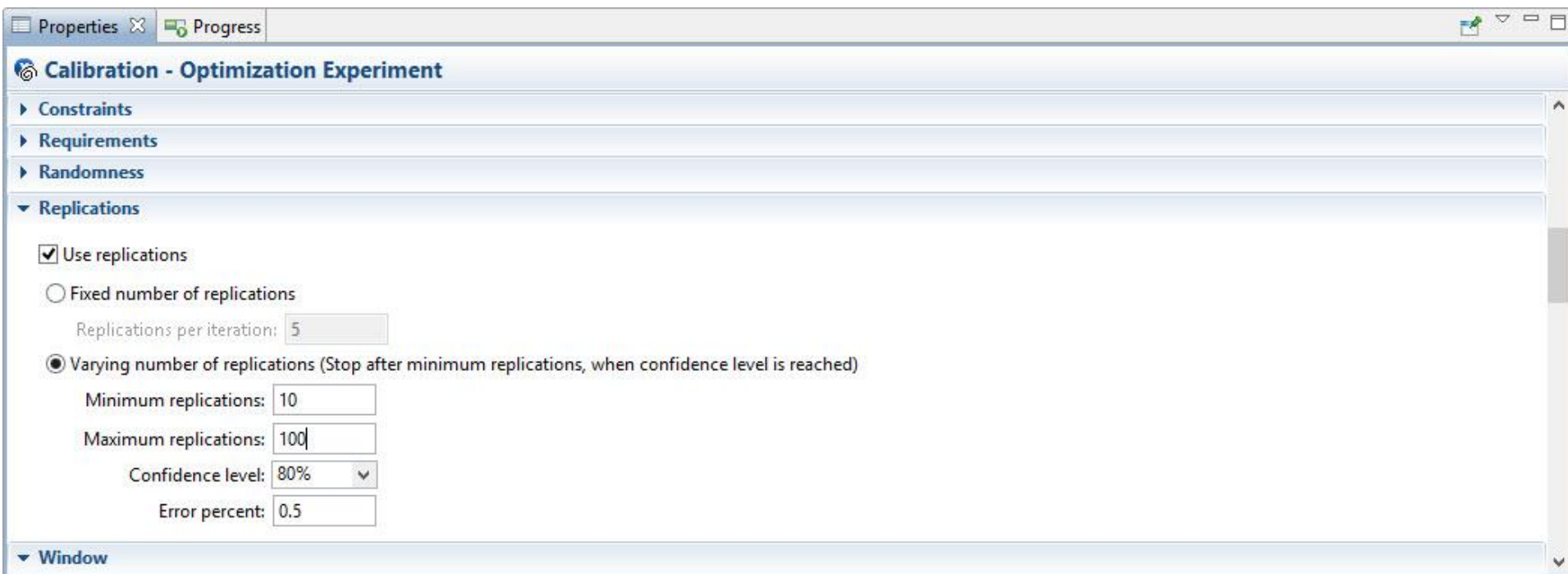
After 40 replications  
**Terminates**

$$\bar{x}_5 \left(1 + \frac{e}{100}\right)$$
$$\bar{x}_5 = \sum_{r=1}^5 \text{payoff}_r$$
$$\bar{x}_5 \left(1 - \frac{e}{100}\right)$$

$$\bar{x}_{10} \left(1 + \frac{e}{100}\right)$$
$$\bar{x}_{10} = \sum_{r=1}^{10} \text{payoff}_r$$
$$\bar{x}_{10} \left(1 - \frac{e}{100}\right)$$

$$\bar{x}_{40} \left(1 + \frac{e}{100}\right)$$
$$\bar{x}_{40} = \sum_{r=1}^{40} \text{payoff}_r$$
$$\bar{x}_{40} \left(1 - \frac{e}{100}\right)$$

# Automatic Throttling of Replications Based on Confidence Intervals for the Average of the Differences between Best and Current



# Enabling Random Variation Between Realizations (“Replications”)

The screenshot displays the AnyLogic Professional interface during a calibration process. The main workspace shows a 'Run calibration' button and a table with the following data:

|              | Current      | Best         |
|--------------|--------------|--------------|
| Iteration:   | infeasible ? | infeasible ? |
| Replication: | ?            | ?            |
| Objective: ↓ | ?            | ?            |

Below the table is a 'Calibration progress' graph with a red line showing the progress over time. The y-axis ranges from 0.6 to 1.0. A 'Parameters' section is visible below the graph.

