

# Introduction, introspection, illustration

Pierre-Henri WUILLEMIN

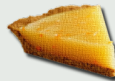
LIP6

`pierre-henri.wuillemin@lip6.fr`

- 1 introduction
  - short history
  - components
  - opensource project
  - next
- 2 introspection : focus on 3 elementary Components
- 3 illustration
  - The model (Liessman Eric Sturlaugson, Montana, 2014)
    - dynamic Bayesian Network
    - Chaîne de Markov à temps continu
    - CTBN
  - Quick implementation of CTBNs using pyAgrum



aGrUM/pyAgrum



# aGrUM/pyAgrum : a (very short) history

(> 10 years) aGrUM's goals (as a tool for laboratory)

# aGrUM/pyAgrum : a (very short) history

(> 10 years) aGrUM's goals (as a tool for laboratory)

- 1 PGM as a library

# aGrUM/pyAgrum : a (very short) history

(> 10 years) aGrUM's goals (as a tool for laboratory)

- 1 PGM as a library (and not as a software  $\Rightarrow$  No IDE).

# aGrUM/pyAgrum : a (very short) history

(> 10 years) aGrUM's goals (as a tool for laboratory)

- 1 PGM as a library (and not as a software  $\Rightarrow$  No IDE).
- 2 C++ !!!

# aGrUM/pyAgrum : a (very short) history

(> 10 years) aGrUM's goals (as a tool for laboratory)

- ➊ PGM as a library (and not as a software  $\Rightarrow$  No IDE).
- ➋ C++ !!!
- ➌ Optimized (as much as possible).



# aGrUM/pyAgrum : a (very short) history

(> 10 years) aGrUM's goals (as a tool for laboratory)

- ➊ PGM as a library (and not as a software  $\Rightarrow$  No IDE).
- ➋ C++ !!!
- ➌ Optimized (as much as possible).
- ➍ Usable and improvable by others than us

# aGrUM/pyAgrum : a (very short) history

(> 10 years) aGrUM's goals (as a tool for laboratory)

- ➊ PGM as a library (and not as a software  $\Rightarrow$  No IDE).
- ➋ C++ !!!
- ➌ Optimized (as much as possible).
- ➍ Usable and improvable by others than us (i.e. students).

# aGrUM/pyAgrum : a (very short) history

(> 10 years) aGrUM's goals (as a tool for laboratory)

- ➊ PGM as a library (and not as a software  $\Rightarrow$  No IDE).
  - ➋ C++ !!!
  - ➌ Optimized (as much as possible).
  - ➍ Usable and improvable by others than us (i.e. students).
- Goals 1 to 3 rather well achieved

# aGrUM/pyAgrum : a (very short) history

(> 10 years) aGrUM's goals (as a tool for laboratory)

- ➊ PGM as a library (and not as a software  $\Rightarrow$  No IDE).
  - ➋ C++ !!!
  - ➌ Optimized (as much as possible).
  - ➍ Usable and improvable by others than us (i.e. students).
- Goals 1 to 3 rather well achieved but 4 was (at least) questionable.

# aGrUM/pyAgrum : a (very short) history

(> 10 years) aGrUM's goals (as a tool for laboratory)

- ① PGM as a library (and not as a software  $\Rightarrow$  No IDE).
  - ② C++ !!!
  - ③ Optimized (as much as possible).
  - ④ Usable and improvable by others than us (i.e. students).
- Goals 1 to 3 rather well achieved but 4 was (at least) questionable.
  - Goals 1 to 3 make aGrUM interesting enough for outside the laboratory.

# aGrUM/pyAgrum : a (very short) history

(> 10 years) aGrUM's goals (as a tool for laboratory)

- ① PGM as a library (and not as a software  $\Rightarrow$  No IDE).
  - ② C++ !!!
  - ③ Optimized (as much as possible).
  - ④ Usable and improvable by others than us (i.e. students).
- Goals 1 to 3 rather well achieved but 4 was (at least) questionable.
  - Goals 1 to 3 make aGrUM interesting enough for outside the laboratory.
  - pyAgrum as a solution.

# aGrUM/pyAgrum : a (very short) history

(> 10 years) aGrUM's goals (as a tool for laboratory)

- ① PGM as a library(and not as a software  $\Rightarrow$  No IDE).
  - ② C++ !!!
  - ③ Optimized (as much as possible).
  - ④ Usable and improvable by others than us (i.e. students).
- Goals 1 to 3 rather well achieved but 4 was (at least) questionable.
  - Goals 1 to 3 make aGrUM interesting enough for outside the laboratory.
  - pyAgrum as a solution.

(< 6 year) pyAgrum's goals :

# aGrUM/pyAgrum : a (very short) history

(> 10 years) aGrUM's goals (as a tool for laboratory)

- ① PGM as a library (and not as a software  $\Rightarrow$  No IDE).
  - ② C++ !!!
  - ③ Optimized (as much as possible).
  - ④ Usable and improvable by others than us (i.e. students).
- Goals 1 to 3 rather well achieved but 4 was (at least) questionable.
  - Goals 1 to 3 make aGrUM interesting enough for outside the laboratory.
  - pyAgrum as a solution.

(< 6 year) pyAgrum's goals :

- ① Wrapper of aGrUM (package, optimized, etc.).



# aGrUM/pyAgrum : a (very short) history

(> 10 years) aGrUM's goals (as a tool for laboratory)

- ① PGM as a library (and not as a software  $\Rightarrow$  No IDE).
  - ② C++ !!!
  - ③ Optimized (as much as possible).
  - ④ Usable and improvable by others than us (i.e. students).
- Goals 1 to 3 rather well achieved but 4 was (at least) questionable.
  - Goals 1 to 3 make aGrUM interesting enough for outside the laboratory.
  - pyAgrum as a solution.

(< 6 year) pyAgrum's goals :

- ① Wrapper of aGrUM (package, optimized, etc.).
- ② Useful, accessible and improvable for as many people as possible.

# aGrUM/pyAgrum : a (very short) history

(> 10 years) aGrUM's goals (as a tool for laboratory)

- ① PGM as a library (and not as a software  $\Rightarrow$  No IDE).
  - ② C++ !!!
  - ③ Optimized (as much as possible).
  - ④ Usable and improvable by others than us (i.e. students).
- Goals 1 to 3 rather well achieved but 4 was (at least) questionable.
  - Goals 1 to 3 make aGrUM interesting enough for outside the laboratory.
  - pyAgrum as a solution.

(< 6 year) pyAgrum's goals :

- ① Wrapper of aGrUM (package, optimized, etc.).
- ② Useful, accessible and improvable for as many people as possible.
- ③ Public, documented and deployed as widely as possible.

# aGrUM/pyAgrum : a (very short) history

(> 10 years) aGrUM's goals (as a tool for laboratory)

- ① PGM as a library(and not as a software  $\Rightarrow$  No IDE).
  - ② C++ !!!
  - ③ Optimized (as much as possible).
  - ④ Usable and improvable by others than us (i.e. students).
- Goals 1 to 3 rather well achieved but 4 was (at least) questionable.
  - Goals 1 to 3 make aGrUM interesting enough for outside the laboratory.
  - pyAgrum as a solution.

(< 6 year) pyAgrum's goals :

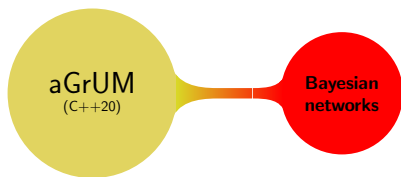
- ① Wrapper of aGrUM (package, optimized, etc.).
  - ② Useful, accessible and improvable for as many people as possible.
  - ③ Public, documented and deployed as widely as possible.
- repository : svn $\rightarrow$ local git $\rightarrow$ local gitlab $\rightarrow$ gitlab
  - Open Source : GPL then LGPL.

# aGrUM/pyAgrum : a framework for modeling, learning and using (probabilistic) graphical models

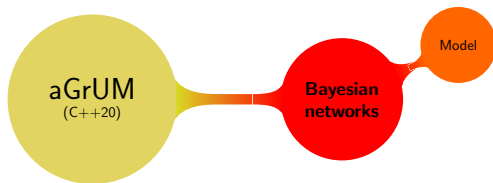


**aGrUM**  
(C++20)

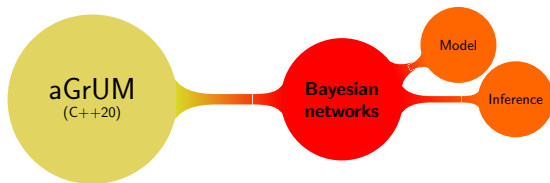
# aGrUM/pyAgrum : a framework for modeling, learning and using (probabilistic) graphical models



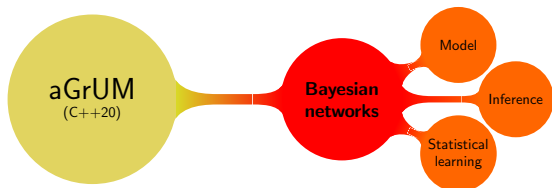
# aGrUM/pyAgrum : a framework for modeling, learning and using (probabilistic) graphical models



