



VENTITY

**Entity-based
System Dynamics
modelling
Intro & Model Reuse**

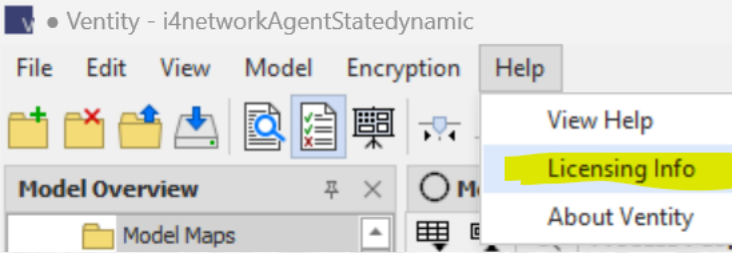
#ISDC 2024

Logistics

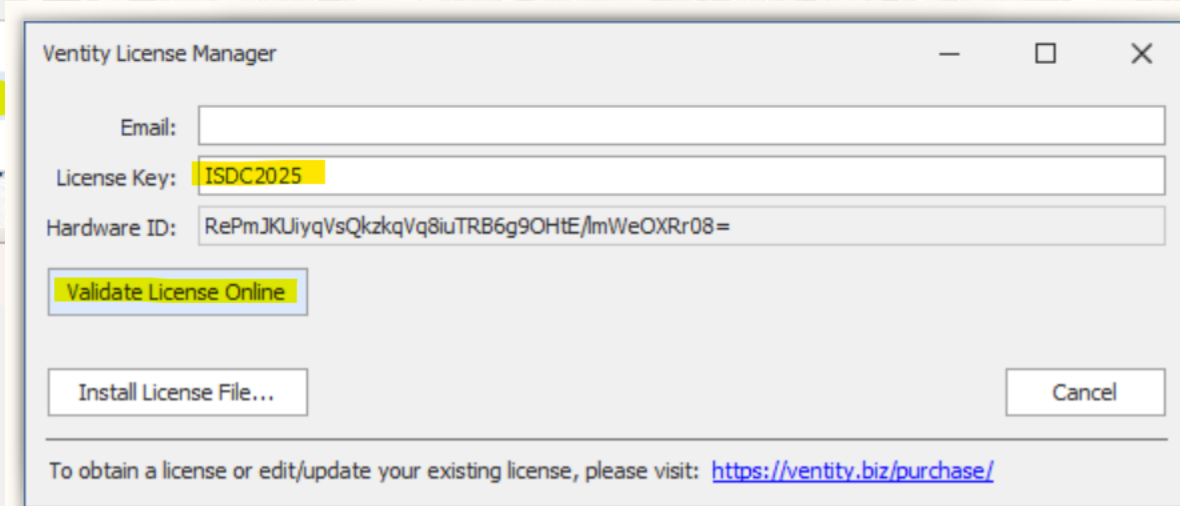
- Download the course material folder and unzip to a convenient location
 - Source: <http://vensim.com/conference>
 - Choose a location you control, i.e. not locked by your operating system
 - Unzip
- Install Ventity 5 from <http://Ventity.biz/download>
 - Licenses are unlocked through the end of August using **ISDC2025** as the license key
 - If you have an existing v4.5 or later, that should be fine too

License Installation

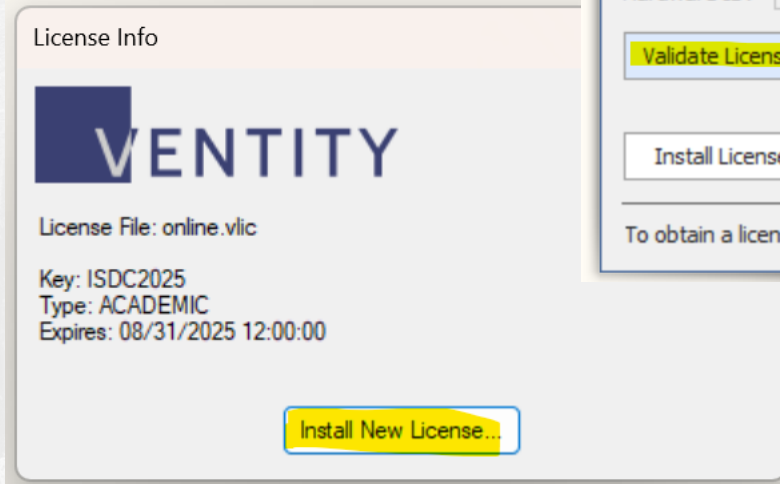
1



3



2



Abstract

- Ventity is a companion to Vensim that provides an elegant way to blend classic system dynamics with discrete event and agent-based modeling. Features facilitate modularity and collaboration, provide a more natural description of detail than arrays, and solve sparse matrix problems.
- This is a hands-on introduction to Ventity. We will introduce the software and new concepts, including collections of entities, attributes, relationships, aggregation and allocation functions, and actions. Our testbed will be a classic SD logistics model similar to the beer game, but we'll try variants with discrete events and agents, and if time permits, networks, all in the same model. This will shed light on key questions: when and how should you aggregate, and what are the tradeoffs between depth and breadth?

Inspiration

<https://metasd.com/2017/03/dynamics-of-the-last-twinkie/>

<https://metasd.com/2010/09/sd-build-bridges/>



MetaSD

Don't just do something, stand there! Reflections on the counterintuitive behavior of complex systems, seen through the eyes of system dynamics.

[Model Library](#) [About](#)

Dynamics of the last Twinkie

Torn

March 6, 2017

Business, Stochastic, Toy

Model, Verity

[Edit](#)

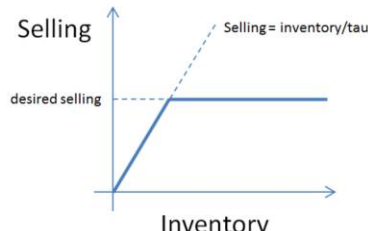
When Hostess went bankrupt in 2012, there was lots of speculation about the fate of the last Twinkie, perhaps languishing on the dusty shelves of a gas station convenience store somewhere in New Mexico. Would that take ten days, ten weeks, ten years?

So, what does this have to do with system dynamics? It calls to mind the problem of modeling the inventory stockout constraint on sales. This problem dates back to [Industrial Dynamics](#) (see the variable NIR driving SSR and the discussion around figs. 15-5 and 15-7).