The Properties panel on the right shows the following settings for the 'Calibration - Optimization Experiment':

- Name: Calibration  Ignore
- Top-level agent: Main
- Objective:  minimize  maximize
- difference ()
- Number of iterations: 500
- Automatic stop
- Maximum available memory: 128 Mb
- Create default UI

The Parameters section is currently collapsed.



# Understanding Replications: Report Results for Each Replication!

The screenshot displays the AnyLogic Professional interface for a calibration experiment. The main workspace shows a 'Run calibration' button and a table with the following data:

|              | Current      | Best         |
|--------------|--------------|--------------|
| Iteration:   | infeasible ? | infeasible ? |
| Replication: | ?            | ?            |
| Objective: ↓ | ?            | ?            |

Below the table is a 'Parameters' section. To the right, a 'Calibration progress' graph shows a red line fluctuating between 0.6 and 1.0 on the y-axis. The bottom panel, titled 'Calibration - Optimization Experiment', contains the following Java actions:

```
Initial experiment setup:  
dsInfectiousHistoric.fillFrom( InfectiousHistoric );  
  
Before each experiment run:  
datasetCurrentObjective.reset ();  
datasetBestFeasibleObjective.reset ();  
  
Before simulation run:  
  
After simulation run:  
dsInfectiousCurrent.fillFrom( root.InfectiousDS );  
traceln("For this particular simulation, the difference is\t" + difference());  
  
After iteration:  
if( getCurrentIteration() == getBestIteration() )  
    dsInfectiousBest.fillFrom( dsInfectiousCurrent );
```

The bottom right corner of the interface shows 'Time units: days'.

# During First Several Realizations (“Replications”, “Runs”), No Results Appear

Calibration of SIR Agent Based Model - AnyLogic Professional

Run calibration

|              | Current | Best |
|--------------|---------|------|
| Iteration:   |         |      |
| Replication: |         |      |
| Objective:   | ↓       |      |

**Parameters**

ContactRate  
InfectionProbability

Copy the best solution to the clipboard

The built-in OptQuest optimizer is used to calibrate an agent based model of contagious disease diffusion. In the model each person has 3 possible states: Susceptible, Infectious and Recovered (SIR). Initially all but few people are susceptible, and few - infectious. A person can contact another person, and in case one is susceptible and another - infectious, the first may get infected with a certain probability. The objective is to find the parameters of the agents (contact frequencies and infection probabilities) so that the output of the simulation model fits best with the historical data - the dynamics of infectious population (disease prevalence). As the model is stochastic, the calibration is done under uncertainty, and simulation replications are used. A separate 1st order Monte Carlo experiment is included to demonstrate the output of the agent based model.

### Calibration progress

Legend: == Current objective, — Best feasible objective

### Historic data, best fitting and current simulation output

Legend: — Historic data, — Current output, — Best output

Run: 0 Idle | Experiment: 0% | Simulations: Stop time not set | 0.0 sec

# Report on Iteration 2 Appears after a Count of Runs Equal to Replications per Iteration

Reports best payoff (objective) yet reached (lower is better), but from where did this number come?

**Calibration of Agent Based SIR Model**

Run calibration

|              | Current | Best     |
|--------------|---------|----------|
| Iteration:   | 9       | 2        |
| Replication: | 10      |          |
| Objective: ↓ | 118,555 | 74,976.6 |

**Parameters**

|                      |       |     |
|----------------------|-------|-----|
| ContactRate          | 1.713 | 0.5 |
| InfectionProbability | 0.107 | 0.1 |

Copy the best solution to the clipboard

The built-in OptQuest optimizer is used to calibrate an agent based model of contagious disease diffusion. In the model each person has 3 possible states: Susceptible, Infectious and Recovered (SIR). Initially all but few people are susceptible, and few - infectious. A person can contact another person, and in case one is susceptible and another - infectious, the first may get infected with a certain probability. The objective is to find the parameters of the agents (contact frequencies and infection probabilities) so that the output of the simulation model fits best with the historical data - the dynamics of infectious population (disease prevalence). As the model is stochastic, the calibration is done under uncertainty, and simulation replications are used. A separate 1st order Monte Carlo experiment is included to demonstrate the output of the agent based model.