# aGrUM/pyAgrum : a framework for modeling, learning and using (probabilistic) graphical models

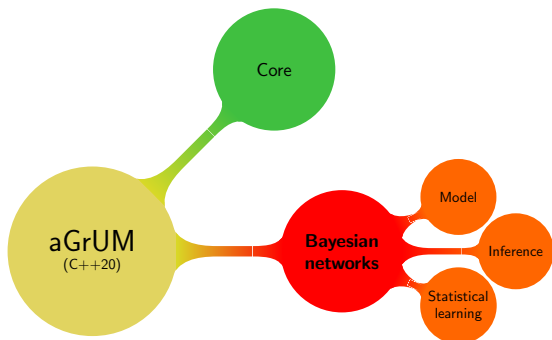


# aGrUM/pyAgrum : a framework for modeling, learning and using (probabilistic) graphical models

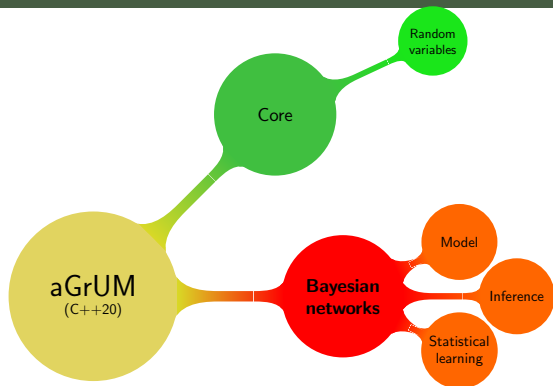




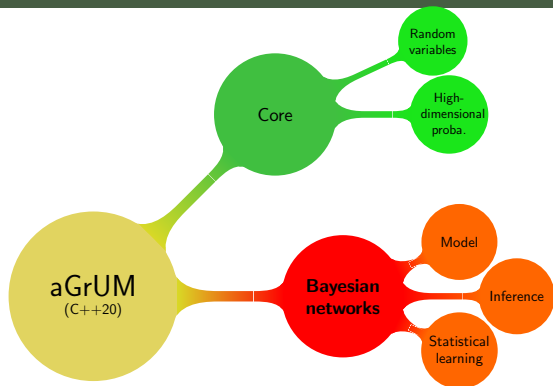
# aGrUM/pyAgrum : a framework for modeling, learning and using (probabilistic) graphical models



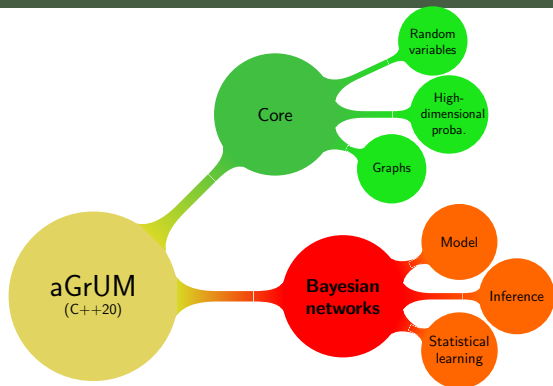
# aGrUM/pyAgrum : a framework for modeling, learning and using (probabilistic) graphical models



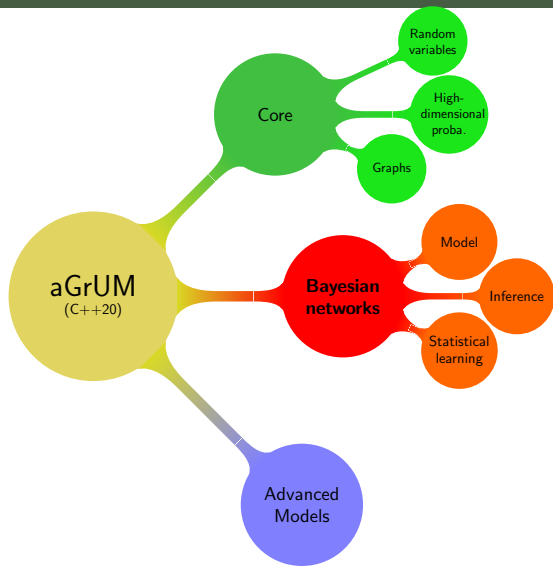
# aGrUM/pyAgrum : a framework for modeling, learning and using (probabilistic) graphical models



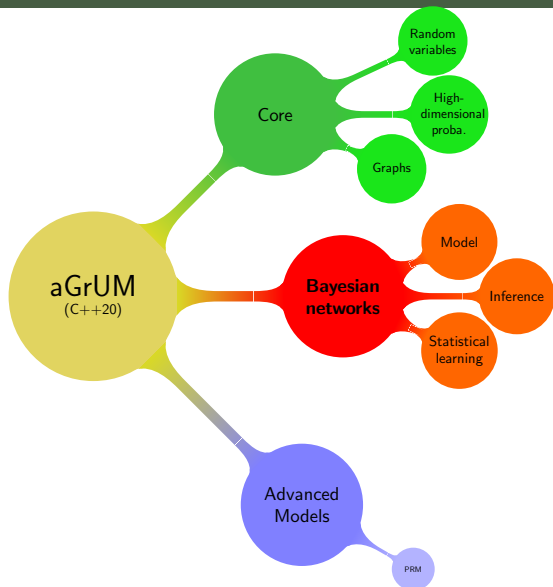
# aGrUM/pyAgrum : a framework for modeling, learning and using (probabilistic) graphical models



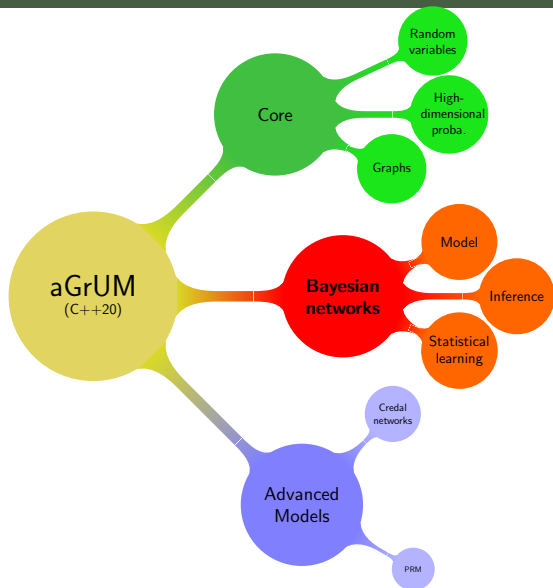
# aGrUM/pyAgrum : a framework for modeling, learning and using (probabilistic) graphical models



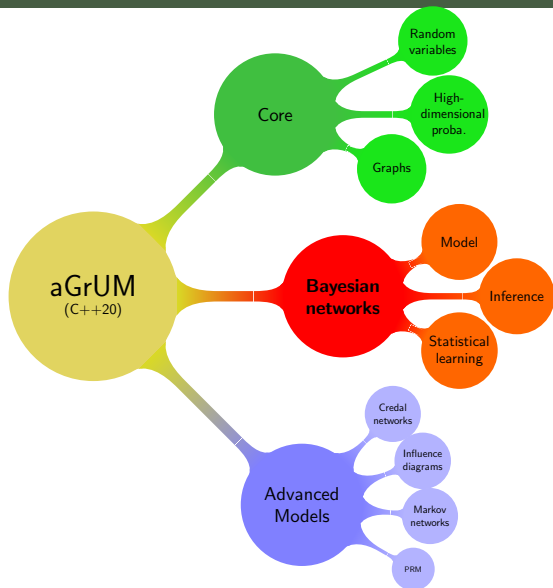
# aGrUM/pyAgrum : a framework for modeling, learning and using (probabilistic) graphical models