If there's just one product in one inventory (i.e. one store), and visibility doesn't matter, the constraint is pretty simple. As long as there's one item left, sales or shipments can proceed. The constraint then is:

$(I) \text{ selling} = \text{MIN}(\text{desired selling}, \text{inventory}/\text{time step})$

In other words, the most that can be sold in one time step is the amount of inventory that's actually on hand. Generically, the constraint looks like this:



Goals

Mechanics

- Environment
- Diagramming
- Equations
- Units
- Run control
- Run naming
- Entity initialization data
- Charts/tables
- Slider controls
- Entity picker
- Entity imports

Modeling Concepts

- Model-data separation
- Entities
- Attributes
- References
- Collections & aggregates
- Actions & triggers

Ventity concepts

- **Entitytype/Entity** – related structure sharing the same level of detail, often representing a sector or agent; analogous to classes and instances in OOP, or tables and rows in relational databases
- **Attribute** – a text tag that may contain a reference, or just categorical data
- **Reference** – a pointer to another entity, often defined by an attribute
- **Collection/Subcollection** – a set or subset of entities of a given type, containing aggregates for variable values of members
- **Action** – a discrete event between time steps that changes system structure or state, dispatched by a **trigger**

Scenario

- You're modeling a hospital
- You're interested in load management
- Patients arrive with a variety of disease/injury severities, characterized by their expected time to recover
- Patients are discharged after their recovery time elapses
- Time horizon: 120 days

Sequence

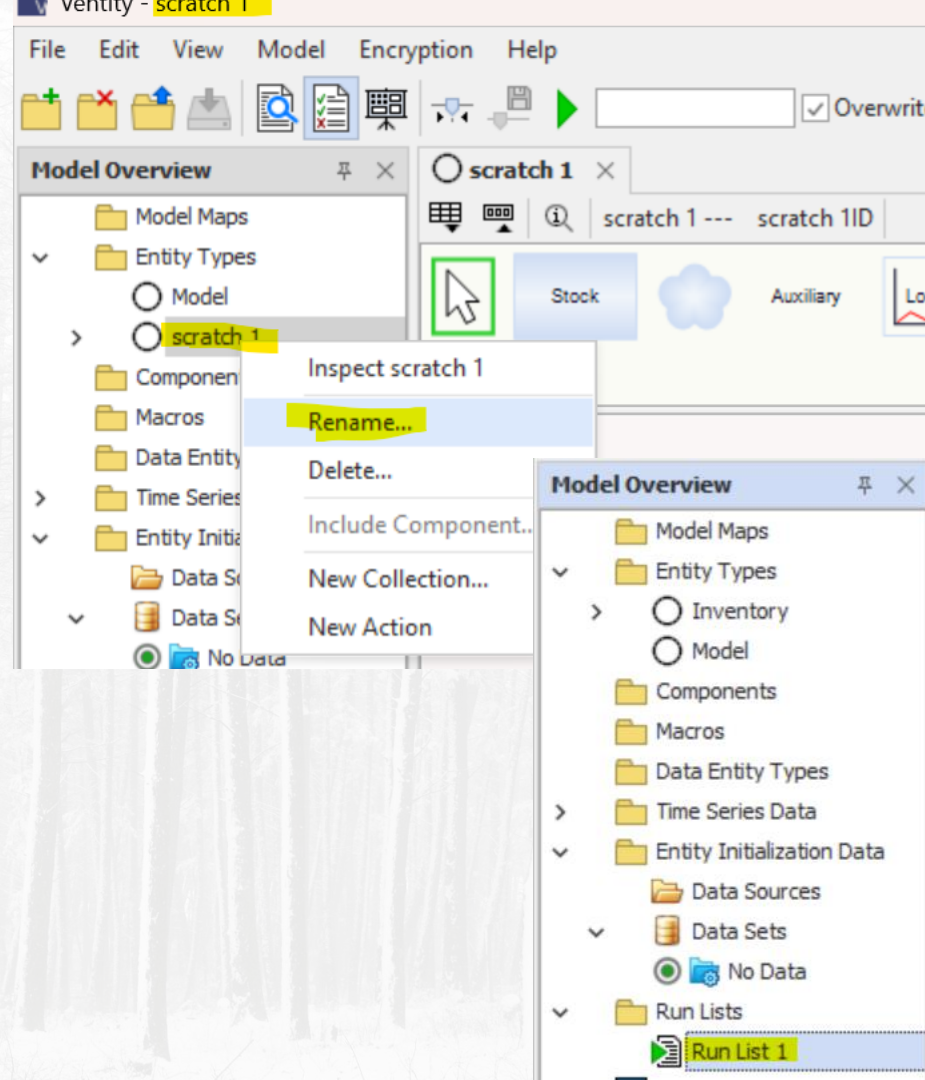
- Model in detail, with patients as agents
- Create an aggregate, continuous first-order stock representation
- Test n aggregate first-order model with discrete stochastic dynamics
- When would we prefer each version?

Create a New Model

- End the filename with a number, like “Patients 1”, and Ventity will increment it automatically when you “Save As...”
- For ease of reference, save the model near the course materials you downloaded

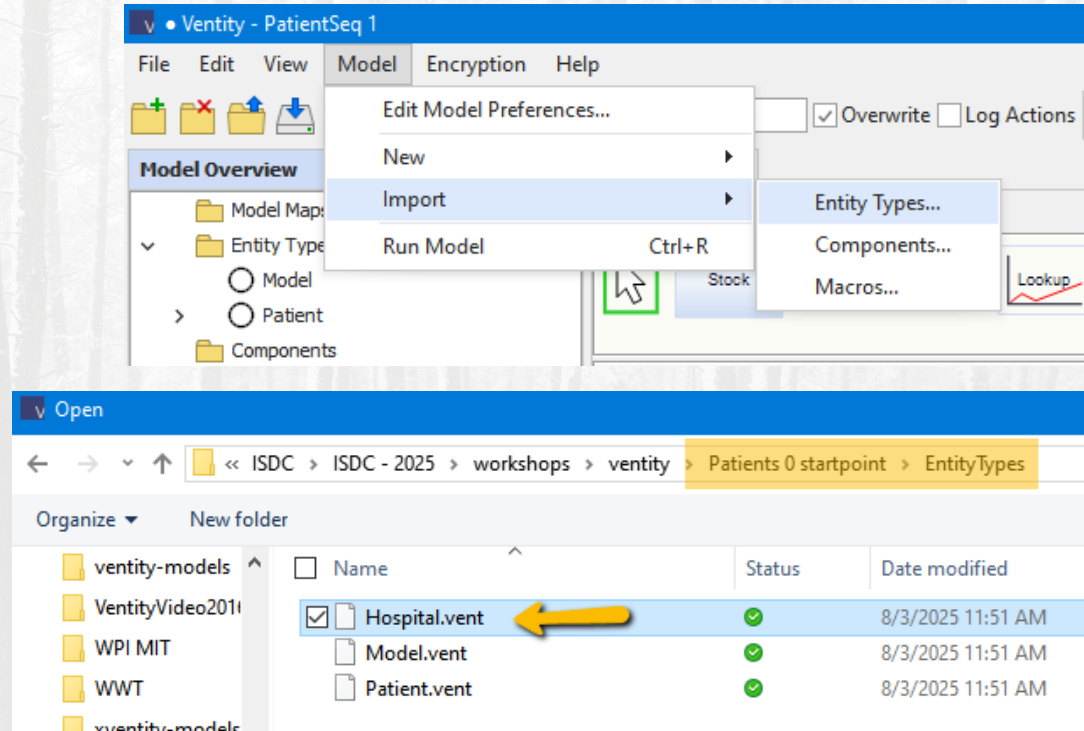
Take a look around ...

