

BNet®.News:
Tips for BNet.EngineKit™

Introduction

In this guide, you'll find a compilation of [BNet.EngineKit](#) tips from BNet.News, the Bayesian Belief Newsletter published by [Charles River Analytics, Inc.](#)

For [BNet.Builder](#)[™] tips, [click here](#).

BNet.EngineKit User Tip

Opening Files

BNet.EngineKit makes it incredibly easy to open Bayesian networks from files. It currently supports XBN, Hugin, and Netica files. Here's how to read a Bayesian network from one of your existing files:

```
BayesianNetwork network = XbnFormat.read(new File("myfile.xbn"));
```

```
BayesianNetwork network = HuginFormat.read(new File("myfile.net"));
```

```
BayesianNetwork network = NeticaFormat.read(new File("myfile.dne"));
```

BNet.EngineKit User Tip

Mutual Information

BNet.EngineKit provides easy access to mutual information values. The `com.cra.bnet.engine.DecisionAnalysisTools` class provides two methods for obtaining mutual information:

- `getMutualInformation(BayesianNetwork network, DiscreteNode queryNode)`
- `getMutualInformation(BayesianNetwork network, DiscreteNode queryNode, double threshold)`

Both methods take a Bayesian network and a query node and return a `java.util.Map` object that maps node names to mutual information values. The second method also takes a threshold value that is used to filter values from the returned Map (only values above the threshold will be included).

The following code sample obtains mutual information values for a query node and prints them out:

```
BayesianNetwork network = ...;  
DiscreteNode node = ...;  
Map values = DecisionAnalysisTools.getMutualInformation(network, node);  
System.out.println("Mutual Information values for " + node.getName());  
System.out.println(values);
```

BNet.EngineKit User Tip

Working with States

The BNet.EngineKit API makes it easy to work with a node's states. The DiscreteNode class provides a `getStates()` method which returns a StateList object. The StateList interface extends the List interface of the Java Collections framework, and makes it easy to add, rename, reorder, and remove states. See the [BNet.EngineKit javadocs](#) for full details.

The next slide shows examples of how to work with states.

BNet.EngineKit User Tip

Working with States - Examples

To add a new state use one of the add methods provided by the List interface. For example, to add a new state named "new state", type :

```
DiscreteNode node = ...;  
StateList states = node.getStates();  
states.add("new state");
```

To rename a state use set method provided by the List interface. For example, to rename the state at index zero to "new state name", type :

```
DiscreteNode node = ...;  
StateList states = node.getStates();  
states.set(0, "new state name");
```

To reorder a state use the reorder method provided by the StateList interface. For example, to reorder the state "true" to index one, type :

```
DiscreteNode node = ...;  
StateList states = node.getStates();  
states.reorder("true", 1);
```

To remove a state use one of the remove methods provided by the List interface. For example, to remove the state "medium", type:

```
DiscreteNode node = ...;  
StateList states = node.getStates();  
states.remove("medium");
```

BNet.EngineKit User Tip

Node Description

You can use the BNet.EngineKit API to set a node's description as a String object. Follow the example shown in this code sample in order to change the description:

```
DiscreteNode node = ...;  
String oldDescription = node.getDescription();  
String newDescription = "This is the new description for the node";  
node.setDescription(newDescription);
```

Note: You can also register your own implementation of the NodeListener interface with a node. Its descriptionChanged method will be called whenever the node's description is changed.

BNet.EngineKit User Tip

Belief Alerts

You can register a BeliefListener with either a DiscreteNode or an entire Bayesian Network to receive events when beliefs change. If you only want to receive belief events that meet specified conditions, you can use a BeliefAlert. For example, to only receive events when the belief for a certain node's "true" state is above 0.9, you can use the ThresholdBeliefCondition class:

```
DiscreteNode node = ...  
BeliefCondition condition = new  
ThresholdBeliefCondition(node, "true",  
ThresholdBeliefCondition.GREATER_THAN, 0.9); BeliefListener listener = ...;  
BeliefAlert alert = new BeliefAlert(condition, listener);
```

When you want to stop receiving these events, call the BeliefAlert object's stop() method. Also note that you can create your own custom implementation of BeliefCondition.

BNet.EngineKit User Tip

CPT Operations

Each node has a Conditional Probability Table (CPT) represented by the Cpt class.

Following are three operations you can perform on a Cpt object:

Normalize - You can use the `normalize()` or `normalizeAll()` methods to make the rows in the CPT add up to one.

Uniform - You can use the `uniform()` or `uniformAll()` methods to set rows in the CPT to a uniform distribution.

Randomize - You can use the `randomizeAll()` method to set all of the CPT entries to random values.

BNet.EngineKit User Tip

Learning

BNet.EngineKit supports learning **Conditional Probability Tables** from data.

- To learn CPTs from fully observed data you can use the **FullyObservableLearningAlgorithm** class.