# aGrUM/pyAgrum : a framework for modeling, learning and using (probabilistic) graphical models

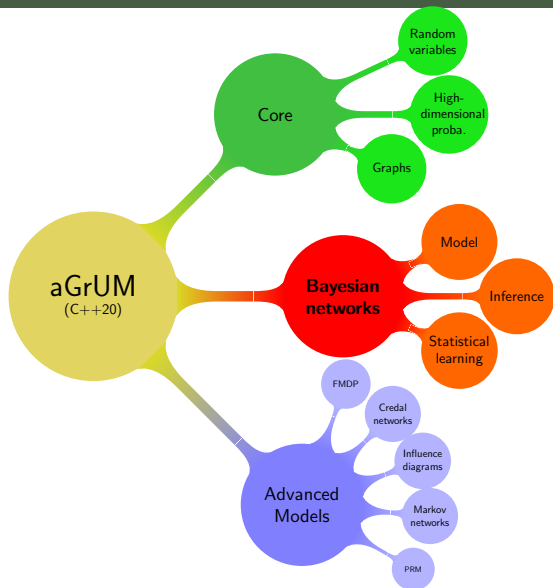


# aGrUM/pyAgrum : a framework for modeling, learning and using (probabilistic) graphical models

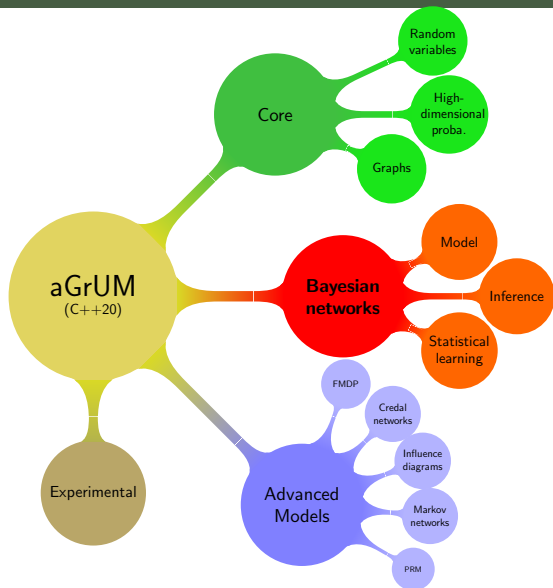




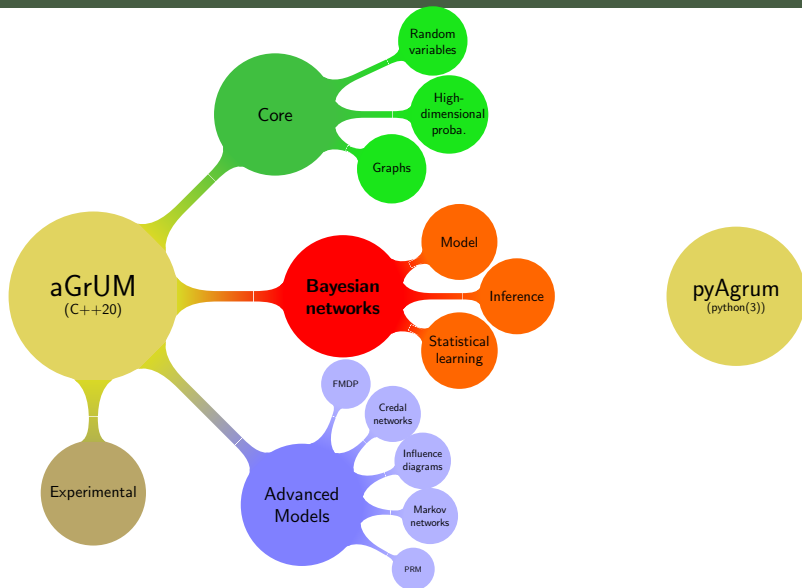
# aGrUM/pyAgrum : a framework for modeling, learning and using (probabilistic) graphical models



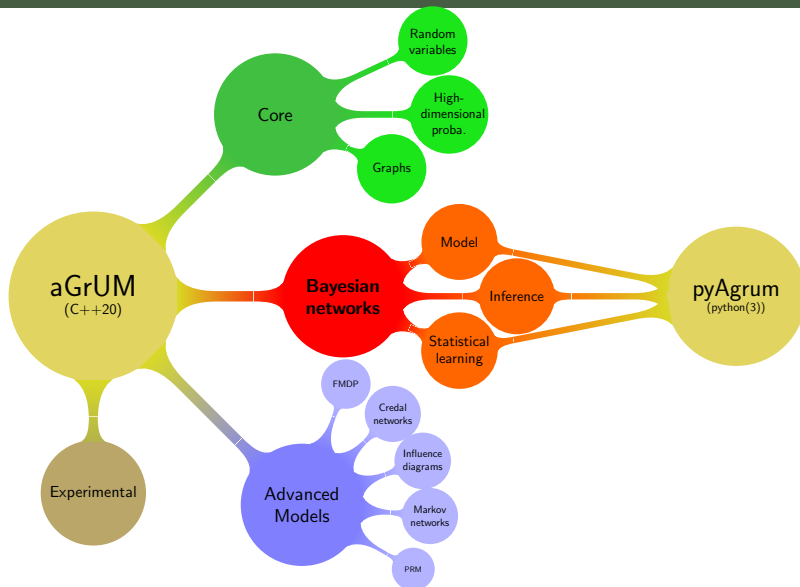
# aGrUM/pyAgrum : a framework for modeling, learning and using (probabilistic) graphical models



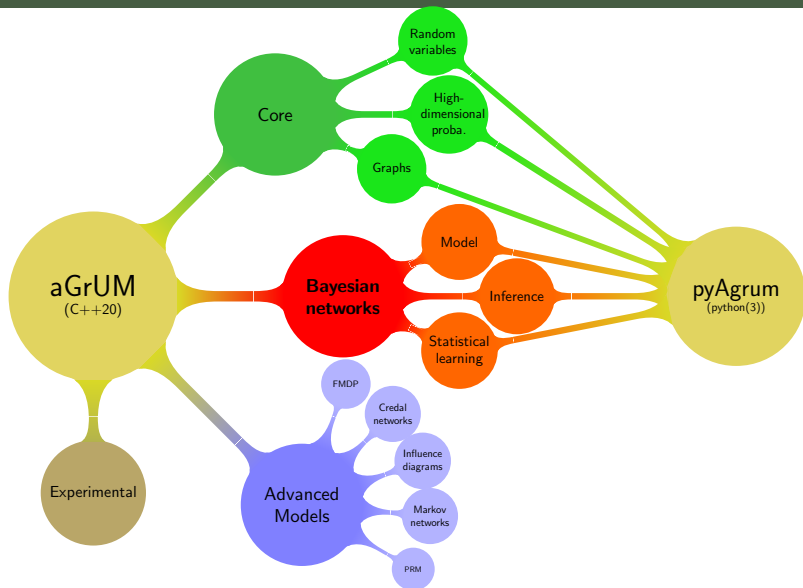
# aGrUM/pyAgrum : a framework for modeling, learning and using (probabilistic) graphical models



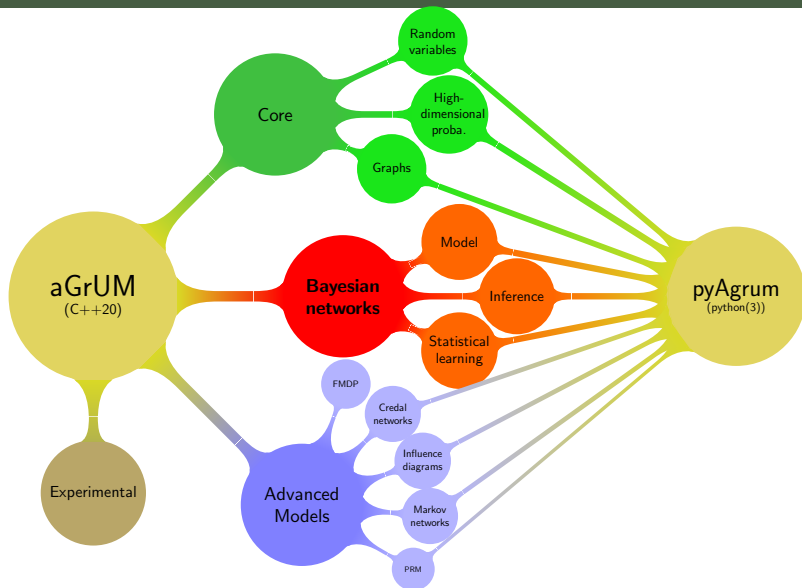
# aGrUM/pyAgrum : a framework for modeling, learning and using (probabilistic) graphical models



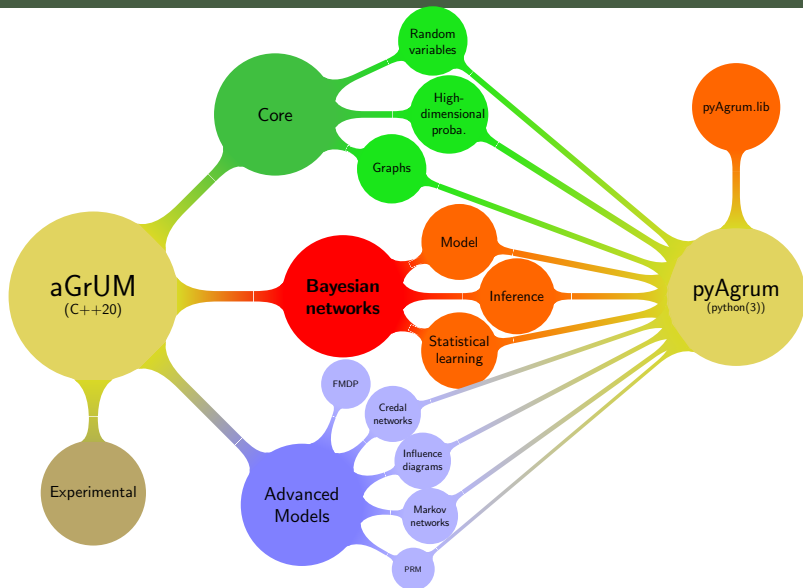
# aGrUM/pyAgrum : a framework for modeling, learning and using (probabilistic) graphical models



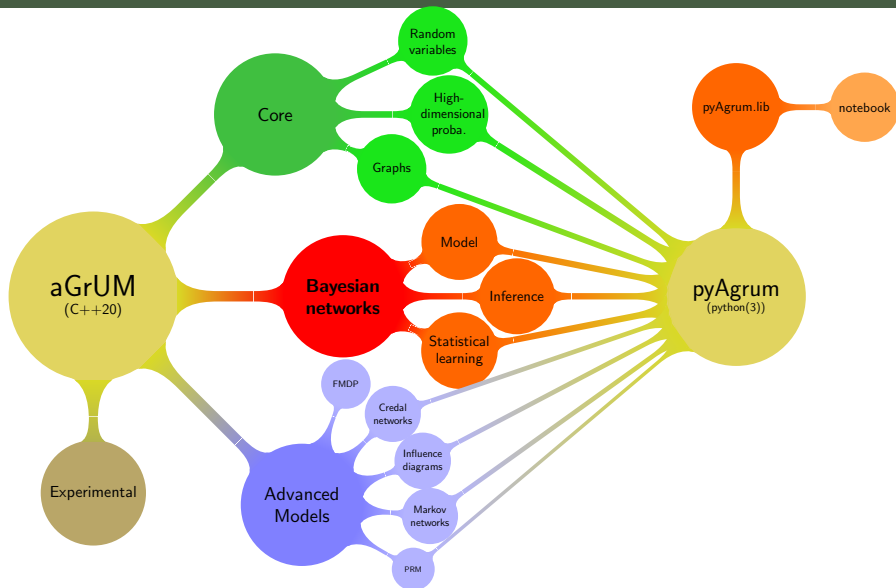
# aGrUM/pyAgrum : a framework for modeling, learning and using (probabilistic) graphical models