- Ventity created a default entitytype with a name matching the model – rename that to “**Patient**” – we’ll use it soon
- Take a look at the time range in the **Run List** (we’ll work with the defaults for now)

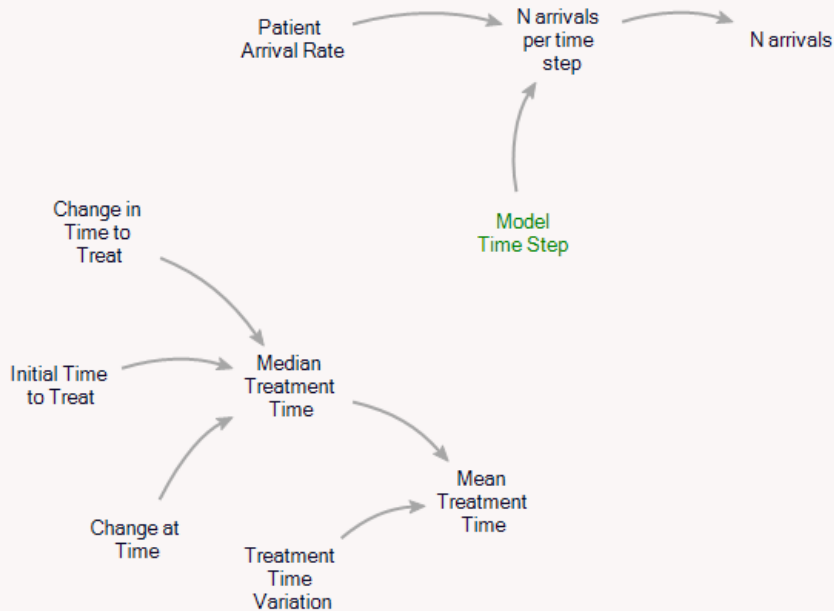


To speed things up, import a Hospital entity

- Import the **Hospital** entity from the **Patient 0 startpoint** in the materials





Take a look around the Hospital you just imported ...

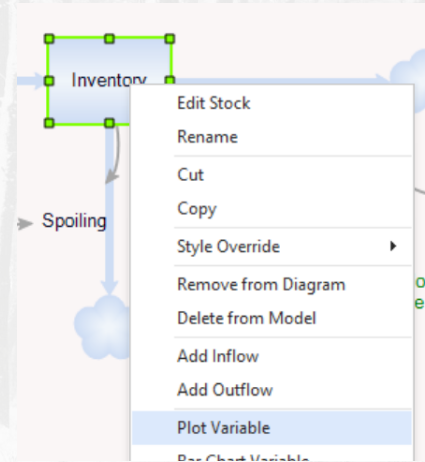


- Randomized arrival process
- Mean expected treatment time

Run & view

- Hit the run button 
- Errors appear in the error list (normally docked at the bottom)
- Right-click variables to plot or view data

| Error List | |
|---|---|
| Source | Description |
|  | Auxiliary "Shelf Life" Missing expression |

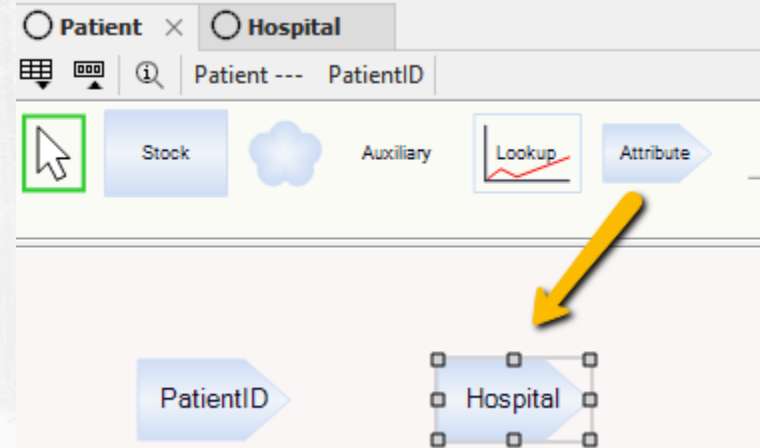
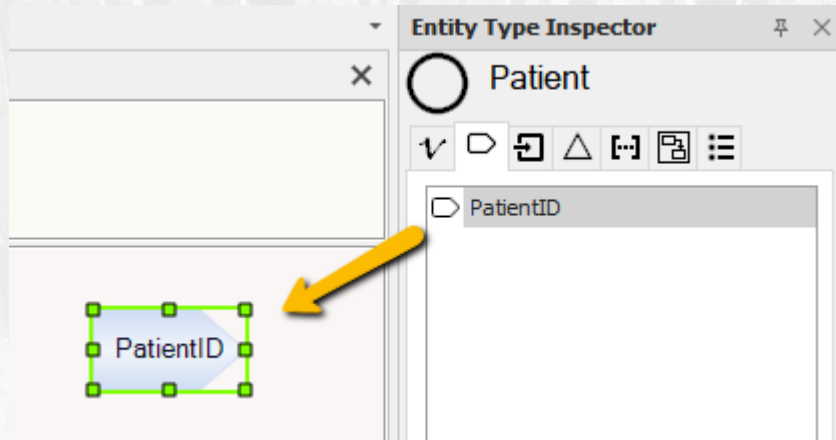


Waypoint

- Before moving ahead, Save As...
- Model so far is saved in **PatientSeq 1.vmdl**