In fully observed data, you have data for all nodes in the network.

- To learn CPTs from partially observed data you can use the **EmEtaLearningAlgorithm** class.

In partially observed data, you may have missing data or no data for some nodes in the network.

For more information:

Chapter 5 in the [BNet.EngineKit Developers Guide](#) has a section titled "Learning condition probabilities from data" that provides more information on learning CPTs from data.

You can also refer to the [BNet.EngineKit javadocs](#) for full details on using these learning classes.

BNet.EngineKit User Tip

Network Name and Description

You can use the BNet.EngineKit API to set a network's name and description as a String object. Follow the example shown in this code sample in order to change the name and description:

```
BayesianNetwork network = ...;  
String oldName = network.getName();  
String newName = "New network name";  
network.setName(newName);  
String oldDescription = network.getDescription();  
String newDescription = "This is the new description for the network";  
network.setDescription(newDescription);
```

Note: You can also register your own implementation of the PropertyChangeListener interface with a BayesianNetwork object. Its propertyChanged method is called whenever the network's name or description is changed. You can use the public NAME_PROPERTY and DESCRIPTION_PROPERTY fields of the BayesianNetwork class to determine which property has changed.

BNet.EngineKit User Tip

Undo/Redo

The BayesianNetwork class was designed to be used in an interactive editor (like BNet.Builder) and therefore it provides support for undo and redo operations.

The BayesianNetwork class provides an **addUndoableEditListener** method, which you can use to register an **UndoableEditListener** (possibly an UndoManager) that will receive notifications when undoable edits occur.

The **BayesianNetwork** class also provides a number of public inner classes that implement **UndoableEdit**, like **BayesianNetwork.AddNodeEdit**. Instances of these classes are passed to the UndoableEditListener and you can then use these public inner classes to undo and redo various changes to the BayesianNetwork object.

BNet.EngineKit User Tip

Adding a Node

The BayesianNetwork class in BNet.EngineKit provides two methods for adding a node. The first method adds a node with a specified name that has the default states true and false. The following code shows how to add a new node named Alarm with states true and false:

```
BayesianNetwork network = ...;  
DiscreteNode node = network.addNode("Alarm");
```

To add a node with states other than true and false, the BayesianNetwork class provides a second form of the addNode method. The following code shows how to add a node named Temperature with states high, medium, and low:

```
BayesianNetwork network = ...;  
List states = new ArrayList();  
states.add("high");  
states.add("medium");  
states.add("low");  
DiscreteNode node = network.addNode("Temperature", states);
```

You can also use the Arrays utility class that comes with the JDK to simplify this a bit:

```
BayesianNetwork network = ...;  
DiscreteNode node = network.addNode("Temperature", Arrays.asList(new String[] {"high", "medium", "low"}));
```

You can pass any List object into this second form of the addNode method. And remember that you can fully modify the list of states returned by the DiscreteNode.getStates() method.

BNet.EngineKit User Tip

Documentation

BNet.EngineKit includes two useful forms of documentation to help you. You can find them in the docs directory or click on the links below:

[Developer's Guide](#) - task-oriented descriptions of the BNet.EngineKit API.

[Javadocs](#) – a complete API reference detailing all of the classes & methods available in BNet.EngineKit.

BNet.EngineKit User Tip

Saving Files

Opening files in EngineKit is a very common operation -- it's the primary way to use a Bayesian network in your application.

You can also save any BayesianNetwork object to a file using the XbnFormat class:

```
BayesianNetwork network = ...;  
File file = ..;  
XbnFormat.write(network, file);
```

The HuginFormat and NeticaFormat classes also have identical write methods for saving to other formats:

```
DiscreteNode node = ...;  
StateList states = node.getStates();  
states.remove("medium");
```

BNet.EngineKit User Tip

Evidence

The BNet.EngineKit API provides several methods for working with evidence:

- You can set the evidence for a specific state using the `DiscreteNode.setEvidence (String, double)` method (the evidence on all other states will stay the same as before).
- You can set evidence for all states of a node at once using the `DiscreteNode.setEvidence (double[])` method.

Note: In all of these methods, evidence is in the range `[0.0, 1.0]`.

- To obtain the evidence for a specific state, use the `DiscreteNode.getEvidence(String)` method.
- To obtain the evidence for all states of a node, use the `DiscreteNode.getEvidence()` method.
- You can remove the evidence from a node with the `DiscreteNode.removeEvidence()` method and you can remove evidence from all nodes in a network using the `BayesianNetwork.clearEvidence()` method.

BNet.EngineKit User Tip

CPT Indices

The Cpt class provides `get(int, int[])` and `set(int, int[], double)` methods for obtaining and changing individual CPT entries.

The `int` parameter to these methods is the index of the child state (think of it as the column in the CPT view of `BNet.Builder`). The `int[]` parameter specifies one state for each parent (think of it as the row in the CPT view).