# aGrUM/pyAgrum : a framework for modeling, learning and using (probabilistic) graphical models

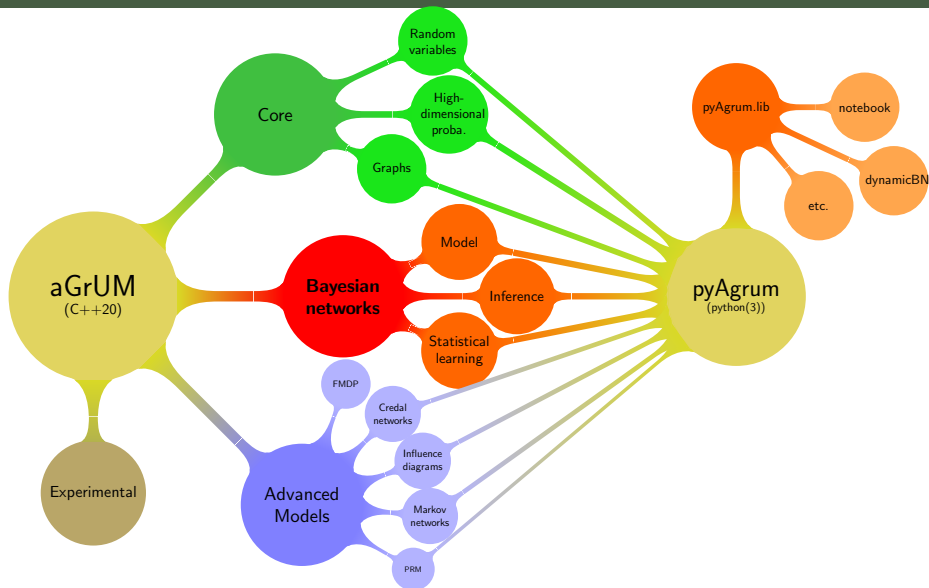


# aGrUM/pyAgrum : a framework for modeling, learning and using (probabilistic) graphical models

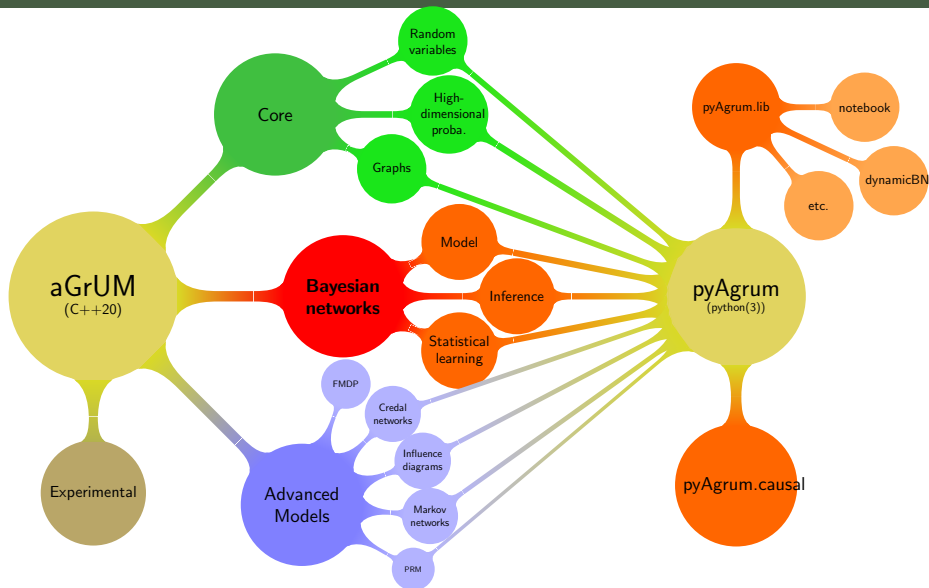




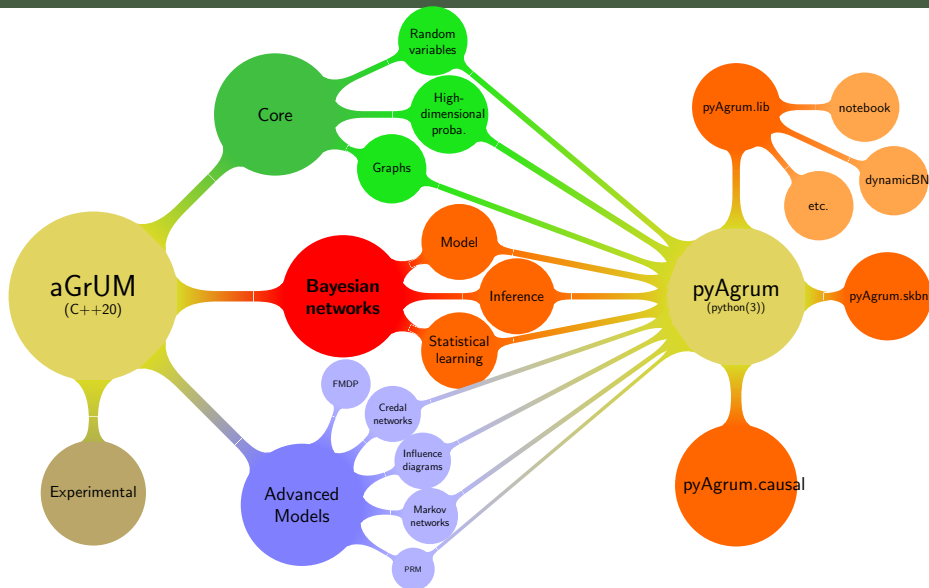
# aGrUM/pyAgrum : a framework for modeling, learning and using (probabilistic) graphical models



# aGrUM/pyAgrum : a framework for modeling, learning and using (probabilistic) graphical models



# aGrUM/pyAgrum : a framework for modeling, learning and using (probabilistic) graphical models





# aGrUM/pyAgrum as OpenSource project



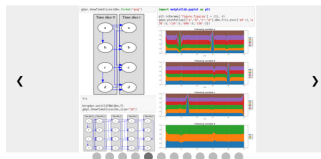
# aGrUM/pyAgrum on the web



The screenshot shows the aGrUM/pyAgrum website with a blue background. At the top is a lemon slice icon. Below it are navigation links for 'Contents', 'aGrUM', 'pyAgrum', 'Documentation', 'aGrUM's doc', 'pyAgrum's ReadTheDocs', 'Notebooks', 'pyAgrum's Tutorials', and 'From The Book Of Why'. A 'RECENT POSTS' section lists updates for aGrUM and pyAgrum versions 0.22.6 and 0.22.1. Below that is a 'Stats on downloads' section with a line graph. At the bottom, there are badges for 'pypi package 0.22.6', 'Anaconda.org 0.22.6', 'dwtids 20.6k/month', and 'dwtids 5.2k/month'. The footer shows logos for aGrUM and pyAgrum.

## aGrUM/pyAgrum

A Graphical Universal Modeler (<https://gitlab.com/agrumer/aGrUM>)



### aGrUM

aGrUM is a C++ library for graphical models. It is designed for easily building applications using graphical models such as Bayesian networks, influence diagrams, decision trees, GAI networks or Markov decision processes.

aGrUM is written to provide the basic building blocks to perform the following tasks :

- ▶ designing graphical models,
- ▶ learning graphical models,
- ▶ elicitation of graphical models,
- ▶ inference within graphical models,
- ▶ planification.

The probabilistic graphical models currently present in the library are the following:

- ▶ Bayesian networks (first and main target),
- ▶ Influence Diagrams,
- ▶ Markov networks,
- ▶ Credal networks,
- ▶ O3PRM (Probabilistic Relational Models).

### pyAgrum



pyAgrum is a Python wrapper for the C++ aGrUM library (using SWIG interface generator). It provides a high-level interface to the part of aGrUM allowing to create, model, learn, use, calculate with and embed Bayesian Networks and other graphical models. Some specific (python and C++) codes are added in order to simplify and extend the aGrUM API.

Several topics have been added to pyAgrum (as pure python modules using pyAgrum) :

- ▶ Scikit-learn-compliant probabilistic classifiers based on Bayesian networks,
- ▶ Probabilistic causality (causal networks, do-calculus),
- ▶ dynamic Bayesian network,
- ▶ tools for explainability in Bayesian networks.

See the [tutorials as jupyter notebooks](#) for more details.

Installation : [here](#)

## Licence




aGrUM/pyAgrum is released under the Gnu Lesser General Public License (LGPL v3.0), which means it can be freely copied and distributed, and costs nothing. Especially, aGrUM can be used and linked into both free software and proprietary software, provided that the code used under the LGPL is re-licensed under the LGPL (the other parts of the software are permitted have other licenses). If you wish to integrate the aGrUM library into your product without being affected by the LGPL, please contact us.