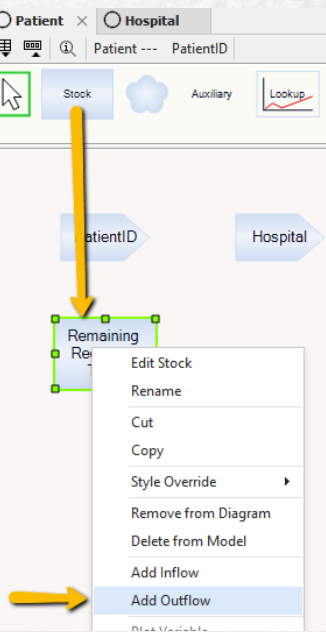
Prepare the Patient agent

- Double-click your **Patient** entitytype to edit
- Drag the **PatientID** attribute from the inspector onto the diagram
- Add a new **Hospital** attribute from the toolbar



Add recovery dynamics

- Drag a **Remaining Recovery Time** stock from the toolbar
- Right-click to add an outflow
- Define the stock initial = 10 days (this will be overridden)
- Define the flow as 1 day/day



Stock - Patient.Remaining Recovery Time

Name: Remaining Recovery Time

Owner: Patient

Units: days

☐ Enable causes glyph on node

Default
Initial
Value: 10

Flow - Patient.Recovering

Name: Recovering

Owner: Patient

Units: day/day

Source: Remaining Recovery Time

Destination:

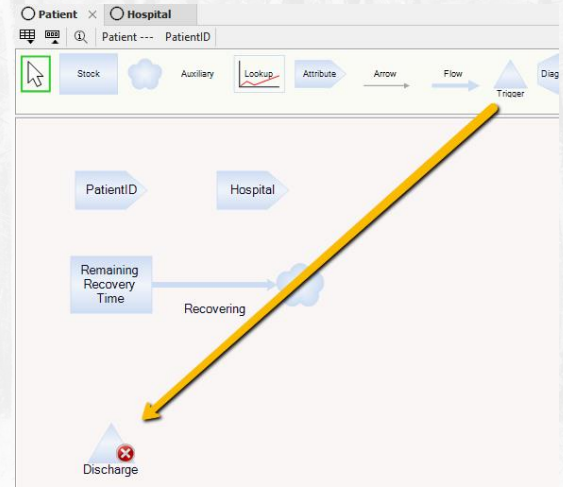
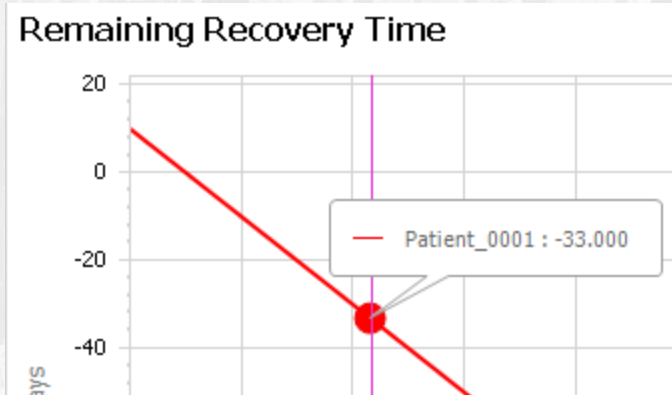
☐ Constant

☐ Enable causes glyph on node

1

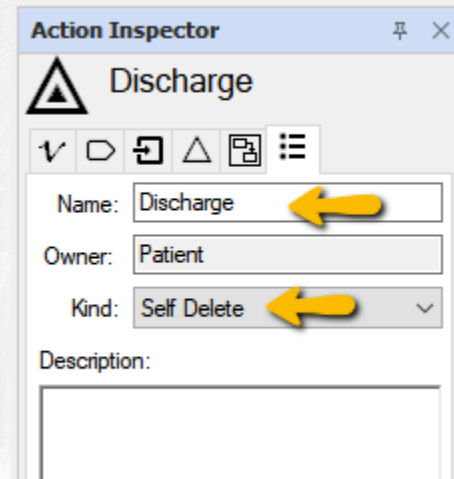
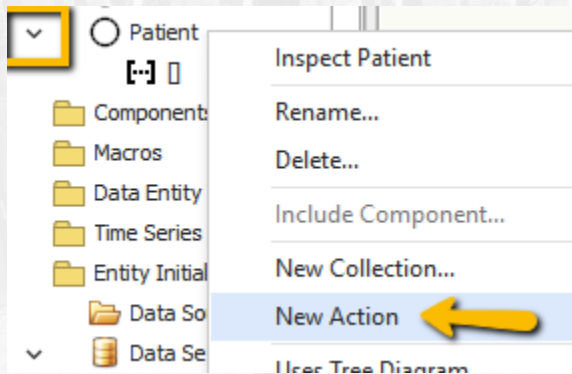
Run a test

- You'll probably see the patient's recovery time go negative (no feedback), but we'll fix that ...
- Add a **Discharge** trigger to the diagram



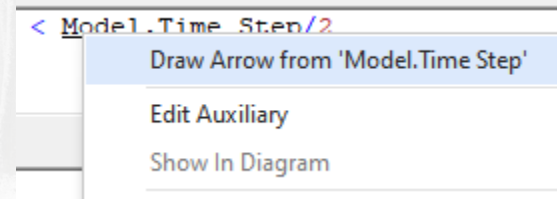
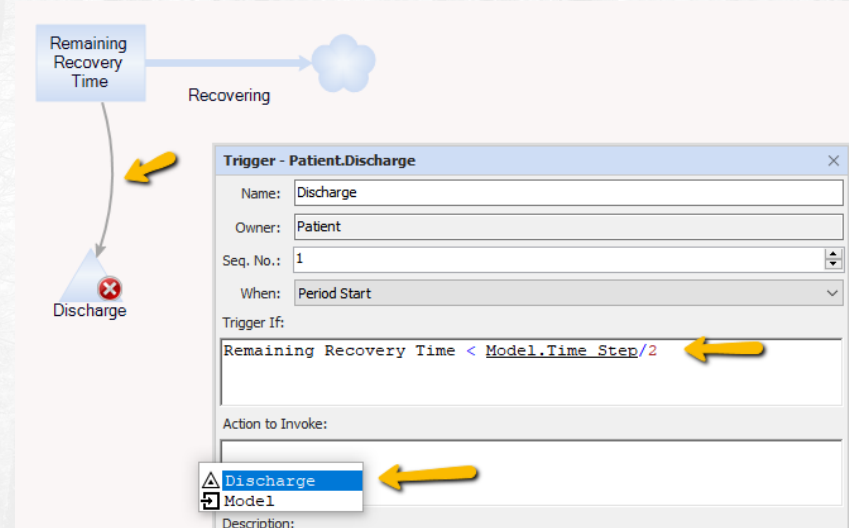
Create a Discharge action