The parent states in the `int[]` should be ordered the same as the parents returned by the `Node.getParents()` method.

For nodes without parents, you can use the `Cpt.EMPTY_PARENT_INDICES` field as the `int[]` parameter.

BNet.EngineKit User Tip

Upgrading to the Commercial Version

The Evaluation version of BNet.EngineKit limits networks to five nodes each. You can purchase a license for BNet.EngineKit to remove the five node limit on our Web site at <http://www.cra.com/bnet.enginekit>.

You can obtain your BNet.EngineKit Activation Key in one of two ways:

- Double-click the bnet-engine.jar file (in the lib directory)
- Execute the following on the command line in the lib directory:
`java -jar bnet-engine.jar`

Once you purchase a license, we will send you a license file called BNet.key. You can use it in one of two ways:

- Create a directory named "license" in the root of your project that uses BNet.EngineKit and place the BNet.key license file in that directory
- Embed the BNet.key license file in the root directory inside bnet-engine.jar (using a tool such as WinZip).

BNet.EngineKit User Tip

Belief Update Settings

You can programmatically control when BNet.EngineKit performs belief updates:

1. Use the `BayesianNetwork.getBeliefUpdater()` method to obtain its `BeliefUpdater` object.
2. Then use the `BeliefUpdater.setUpdateOnXxx()` methods to control which events cause EngineKit to update beliefs.

There are also `setUpdateOnEvidence()` and `setUpdateOnAnything()` convenience methods to reduce the number of calls you need to typically make.

BNet.EngineKit User Tip

Handling Events

BNet.EngineKit provides a number of listeners to allow for event handling:

Interface	Interface
NodeListener	- Node name, description, or location changed - Node parents reordered
StateListener	- Node states added, removed, reordered, or names changed
CptListener	- One entry in a node's CPT changed, or the entire CPT changed
EvidenceListener	- Evidence posted or retracted from a node - All evidence retracted from a network
BeliefListener	- Node beliefs changed
PropertyChangeListener	- Node or BayesianNetwork user properties changed

BNet.EngineKit User Tip

Multi-threading with BNet.EngineKit networks

If your software application has multiple threads that may access a BayesianNetwork object (or anything it contains) at the same time, you must use synchronized blocks to handle access of the BayesianNetwork by those threads. If you do not synchronize the access, the BayesianNetwork might provide invalid information or an exception might be thrown, and these errors will be extremely difficult to diagnose.

BNet.EngineKit User Tip

Getting the topology of a belief network

You work with a belief network's topology only when you want to write graph-theoretic algorithms on the belief network. For example, to find all of the nodes in a belief network reachable from a particular node, you would use a standard depth-first search algorithm:

```
BayesianNetwork mobNet = getNetwork("C:\MyBeliefNetworks\MobReaction.xbn");
Node startNode = mobNet.getNode("Instigator");
Graph graph = mobNet.getTopology();
Set visited = new HashSet(Collections.singleton(startNode));
LinkedList stack = new LinkedList(visited);
while (!stack.isEmpty())
{
    Node parent = (Node) stack.removeFirst();
    for (Iterator iterator = graph.outEdgeSet(parent).iterator();
        iterator.hasNext();)
    {
        Edge edge = (Edge) iterator.next();
        Node child = (Node) edge.getOpposite(parent);
        if (!visited.contains(child))
        {
            visited.add(child);
            stack.addFirst(child);
        }
    }
}
```

BNet.EngineKit User Tip

Creating Unique Node Names

The `getUniqueNodeName` method of the `BayesianNetworks` class can be used to create unique node names by adding an integer to a name you provide.

BNet.EngineKit User Tip

Retracting Evidence for all Nodes in a Network

If you need to retract evidence for all nodes in the network, use the `clearEvidence` method of the `BayesianNetwork` class.

BNet.EngineKit User Tip

Getting Beliefs from a Discrete Node

The `getBeliefs` or `getBelief` methods on a discrete node will cause beliefs to be updated to return the most up-to-date beliefs. In contrast, the `getLastBeliefs` and `getLastBelief` methods do not cause beliefs to be updated before returning the beliefs.

BNet.EngineKit User Tip

Handling Errors

You can handle errors generated by BNet.EngineKit classes by registering an `ErrorListener` with the `ErrorNotifier` class.

BNet.News

BNet.News is a free newsletter to keep you up to date on BNet software and applications of belief networks. It is published approximately every other month.

BNet.News brings you:

- Tips and examples you can use in applying belief networks and BNet software
- Interesting applications of belief networks
- Upcoming BNet tutorials
- Product announcements
- Price and Sale announcements
- Beta test opportunities

To subscribe or view past issues of BNet.News, [click here](#) or visit www.cra.com/bnet.news.

More Information

To learn more about our BNet products, visit

www.cra.com/bnet

Or contact bnet@cra.com.