# aGrUM/pyAgrum on gitlab.com

GitLab Next Menu Search GitLab

agrumery > aGrUM




**aGrUM**  Project ID: 2321198 [Leave project](#)

[Bayesian Net...](#) [Graphical Mo...](#) [Probabilist...](#) + 1 more

5,128 Commits 18 Branches 60 Tags 13.8 MB Files 5.9 GB Storage 43 Releases

aGrUM is a C++ library designed for easily building applications using graphical models such as Bayesian networks, influence diagrams, decision trees, GAI networks or Markov decision processes.

master aGrUM / [History](#) [Find file](#) [Web IDE](#) [Clone](#)

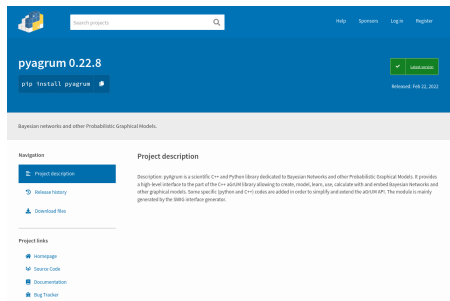
 **[skip-ci] updating 01-Tutorial with fractions**  86547958   
Pierre-Henri Wuillemin authored 2 days ago

[Upload File](#) [README](#) [Other](#) [CHANGELOG](#) [CONTRIBUTING](#) [CI/CD configuration](#) [Add Kubernetes cluster](#)

[Configure Integrations](#)

Name	Last commit	Last update
acttools	[skip-ci] [wheel] update linux_i686 tag	1 month ago
apps	[pyAgrum] pydot instead of pydotplus	3 months ago
binder	[pyAgrum] pydot instead of pydotplus	3 months ago
src	[pyAgrum] empirical gum.DiscretizedVariable used L...	2 weeks ago
wrappers	[skip-ci] updating 01-Tutorial with fractions	2 days ago
.clang-format	[aGrUM] updating guidelines	9 months ago
.clang-tidy	[aGrUM] new heuristics for propagating orientation...	9 months ago
.gitignore	[aGrUM/pyAgrum] corrections for annotations and "...	6 months ago
.gitlab-ci.yml	[ci] remove jobs for win32	1 month ago
mailman	[skip-ci] updating mailman	1 week ago

# pyAgrum on pipy/anaconda/binder/readthedocs



The screenshot shows the PyAgrum project page on the Python Package Index (PyPI). The header is blue with the PyAgrum logo on the left, a search bar, and links for Help, Sponsors, Log in, and Register. The main content area is also blue and features the text "pyagrum 0.22.8" in large white font. Below this, there is a green button that says "pip install pyagrum" and a smaller green button that says "Get it on PyPI". To the right of these buttons, it says "Released: Feb 22, 2022". Below the main content area, there is a light gray section with the text "Bayesian networks and other Probabilistic Graphical Models." followed by a horizontal line. Below the line, there are two columns. The left column is titled "Navigation" and contains three links: "Project description" (highlighted in blue), "Release history", and "Download files". The right column is titled "Project description" and contains a paragraph of text: "Description: pyagrum is a scientific C++ and python library dedicated to Bayesian Networks and other Probabilistic Graphical Models. It provides a high-level interface to the part of the C++ adabo library allowing to create, model, learn, use, calculate with and extend Bayesian Networks and other graphical models. Some specific (python and C++) codes are added in order to simplify and extend the API/USAGE. The module is mainly generated by the SWIG interface generator."

Search projects

Help Sponsors Log in Register

pyagrum 0.22.8

pip install pyagrum

Get it on PyPI

Released: Feb 22, 2022

Bayesian networks and other Probabilistic Graphical Models.

### Navigation

- Project description
- Release history
- Download files

### Project description

Description: pyagrum is a scientific C++ and python library dedicated to Bayesian Networks and other Probabilistic Graphical Models. It provides a high-level interface to the part of the C++ adabo library allowing to create, model, learn, use, calculate with and extend Bayesian Networks and other graphical models. Some specific (python and C++) codes are added in order to simplify and extend the API/USAGE. The module is mainly generated by the SWIG interface generator.

### Project links

- Homepage
- Source Code
- Documentation
- Bug Tracker

# pyAgrum on pipy/anaconda/binder/readthedocs



PyAgrum 0.22.8

pip install pyagrum

Released Feb 22, 2022

Bayesian networks and other Probabilistic Graphical Models.

## Navigation

[Project description](#)[Release history](#)[Download files](#)

## Project links

[Homepage](#)[Source Code](#)[Documentation](#)[Bug Tracker](#)

## Project description

Description: pyagrum is a scientific C++ and python library dedicated to Bayesian Networks and other Probabilistic Graphical Models. It provides a high level interface to the part of the C++ astarlib library allowing to create, model, learn, use, calculate with and extend Bayesian Networks and other graphical models. Some specific (python and C++) codes are added in order to simplify and extend the astarlib API. The module is mainly generated by the SWIG interface generator.

[ANACONDA CLOUD](#)[Gallery](#) [About](#) [Anaconda](#) [Help](#) [Download Anaconda](#) [Sign In](#)

## conda-forge / packages / pyagrum 0.14.2

A wrapper for the Agrum library, to make flexible and scalable probabilistic graphical models.

Conda	Files	Libraries	Binaries
<p>License: GPL3</p> <p>Home: <a href="http://pyagrum.gforge.inria.fr/">http://pyagrum.gforge.inria.fr/</a></p> <p>67226 total downloads</p> <p>Last updated: 16 hours and 38 minutes ago</p>			

## Installers

Info: This package contains files in non-standard format.

## conda install

conda install -c conda-forge pyagrum

To install this package with conda run one of the following

```
conda install -c conda-forge pyagrum
conda install -c conda-forge/label/gcc7 pyagrum
conda install -c conda-forge/label/cf201901 pyagrum
```



# pyAgrum on pipy/anaconda/binder/readthedocs

pyAgrum 0.22.8

pip install pyAgrum

Released: Feb 22, 2022

Bayesian networks and other Probabilistic Graphical Models.

## Navigation

Project description

Release history

Download files

## Project links

Homepage

Source Code

Documentation

Bit Tracker

## Project description

Description: pyAgrum is a scientific C++ and Python library dedicated to Bayesian Networks and other Probabilistic Graphical Models. It provides a high-level interface to the part of the C++ `agrum` library allowing to create, model, learn, use, calculate with and embed Bayesian networks and other graphical models. Some specific (python and C++) codes are added in order to simplify and extend the `agrum` API. The module is mainly generated by the `SWIG` interface generator.

ANACONDA CLOUD

Search Anaconda Cloud

Gallery About Anaconda Help Download Anaconda Sign In

## conda-forge / packages / pyAgrum 0.24.2

A wrapper for the Agrum library, to make flexible and scalable probabilistic graphical models.

Conda	Files	Labels	Badges
<p>License: GPL3</p> <p>Home: <a href="http://agrum.gifalab.io/">http://agrum.gifalab.io/</a></p> <p>67226 total downloads</p> <p>Last upload: 16 hours and 38 minutes ago</p>			

## Installers

Info: This package contains files in non-standard labels.

### conda install

conda install -c conda-forge pyAgrum

To install this package with conda run one of the following:

```
conda install -c conda-forge pyAgrum
conda install -c conda-forge/label/gcc7 pyAgrum
conda install -c conda-forge/label/cf201901 pyAgrum
```

## Description

ANACONDA.ORG

Search Anaconda.org

Gallery About Anaconda Help Download Anaconda Sign In

## conda-forge / packages / pyAgrum 0.22.8

pyAgrum is a Python wrapper for the C++ `agrum` library. It provides a high-level interface to the part of `agrum` allowing to create, model, learn, use, calculate with and embed Bayesian networks and other graphical models. Some specific (Python and C++) codes are added in order to simplify and extend the `agrum` API.

Conda Files Labels Badges

License: LGPL-3.0-only

Home: <http://agrum.gifalab.io/>

498998 total downloads

Last upload: 22 days and 9 hours ago

## Installers

Info: This package contains files in non-standard labels.

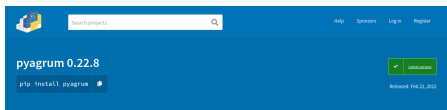
### conda install

conda install -c conda-forge pyAgrum

To install this package with conda run one of the following:

```
conda install -c conda-forge pyAgrum
conda install -c conda-forge/label/gcc7 pyAgrum
conda install -c conda-forge/label/cf201901 pyAgrum
conda install -c conda-forge/label/cf202003 pyAgrum
```

# pyAgrum on pipy/anaconda/binder/readthedocs



pyAgrum 0.22.8

pip install pyAgrum

Released: Feb 22, 2022

Bayesian networks and other Probabilistic Graphical Models.

### Project description

Description: pyAgrum is a scientific C++ and Python library dedicated to Bayesian Networks and other Probabilistic Graphical Models. It provides a high-level interface to the part of the C++ `agrum` library allowing to create, model, learn, use, calculate with and embed Bayesian networks and other graphical models. Some specific (python and C++) codes are added in order to simplify and extend the `agrum` API. The module is mainly generated by the `SWIG` interface generator.