- Expand the **Patient** entity
- Right-click the **Patient** to create a **New Action**
- Name the action and set its effect to **Self Delete**



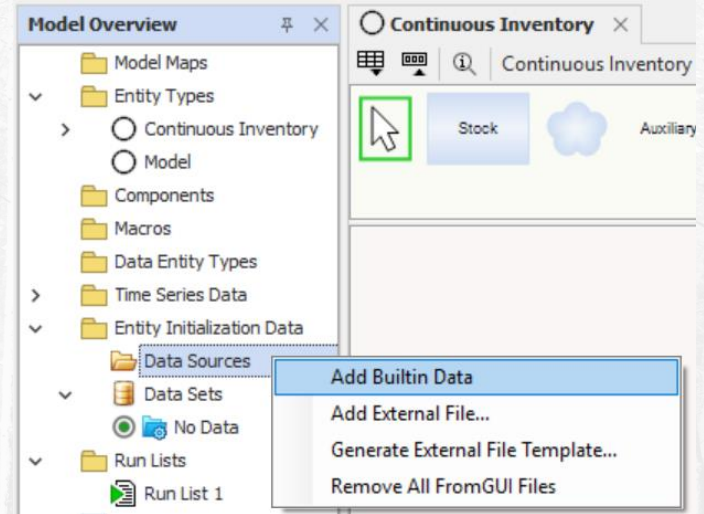
Return to the Patient entity

- Connect **Remaining Recovery Time** to your **Discharge** trigger
- Right-click to edit the trigger
- Set the “Trigger If” condition
- Set the “Action to Invoke”
- Right-click **Model.Time Step** in the equation to draw the needed arrow











Create initialization data

- We want to create multiple **Patients**.
- Right-click **Data Sources** in the Entity Initialization Data
- **Add Builtin Data**



Edit the initialization data

Entities 1

Number of rows to add:         All None Invert ☒ Show All Types

| Hospital | | | Model | | | | | Patient | |
|----------|-------------------------------------|------|--------------|--------------|----------------|-------------------------|-----------------------|----------------------|--------------------------|
| | <input checked="" type="checkbox"/> | Time | CalendarTime | HospitalID | Change at Time | Change in Time to Treat | Initial Time to Treat | Patient Arrival Rate | Treatment Time Variation |
| ▶ | <input checked="" type="checkbox"/> | | | Mass General | | | | | |
| ✶ | <input type="checkbox"/> | | | | | | | | |

Time of entity injection
(default = start time)

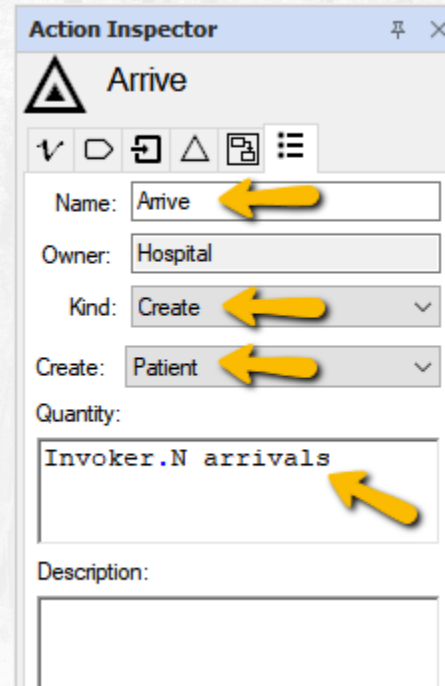
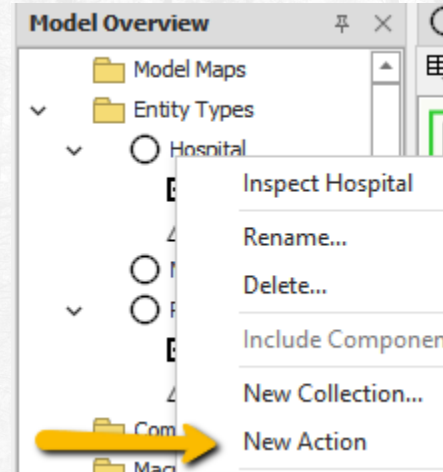
Attribute
containing the
entity key ID

Initial Stocks and
Constants
(default =
equation value)

- We can leave most of the fields blank (= uses equation defaults), and we don't need Patients

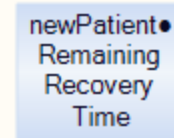
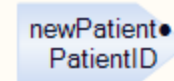
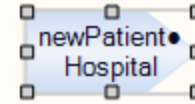
Use an action to create arriving Patients

- Create a new action in the **Hospital**
- Name it **Arrive**
- Set it to **Create** new Patient entities
- Set the Quantity to **Invoker.N arrivals**



Initialize the Patients created

- The new entity contains surrogates that point to the states of the **Patient**
- These can be left blank or used to override the Patient's default values



Initialize the Patient states

- Again, we'll use the **Invoker** to reference values in the **Hospital**
- You can right-click to add arrows

Attribute - Arrive.newPatient•Hospital

Name:

Owner:

Refers to:

☐ Key

New Value:

Stock - Arrive.newPatient•Remaining Recovery Time

Name:

Owner:


Units:

☐ Enable causes glyph on node

Default Initial Value:

Connect the trigger

- Return to the **Hospital**
- Connect **N arrivals** to the trigger
- Set the Action to Invoke to your new **Arrive**

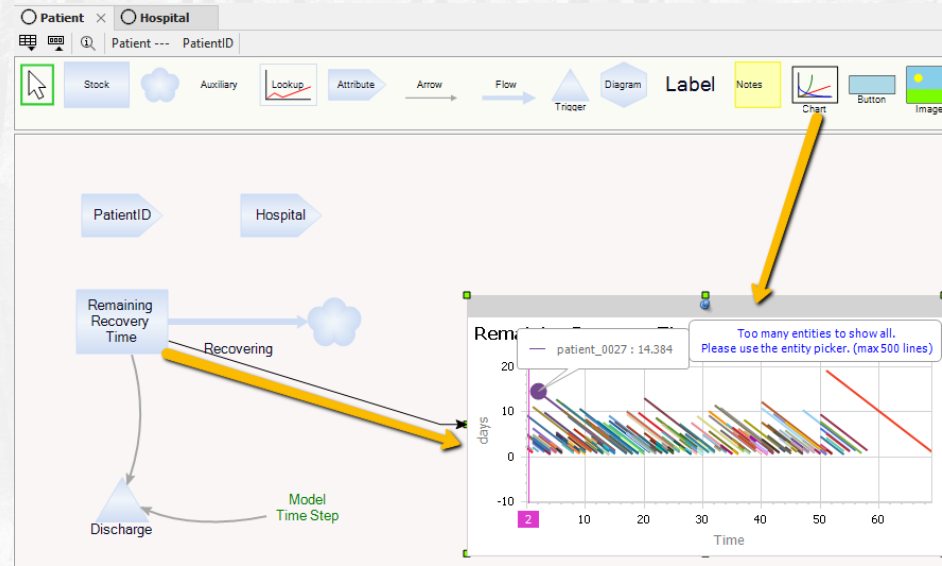


The diagram shows a curved arrow pointing from the text 'N arrivals' to a trigger icon. The icon is a blue triangle with a red 'X' inside, and the word 'Arrive' is written below it.