**Calibration progress**

Legend: ● Current objective, — Best feasible objective

**Historic data, best fitting and current simulation output**

Legend: — Historic data, — Current output, — Best output

Run: 47 Paused Experiment: 0% Simulations: Stop time not set 6.5 sec

# Output

The screenshot displays the AnyLogic Professional interface. The main workspace shows a calibration progress chart and a table of results. The table has columns for 'Current' and 'Best' values. The 'Current' column shows 'infeasible' for both 'Iteration' and 'Replication', with a red arrow pointing to the 'Objective' row. The 'Best' column shows '?' for all rows. A 'Run calibration' button is visible above the table. To the right, a 'Calibration progress' chart shows a red line fluctuating between 0.6 and 1.0. The console window at the bottom shows the output of the simulation, with a red box highlighting a list of difference values and a red arrow pointing to the 11th value, 74987.

|              | Current      | Best         |
|--------------|--------------|--------------|
| Iteration:   | infeasible ? | infeasible ? |
| Replication: | infeasible ? | infeasible ? |
| Objective:   | ↓ ?          | ? ?          |

Calibration progress

anylogic config [Java Application] D:\Program Files\AnyLogic 7 Professional\jre\bin\javaw.exe (2014-04-11 8:15:11 PM)

```
For this particular simulation, the difference is 17694
For this particular simulation, the difference is 74984
For this particular simulation, the difference is 113475
For this particular simulation, the difference is 74986
For this particular simulation, the difference is 4118
For this particular simulation, the difference is 74986
For this particular simulation, the difference is 61290
For this particular simulation, the difference is 74987
For this particular simulation, the difference is 81885
For this particular simulation, the difference is 73328
For this particular simulation, the difference is 118555
For this particular simulation, the difference is 119492
For this particular simulation, the difference is 74898
For this particular simulation, the difference is 2038
For this particular simulation, the difference is 31451
For this particular simulation, the difference is 59201
For this particular simulation, the difference is 64188
```

The reported payoff for the iteration is the average of the payoffs for each replication *within* the replication



# Considerations

- Adding constraints helps increase identifiability (selection of realistic best fit)
- Adding parameters to tune leads to larger space to explore
- Adding too many parameters to tune can lead to underdetermined situation
  - Use non-dimensionalization to reduce parameter count
- All fits are within constraints of model

# Dealing with Calibration Problems: Experiments

- Try to “outsmart” calibration
  - Adopt best parameter values from calibration
  - Try to adjust parameters to do better than calibration
    - If is better, it may be that the parameter space is too large, or that the range constraints are too tight
    - Typically this does not do as well: Opportunity to learn
      - Model not respond in the way that anticipated to parameter change
      - May just shift the discrepancy from one variable to another
        - » Assumptions of model structure/values may not permit both variables to simultaneously match well!
- Set very high weight on thing that want to match, and see other matches
- Set all other weights to 0 (see if can possibly match)

# Dealing with Calibration Problems: Additional Experiments

- Increase parameter range
- Increase # of parameters
- Examine impact of changed model structure
- Run for larger number of optimization runs
- Find other estimates for uncertain parameters



# Important Cross-Checks: Uniqueness

- Are the calibration values unique? If so, good; if not,
  - Do they give the same underlying interpretation?
  - Do the different interpretations lead to parameters that “trade off” in some structured way?
- Ways of addressing significantly different interpretations
  - Collect more primary data!
  - Impose additional constraints (in terms of time series, etc.)
  - Simplify model
  - Find other estimates for uncertain parameters

# Important Cross-Checks: Binding Constants

- Look for calibrated parameter values that are at the edges of their permissible ranges
  - If “best” value is at the edge of the range, it may be that even better calibrations would have been possible if continuing in that direction
- To deal with those at the edge
  - Relax constraints
  - Collect more data on plausible values
  - Question model structure

# Capturing Parameter Interdependencies in Calibration

- If we want parameter B adjusted during calibration to be at least as big as parameter A
  - In vensim, we can't enforce this constraint using the typical calibration machinery, because the range limits for parameters must be constants
  - we can accomplish this by calibrating only parameter A, and a parameter representing the ratio B/A.
- If we want to adjust two or more parameters such that they still sum to 1 (e.g. fraction of initial population in each of  $n$  or more stocks), we can adjust each of  $n$  non-normalized weights, and then take the corresponding normalized amount to be frac. falling in that category

# Calibrating Initial Conditions

- The initial conditions can be one of the best values to calibrate
- Sometimes need to divide a fixed population into several stocks

# Calibration & Regression: Similarities & Differences

- Model calibration is similar to regression in that we are seeking to find the parameter values allowing the best match of model & data
  - As in non-linear regression, for non-linear simulation models no “closed form” solution of best parameter values is possible  $\Rightarrow$  optimization is required
- A big difference:
  - **Regression models:** the “functional form” (dependence of model output on par’ms/indep vars) is given explicitly
  - **Simulation models:** behavior is only *implicitly* specified (e.g. via giving differentials); model output is a complex resultant (even emergent) property of structure