### Navigation

- Project description
- Release history
- Download files

### Project links

- Homepage
- Source Code
- Documentation
- Bit Tracker

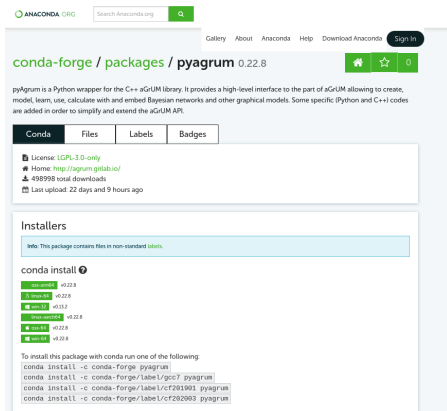
ANACONDA CLOUD Search Anaconda Cloud

Gallery About Anaconda Help Download Anaconda Sign In

## conda-forge / packages / pyAgrum 0.24.2

A wrapper for the Agrum library, to make flexible and scalable probabilistic graphical models.

Conda	Files	Labels	Badges
<p>License: GPL3</p> <p>Home: <a href="http://pyAgrum.github.io/">http://pyAgrum.github.io/</a></p> <p>67226 total downloads</p> <p>Last upload: 16 hours and 38 minutes ago</p>			
<h3>Installers</h3> <p>Info: This package contains files in non-standard labels.</p> <p>conda install</p> <ul style="list-style-type: none"><li>conda-forge/pyAgrum</li><li>conda-forge/pyAgrum</li><li>conda-forge/pyAgrum</li><li>conda-forge/pyAgrum</li><li>conda-forge/pyAgrum</li><li>conda-forge/pyAgrum</li></ul> <p>To install this package with conda run one of the following:</p> <pre>conda install -c conda-forge pyAgrum conda install -c conda-forge/label/gcc7 pyAgrum conda install -c conda-forge/label/cf201901 pyAgrum conda install -c conda-forge/label/cf201901 pyAgrum</pre>			
<h3>Description</h3>			

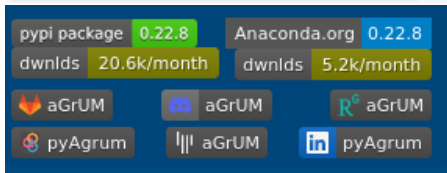


ANACONDA.ORG Search Anaconda.org

conda-forge / packages / pyAgrum 0.22.8

pyAgrum is a Python wrapper for the C++ `agrum` library. It provides a high-level interface to the part of `agrum` allowing to create, model, learn, use, calculate with and embed Bayesian networks and other graphical models. Some specific (Python and C++) codes are added in order to simplify and extend the `agrum` API.

Conda	Files	Labels	Badges
<p>License: LGPL-3.0-only</p> <p>Home: <a href="http://pyAgrum.github.io/">http://pyAgrum.github.io/</a></p> <p>498998 total downloads</p> <p>Last upload: 22 days and 9 hours ago</p>			
<h3>Installers</h3> <p>Info: This package contains files in non-standard labels.</p> <p>conda install</p> <ul style="list-style-type: none"><li>conda-forge/pyAgrum</li><li>conda-forge/pyAgrum</li><li>conda-forge/pyAgrum</li><li>conda-forge/pyAgrum</li><li>conda-forge/pyAgrum</li><li>conda-forge/pyAgrum</li></ul> <p>To install this package with conda run one of the following:</p> <pre>conda install -c conda-forge pyAgrum conda install -c conda-forge/label/gcc7 pyAgrum conda install -c conda-forge/label/cf201901 pyAgrum conda install -c conda-forge/label/cf201901 pyAgrum</pre>			



pyAgrum package 0.22.8

Anaconda.org 0.22.8

dwnlds 20.6k/month

dwnlds 5.2k/month

aGrUM

aGrUM

R<sup>6</sup> aGrUM

pyAgrum

aGrUM

pyAgrum

## Code quality in aGrUM/pyAgrum : documentation

[illegible]

# Code quality in aGrUM/pyAgrum : documentation

**aGrUM 0.22.0**  
a C++ library for probabilistic graphical models

Public Attributes | Public Member Functions | Public Types | Protected Attributes | Protected Member Functions | Static Protected Member Functions | List of all

**gum::learning::BNLearner< GUM\_SCALAR >**  
Class Template Reference

Tools for learning

A pack of learning algorithms that can easily be used. More...

```
#include <BNLearner.h>
```

Inference diagram for gum::learning::BNLearner< GUM\_SCALAR >:

Collaboration diagram for gum::learning::BNLearner< GUM\_SCALAR >:

• We can measure the causal effect of  $W$  on  $X$  using the front-door formula:

```
In [51]: print(" = Front-door doing W on X :", stats.fEffectSize(fModel, frontDoor("w", "x")))
= front-door doing W on X : 1.71
```

• In order to measure the causal effect of  $X$  on  $Y$ , we can use neither the back-door adjustment nor the front-door formula:

```
In [52]: print(" = Backdoor doing X on Y :", stats.fEffectSize(fModel, backDoor("x", "y")))
print(" = Frontdoor doing X on Y :", stats.fEffectSize(fModel, frontDoor("x", "y")))
= backdoor doing X on Y : none
= frontdoor doing X on Y : none
```

• In this case, the only way to measure the causal effect of  $X$  on  $Y$  is to use the do-calculus:

```
In [53]: cslrb.showCausalImpact(fModel, var="y", doing="x")
```



$$P(y|w) = \sum_x P(x|w, 0) \cdot P(y|w, x, 0) = \sum_x P(x|w, 1) \cdot P(y|w, x, 1)$$

	0	1
0	0.6139 0.3860	0.4501 0.5499
1	0.6072 0.3928	0.6119 0.3881

The notebooks proposed here present some of the examples of the **Book of W H Y** from Judea Pearl and Dana Mackenzie. The notebooks are written with the **Jupyter** python interface.

Chapter 1, page 96 - Mini Turing Test

Chapter 1, page 96 - Small Poet

Chapter 2, page 115 - Airport Bag Example

Chapter 4, page 126 - Smoking Example

Chapter 4, page 126 - Back Door Criterion

Chapter 5, page 144 - Smoking

Chapter 6, page 178 - Monty Hall Problem

Chapter 7, page 213 - De Causas

Chapter 7, page 224 - Common Causes For De Trois

Chapter 7, page 226 - Good And Bad Cholesterol

Chapter 8, page 251 - Inductive And Experience

# Code quality in aGrUM/pyAgrum : documentation

**aGrUM v0.22.0**  
a GrUM library for probabilistic graphical models

Public Attributes | Public Member Functions | Public Types | Protected Attributes | Protected Member Functions | Static Protected Member Functions | List of all members

## gum::learning::BNLearner< GUM\_SCALAR > Class Template Reference

Tools for learning

A pack of learning algorithms that can easily be used. More...

```
#include <BNLearner.h>
```

Inference diagram for gum::learning::BNLearner< GUM\_SCALAR >

Collaboration diagram for gum::learning::BNLearner< GUM\_SCALAR >

The notebooks proposed here present some of the examples of the **Book of WHY**, these Jupyter Notebook and these Machine-readable notebooks are access with the **Tableau** python module.

- Chapter 1, page 96 : Mini Turing Test
- Chapter 1, page 96 : Louis XIV
- Chapter 2, page 115 : airport bag example
- Chapter 4, page 120 : smoking example
- Chapter 4, page 126 : back door criterion
- Chapter 5, page 130 : smoking
- Chapter 6, page 178 : mummy Hall Problem
- Chapter 7, page 213 : de Causibus
- Chapter 7, page 224 : common Causes For De Tores
- Chapter 7, page 226 : good And bad cholesterol
- Chapter 8, page 251 : educative And Experience

```
2in 512: print(" > Front-door doing W on X :",stats.fisherz(f@data.de.FrontDoor("w","x")))
> Front-door doing W on X : [1.7]
```

In order to measure the causal effect of X on Y, we can use neither the back-door adjustment nor the front-door formula:

```
2in 512: print(" > Backdoor doing X on Y :",stats.fisherz(f@data.de.BackDoor("x","y")))
print(" > Frontdoor doing X on Y :",stats.fisherz(f@data.de.FrontDoor("x","y")))
> Backdoor doing X on Y : None
> Frontdoor doing X on Y : None
```

In this case, the only way to measure the causal effect of X on Y is to use the do-calculus:

```
2in 512: cxlnb.showCausalImpact(f@data.de,cm="y",doing="x")
```

	g	w	1
0	0.1919	0.3969	
0	0.4501	0.5479	
1	0.0072	0.3329	
1	0.4119	0.5801	

$$P(g|w) = \frac{\sum_x P(x|w,0) \cdot P(w) \cdot P(g|x,w,0)}{\sum_{x,y} P(x|w,1) \cdot P(w) \cdot P(g|x,w,0)}$$

### Lazy Propagation

Shutter Sherry Inference  
Variable Elimination

#### Approximated Inference

Learning

Influence Diagram  
Credal Network  
Markov Network  
Probabilistic Relational Models

#### CAUSALITY

pyAgrum.causal documentation

#### SCRIPT LEARN LIKE BN CLASSIFIERS

pyAgrum.kknn documentation

#### PROGRAM LIB MODULES

pyAgrum.lib.notebook  
pyAgrum.lib.image  
pyAgrum.lib.explain  
pyAgrum.lib.dynamicBN  
other pyAgrum.lib modules

#### MISCELLANEOUS

Functions from pyAgrum  
Other functions from aGrUM  
Exceptions from aGrUM

#### CUSTOMIZING PYAGRUM

Configuration for pyAgrum

## Lazy Propagation

Lazy Propagation is the main exact inference for classical Bayesian networks in aGrUM/pyAgrum.