| Trigger - Hospital.Arrive | |
|---------------------------|----------------|
| Name: | Arrive |
| Owner: | Hospital |
| Seq. No.: | 1 |
| When: | Period Start |
| Trigger If: | N arrivals > 0 |
| Action to Invoke: | Arrive |

Run and plot

- Switch to the **Patient**
- Drag a chart onto the diagram
- Drag to connect **Remaining Recovery Time** to the chart

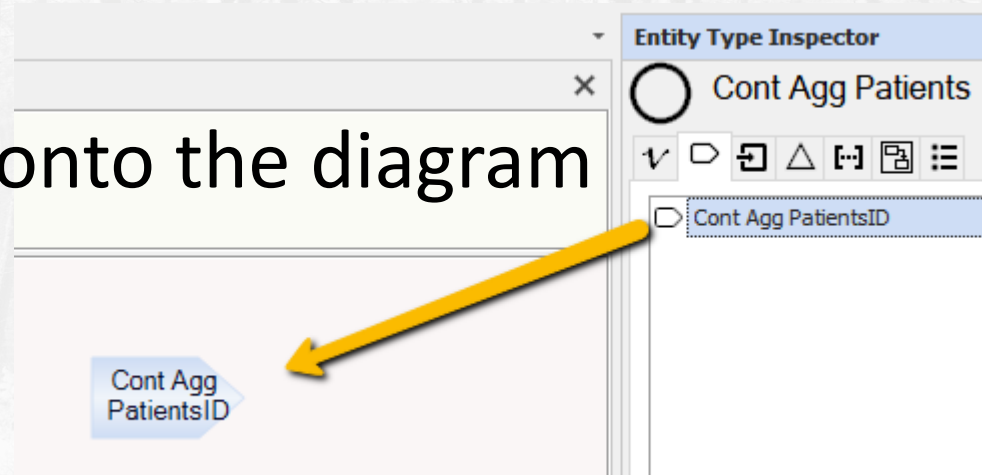


Waypoint

- Before moving ahead, Save As...
- Model so far is saved in PatientSeq 2.vmdl

Create an aggregate patient entity (classic continuous SD)

- Right-click the Entity Types folder to create a new entity type
- Name it something like “Continuous Aggregate Patients”
- Drag its key attribute onto the diagram



Create a reference

- Rename the key attribute

- Create a **Hospital** reference in the inspector and point it to the **Hospital** entity type

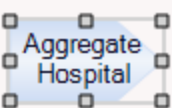


Diagram showing an **Aggregate Hospital** entity (represented by a blue box with a cloud icon) connected to the **Attribute - Cont Agg Patients.Aggregate Hospital** form.

Attribute - Cont Agg Patients.Aggregate Hospital

Name:

Owner:

Refers to:

☒ Key

Initial Value:

Entity Type Inspector

Cont Agg Patients

> Hospital <Hospital> ←

> Model <Model>

Reference - Cont Agg Patients.Hospital

Name: ←

Owner:

Target Type:

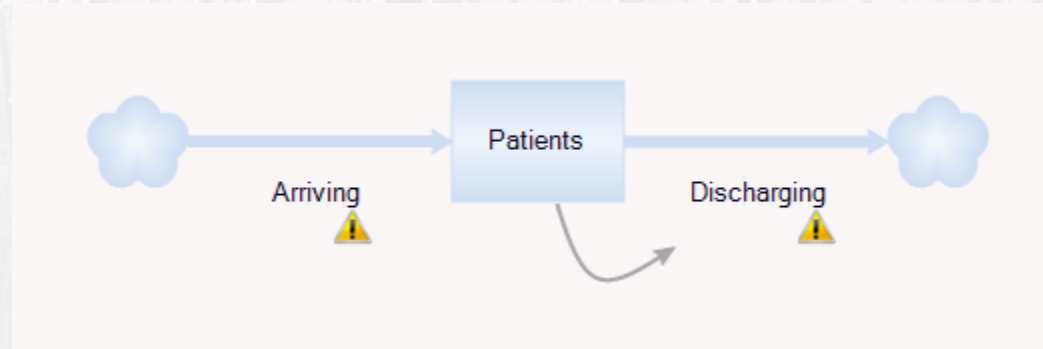
| Conditions: | Value to Match |
|---------------|--------------------|
| Attribute Key | |
| HospitalID | Aggregate Hospital |

Description:

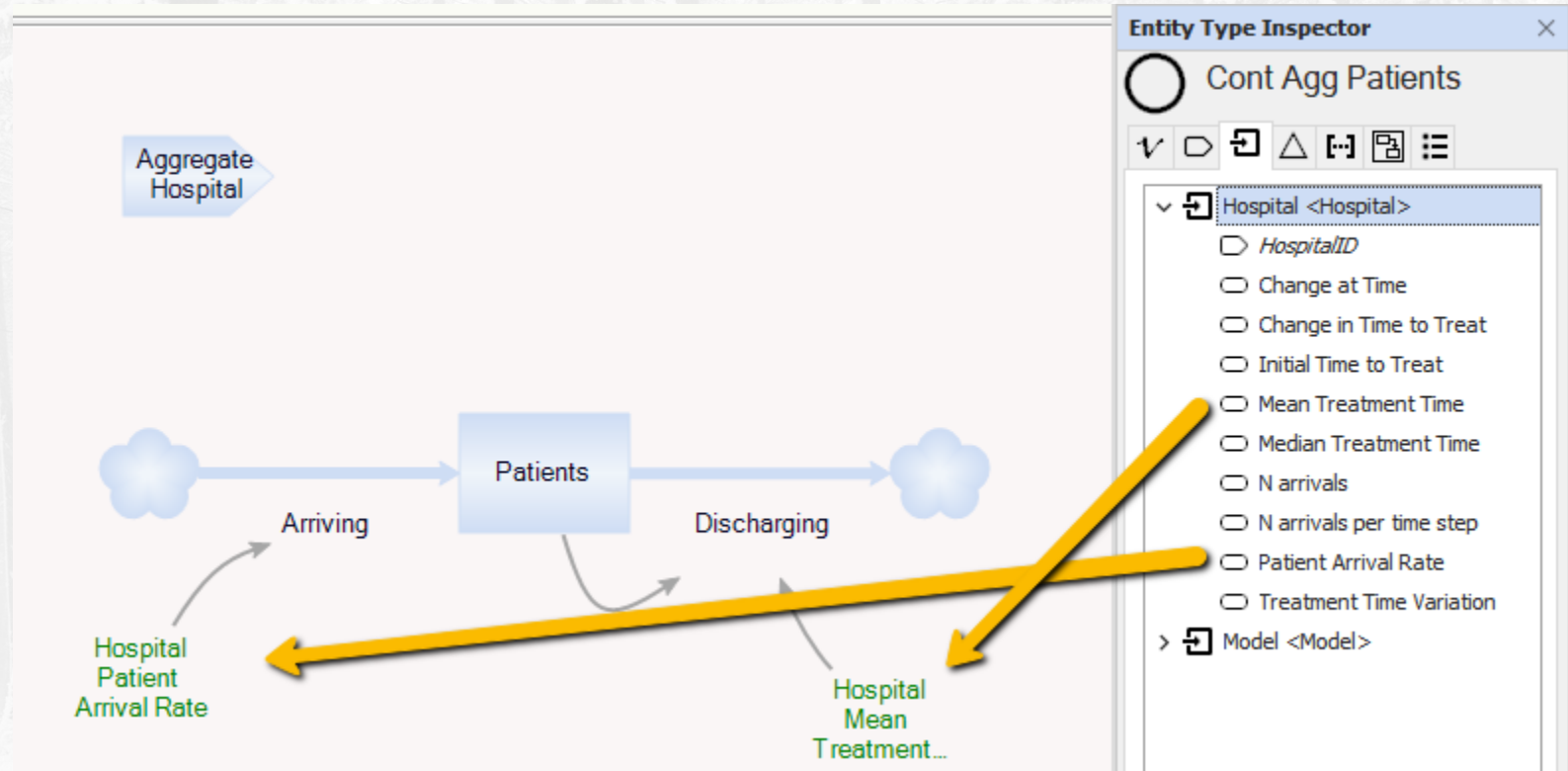
Cont Agg Patients ←

Draw a stock-flow structure

- Hint: right-clicking the stock to add inflows and outflows is quick

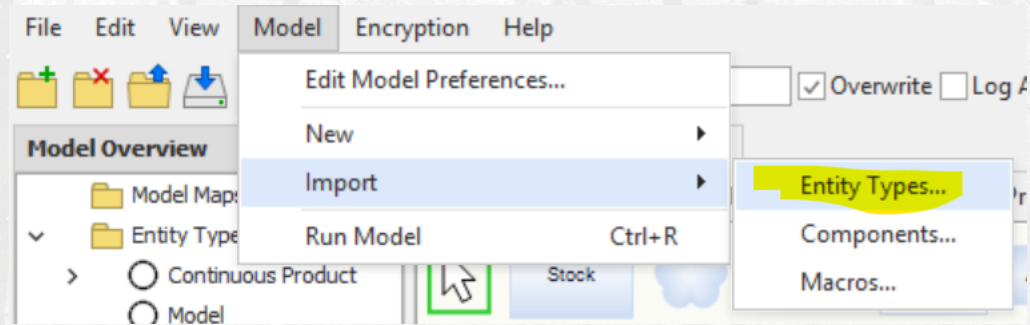


Add equations referencing the Hospital



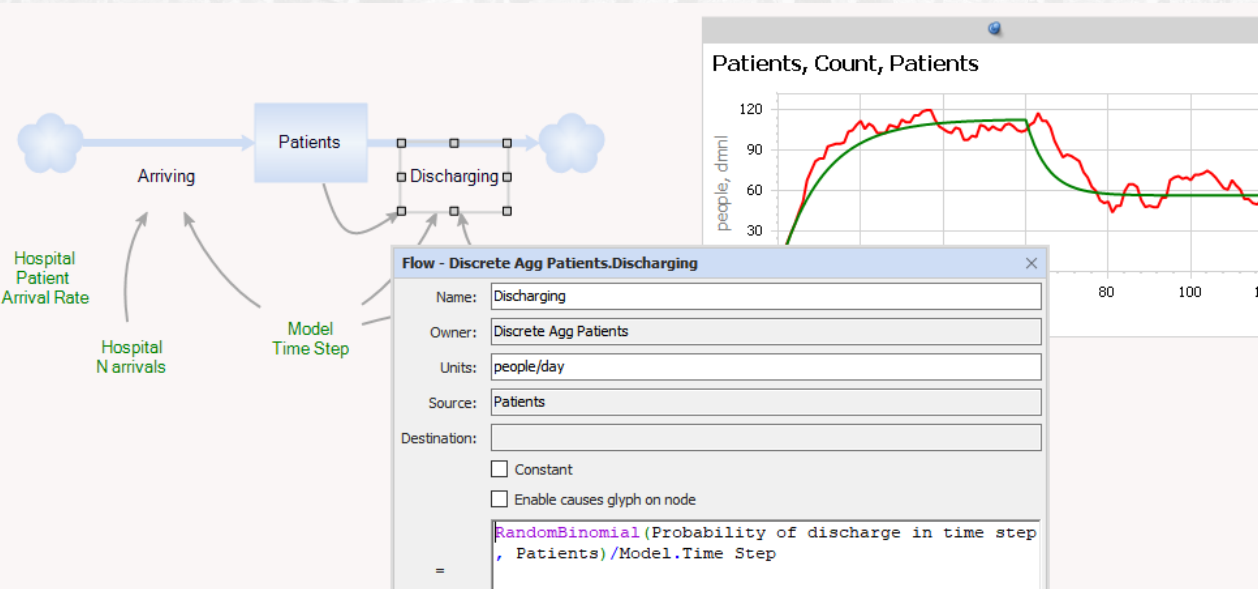
Import a discrete random aggregate patient stock

- Import the **Discrete Agg Patient** entity from the **Patients 4** model in the materials



What's different?