`class pyAgrum.LazyPropagation(target)`

Class used for Lazy Propagation

`LazyPropagation(bn) -> LazyPropagation`

Parameters:

`bn (pyAgrum.BayesNet)` – a Bayesian network

`BN()`

Returns:

A constant reference over the `IBayesNet` referenced by this class.

Return type:

`pyAgrum.IBayesNet`

Raises:

`pyAgrum.UndefinedElement` – If no Bayes net has been assigned to the inference.

`H('arg')`

Parameters:

`X (int)` – a node id

`nodeName (str)` – a node name

Returns:

the computed Shannon's entropy of a node given the observation

Return type:

`float`



## Code quality in aGrUM/pyAgrum : tests

```

PMPHypeTestsSuite.h ..... [1 ms] color bar switch colors from (white/70(blue) to (gray/20(green))
PartialOrderedTriangulationTestSuite.h ..... [2 ms]
PotentialTestsSuite.h ..... [22 ms] color bar switch colors from (white/70(blue) to (gray/20(green))
PriorityQueueTestsSuite.h ..... [1 ms] connection bar switch colors from (white/70(blue) to (gray/20(green))
ProjectorMultiDimFunctionGraphTestSuite.h ..... [0 ms] color bar switch colors from (white/70(blue) to (gray/20(green))
RangeVariableTestSuite.h ..... [0 ms]
RangeDatabaseTableTestSuite.h ..... [19 ms] color bar switch colors from (white/70(blue) to (range)) (slow)
RecordCounterTestSuite.h ..... [1010 ms] connection bar switch colors from (white/70(blue) to (gray/20(green))
RefFitTestSuite.h ..... [0 ms]
RegressTestSuite.h ..... [1 ms] color bar switch colors from (white/70(blue) to (range)) (slow)
SlyTestSuite.h ..... [176 ms]
ScalarAttributeTestSuite.h ..... [3 ms]
ScheduleCliqueStoreMultiDimTestSuite.h ..... [0 ms]
ScheduleCombinationBasicTestSuite.h ..... [1 ms]
ScheduleCombineTestSuite.h ..... [0 ms]
ScheduleLabelerMultiDimTestSuite.h ..... [0 ms]
ScheduleMultiDimTestSuite.h ..... [0 ms] # vs OpenSource project!
ScheduleOperationTestSuite.h ..... [0 ms]
ScheduleProjectionBasicTestSuite.h ..... [0 ms]
ScheduleProjectionBasicTestSuite.h ..... [257 ms]
ScheduleSeparatorStoreMultiDimTestSuite.h ..... [0 ms]
ScheduleTestSuite.h ..... [0 ms] # vs [img]/img/3.png
SchedulerBasicTestSuite.h ..... [1 ms]
ScoreDictTestSuite.h ..... [11 ms]
ScoreDictTestSuite.h ..... [0 ms]
ScoreDictTestSuite.h ..... [0 ms] num avg [img]
ScoreDictTestSuite.h ..... [0 ms] [img]/img/2.png
ScoreDictTestSuite.h ..... [0 ms]
ScoreDictTestSuite.h ..... [0 ms]
ScoreDictTestSuite.h ..... [25 ms] play/anaconda/binder/readthedocs/
ScoringCacheTestSuite.h ..... [0 ms]
SequenceTestSuite.h ..... [0 ms]
SeqTestSuite.h ..... [0 ms] # vs [img]
SetTestSuite.h ..... [0 ms]
ShaferShenoyIncrementalTestSuite.h ..... [17 ms] # vs [img]
ShaferShenoyInferenceTestSuite.h ..... [1391 ms]
SlotChartTestSuite.h ..... [117 ms]
LocalObjectAllocatorTestSuite.h ..... [0 ms]
SpanningForestPrinterTestSuite.h ..... [0 ms] # vs [img]/img/8.png
StructuralComparatorTestSuite.h ..... [1 ms]
StructureDBayesBallTestSuite.h ..... [169 ms] # vs [img]/img/7.png
StructureInferenceTestSuite.h ..... [850 ms]
StructurePrinterTestSuite.h ..... [1515 ms]
ThreeOffTwoTestSuite.h ..... [539 ms]
TimerTestSuite.h ..... [3000 ms]
TorcsTestSuite.h ..... [7 ms]
UnlabeledTestSuite.h ..... [2 ms]
UnlabeledTestSuite.h ..... [0 ms] # vs agrum/agrum - continuous integration!
UndiGraphTestSuite.h ..... [1 ms]
VariableEliminationTestSuite.h ..... [2500 ms]
VariableNodeMapTestSuite.h ..... [1 ms]
WeightedSamplingTestSuite.h ..... [261 ms]
DocumentationTestSuite.h ..... [16 ms]
# Profiling : 72321 ms #
# Failed 0 of 1699 tests
Agnum Test Unit Moduler: 100% OK !
Time spent in cmake : 2.448s , make : 596.965s and post : 72.491s

```

# Code quality in aGrUM/pyAgrum : continuous integration

## CI on different platforms

The screenshot shows the GitLab CI/CD interface for the aGrUM project. The left sidebar contains navigation links: Project, Repository, Issues (2), Merge Requests (1), CI/CD, Pipelines, Jobs, Schedules, Charts, Operations, Registry, Wiki, Snippets, and Settings. The main content area displays the 'Pipelines' view for the 'aGrUM' project. It shows a 'passed' status for Pipeline #49619229, triggered 6 days ago. Below this, it indicates '8 jobs from master in 58 minutes and 36 seconds'. A 'Build' section lists six jobs: linux\_agrum, linux\_pyagrum, macos\_agrum, macos\_pyagrum, and windows\_agrum, all with 'passed' status and refresh icons.

## Deployment (to be continued)

The screenshot shows the GitLab CI/CD interface for the aGrUM project, specifically the 'Deploy' pipeline. The left sidebar is the same as the previous screenshot. The main content area displays the 'Deploy' pipeline, which is 'passed'. It shows '3 jobs from 0.14.2 in 18 minutes and 38 seconds (queued for 1 second)'. A 'Build' section lists three jobs: linux\_agrum, linux\_pyagrum, and linux\_build, all with 'passed' status and refresh icons. A 'Deploy' section shows a single job, linux\_build, also with 'passed' status and a refresh icon.

## Nightly build (and tests)

The screenshot shows the GitLab CI/CD interface for the aGrUM project, displaying a list of nightly builds. The left sidebar is the same as the previous screenshots. The main content area shows a list of builds, each with a 'passed' status, a 'latest' tag, and a refresh icon. The builds are triggered by 'Ymaster' and include updates to 'VERSION.txt a...'. The builds are dated 3 hours ago, 1 day ago, and 00:34:15.



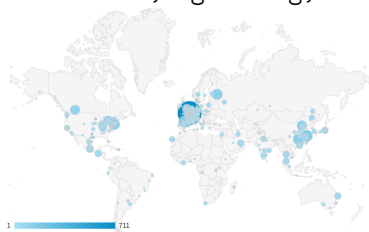
# Some stats

# Some stats

- Visits (`readthedocs`, `agrum.org`, `notebooks`)

# Some stats

## ● Visits (readthedocs, agrum.org, notebooks)



### Vues uniques

16 févr. 2022 - 17 mars 2022

**5 835**

% du total : 100,00 % (5 835)



16 févr. 2021 - 17 mars 2021

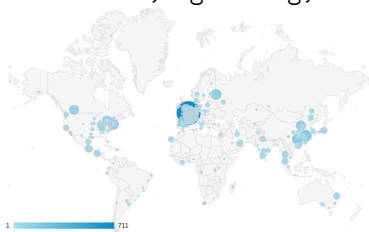
**3 079**

% du total : 100,00 % (3 079)



# Some stats

## ● Visits (readthedocs, agrum.org, notebooks)



### Vues uniques

16 févr. 2022 - 17 mars 2022

**5 835**

% du total : 100,00 % (5 835)



16 févr. 2021 - 17 mars 2021

**3 079**

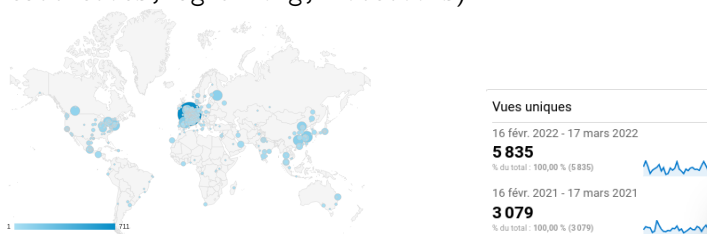
% du total : 100,00 % (3 079)



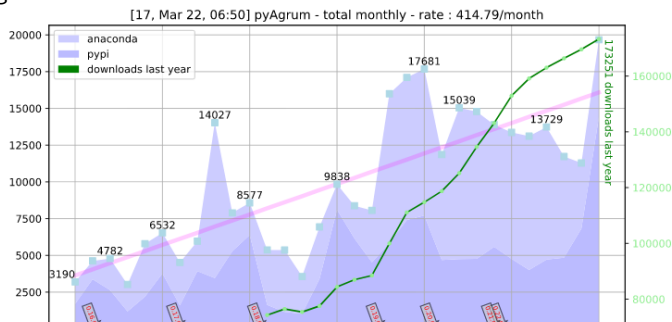
## ● Téléchargements

# Some stats

## ● Visits (readthedocs, agrum.org, notebooks)



## ● Téléchargements



# And now ?

- aGrUM/pyAgrum still a lab/academic tool. We will not stop maintaining & developing !