- Flows use discrete random values, so the **Patients** stock respects integer units



Results: Run 1 - Component: Discrete Agg Patients - Elem

Drag a column header here to group by that column

| | DateTime | Run | Aggregate Hospital | Patients |
|---|----------|-------|--------------------|----------|
| ▶ | 0 | Run 1 | Mass General | 0 |
| | 1 | Run 1 | Mass General | 3 |
| | 2 | Run 1 | Mass General | 14 |
| | 3 | Run 1 | Mass General | 24 |
| | 4 | Run 1 | Mass General | 27 |
| | 5 | Run 1 | Mass General | 33 |
| | 6 | Run 1 | Mass General | 42 |
| | 7 | Run 1 | Mass General | 50 |
| | 8 | Run 1 | Mass General | 59 |
| | 9 | Run 1 | Mass General | 61 |
| | 10 | Run 1 | Mass General | 68 |
| | 11 | Run 1 | Mass General | 75 |

Update your initialization data

- Delete your initialization data and replace with a fresh copy
- Be sure names match

The image shows two screenshots of a software interface, likely a data management tool, illustrating the process of updating initialization data. A red arrow points from the 'Mass General' entry in the top table to the 'Mass General' entry in the bottom table, highlighting the importance of matching names.

Top Screenshot:

Entities 1

Number of rows to add: [All](#) [None](#) [Invert](#) ☒ Show All Types

| Cont Agg Patients | | | | Hospital | | M |
|-------------------|-------------------------------------|------|--------------|--------------------|----------|---|
| | <input checked="" type="checkbox"/> | Time | CalendarTime | Aggregate Hospital | Patients | |
| ▶ | <input checked="" type="checkbox"/> | | | Mass General | | |
| ★ | <input type="checkbox"/> | | | | | |

Bottom Screenshot:

Entities 1

Number of rows to add: [All](#) [None](#) [Invert](#) ☒ Show All Types

| Hospital | | | | Cont Agg Patients | | | | Mod |
|----------|-------------------------------------|------|--------------|-------------------|----------------|-------------------------|-----------------------|-----|
| | <input checked="" type="checkbox"/> | Time | CalendarTime | Hospital | Change at Time | Change in Time to Treat | Initial Time to Treat | |
| ▶ | <input checked="" type="checkbox"/> | | | Mass General | | | | |
| ★ | <input type="checkbox"/> | | | | | | | |

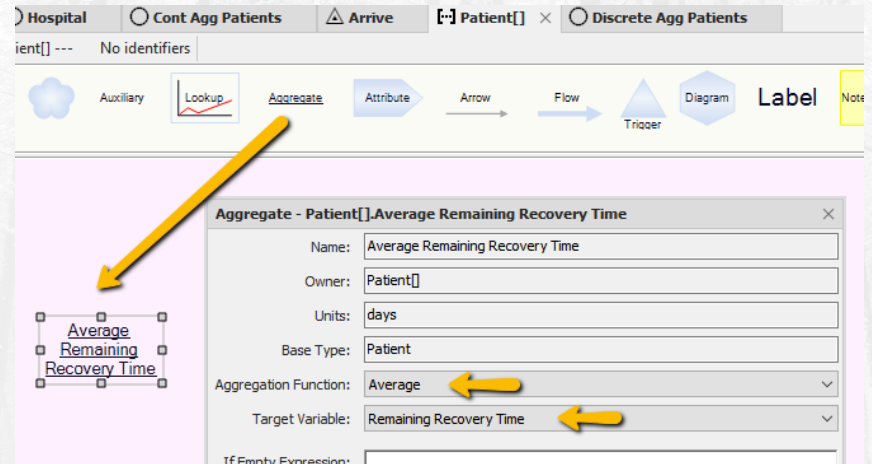
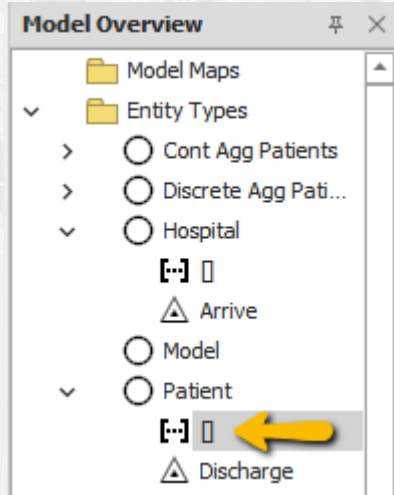
Waypoint

- Before moving ahead, Save As...
- Model so far is saved in PatientSeq 3.vmdl

Count and average agent patients

Open the top [] collection for Patients

Add an aggregate to average



Add controls

Don't see your parameter ? Be sure it's marked "Constant"

Auxiliary - Hospital.Change in Time to Treat

Name: Change in Time to Treat

Owner: Hospital

Units: days

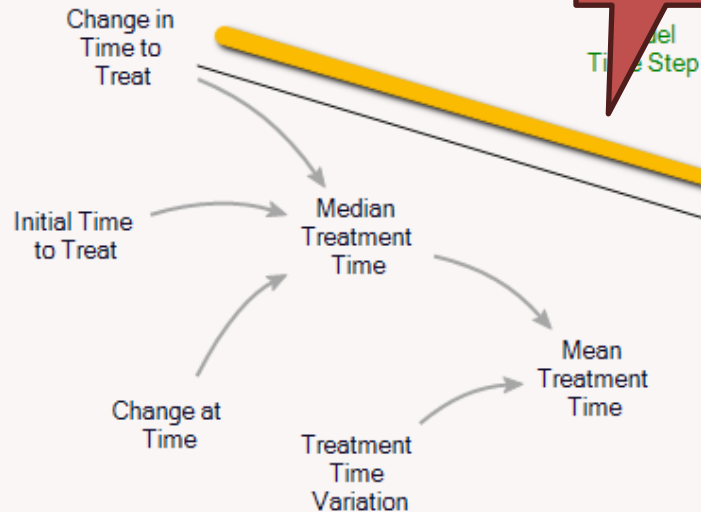
☒ Constant

☐ Enable causes glyph on node

-5

Drag to
connect

Right-click
to edit



| Hospital | Min | Change in Time to Treat: (days) | Max | Input |
|--------------|-----|---------------------------------|------|-------|
| Mass General | -10 | <input type="range"/> | -2.5 | -5 |

Waypoint

- Model so far is saved in PatientSeq 4.vmdl

Notes

- So far, both the continuous and discrete stock-flow representations of the agent patient population have reasonable fidelity
- But ... this breaks down under some conditions, like nonlinearity or strategic behavior among agents



THANKS!