# And now ?

- aGrUM/pyAgrum still a lab/academic tool. We will not stop maintaining & developing !
- Many users imply many responsibilities

# And now ?

- aGrUM/pyAgrum still a lab/academic tool. We will not stop maintaining & developing !
- Many users imply many responsibilities
  - Interaction



# And now ?

- aGrUM/pyAgrum still a lab/academic tool. We will not stop maintaining & developing !
- Many users imply many responsibilities
  - Interaction  
gitlab issues, discord, gitter, linkedin, researchGate, what else ?
  - Structuration

# And now ?

- aGrUM/pyAgrum still a lab/academic tool. We will not stop maintaining & developing !
- Many users imply many responsibilities
  - Interaction  
gitlab issues, discord, gitter, linkedin, researchGate, what else ?
  - Structuration  
communaute ( ? ), consortium ( ? )
  - Scientific orientation ?
    - models
    - algorithms
    - scientific committee
    - ?

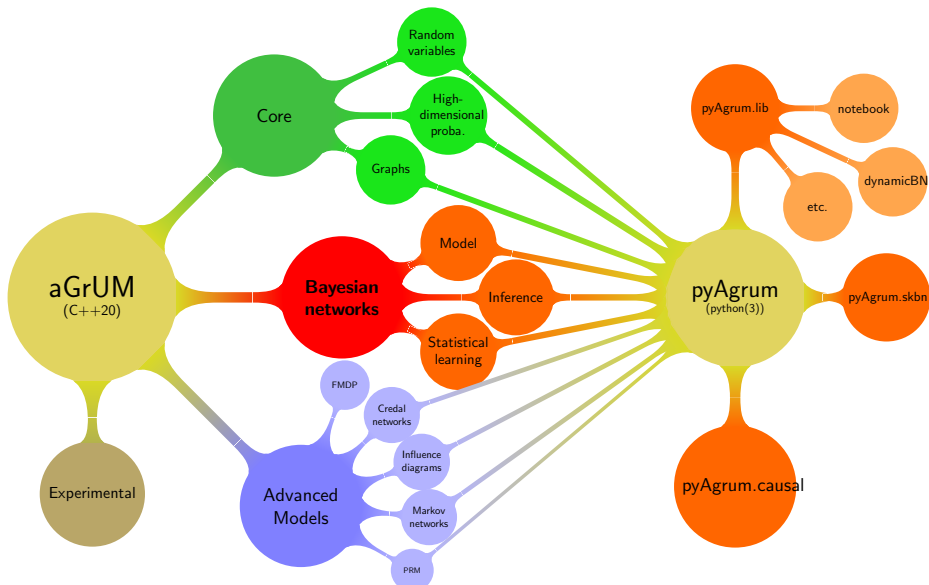
# And now ?

- aGrUM/pyAgrum still a lab/academic tool. We will not stop maintaining & developing !
- Many users imply many responsibilities
  - Interaction  
gitlab issues, discord, gitter, linkedin, researchGate, what else ?
  - Structuration  
communaute ( ? ), consortium ( ? )
  - Scientific orientation ?
    - models
    - algorithms
    - scientific committee
    - ?
  - Development orientation ?

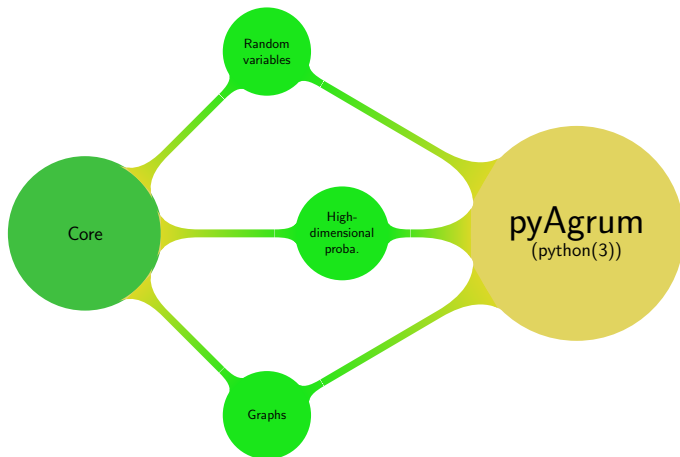
# And now ?

- aGrUM/pyAgrum still a lab/academic tool. We will not stop maintaining & developing !
- Many users imply many responsibilities
  - Interaction  
gitlab issues, discord, gitter, linkedin, researchGate, what else ?
  - Structuration  
communaute ( ? ), consortium ( ? )
  - Scientific orientation ?
    - models
    - algorithms
    - scientific committee
    - ?
  - Development orientation ?
    - weaknesses, strengths
    - missing features
    - Ragrum, JSagrum
    - Steering committee
    - ?

# Introspection : focus on 3 elementary components



# Introspection : focus on 3 elementary components



# Representation of Discrete Variable

# Representation of Discrete Variable

Data structure : `DiscreteVariable`

**goal** : map a finite domain  $[0, \dots, \text{domainSize}]$  on a list of labels.



# Representation of Discrete Variable

## Data structure : DiscreteVariable

**goal** : map a finite domain  $[0, \dots, \text{domainSize}]$  on a list of labels.

For a DiscreteVariable  $X$  that can take the values  $a, e, i, o, u, y$ ,  $X$  is represented by an array :

index	0	1	2	3	4	5
label	a	e	i	o	u	y

# Representation of Discrete Variable

## Data structure : DiscreteVariable

**goal** : map a finite domain  $[0, \dots, domainSize]$  on a list of labels.

For a DiscreteVariable  $X$  that can take the values  $a, e, i, o, u, y$ ,  $X$  is represented by an array :

index	0	1	2	3	4	5
label	a	e	i	o	u	y

The kind of labels defines 4 different types of discrete variables :

# Representation of Discrete Variable

## Data structure : DiscreteVariable

**goal** : map a finite domain  $[0, \dots, \text{domainSize}]$  on a list of labels.

For a DiscreteVariable  $X$  that can take the values  $a, e, i, o, u, y$ ,  $X$  is represented by an array :

index	0	1	2	3	4	5
label	a	e	i	o	u	y

The kind of labels defines 4 different types of discrete variables :

- LabeledVariable : list of generic labels (as string),

# Representation of Discrete Variable

## Data structure : DiscreteVariable

**goal** : map a finite domain  $[0, \dots, \text{domainSize}]$  on a list of labels.

For a DiscreteVariable  $X$  that can take the values  $a, e, i, o, u, y$ ,  $X$  is represented by an array :

index	0	1	2	3	4	5
label	a	e	i	o	u	y

The kind of labels defines 4 different types of discrete variables :

- LabeledVariable : list of generic labels (as string),
- RangeVariable : list of contiguous integer labels,

# Representation of Discrete Variable

## Data structure : DiscreteVariable

**goal** : map a finite domain  $[0, \dots, \text{domainSize}]$  on a list of labels.

For a DiscreteVariable  $X$  that can take the values  $a, e, i, o, u, y$ ,  $X$  is represented by an array :

index	0	1	2	3	4	5
label	a	e	i	o	u	y

The kind of labels defines 4 different types of discrete variables :

- LabeledVariable : list of generic labels (as string),
- RangeVariable : list of contiguous integer labels,
- DiscretizedVariable : list of labels defined by a list of float ticks (see below),

# Representation of Discrete Variable

## Data structure : DiscreteVariable

**goal** : map a finite domain  $[0, \dots, \text{domainSize}]$  on a list of labels.

For a DiscreteVariable  $X$  that can take the values  $a, e, i, o, u, y$ ,  $X$  is represented by an array :

index	0	1	2	3	4	5
label	a	e	i	o	u	y

The kind of labels defines 4 different types of discrete variables :

- LabeledVariable : list of generic labels (as string),
- RangeVariable : list of contiguous integer labels,
- DiscretizedVariable : list of labels defined by a list of float ticks (see below),
- IntegerVariable : list of non-contiguous integer labels.

# Representation of Discrete Variable

## Data structure : DiscreteVariable

**goal** : map a finite domain  $[0, \dots, \text{domainSize}]$  on a list of labels.

For a DiscreteVariable  $X$  that can take the values  $a, e, i, o, u, y$ ,  $X$  is represented by an array :

index	0	1	2	3	4	5
label	a	e	i	o	u	y

The kind of labels defines 4 different types of discrete variables :

- LabeledVariable : list of generic labels (as string),
- RangeVariable : list of contiguous integer labels,
- DiscretizedVariable : list of labels defined by a list of float ticks (see below),
- IntegerVariable : list of non-contiguous integer labels.

# Discrete variables as list of labels

The kind of labels defines 4 different types of discrete variables :

- `LabelizedVariable` : list of generic labels (as string),
- `RangeVariable` : list of contiguous integer labels,
- `DiscretizedVariable` : list of labels defined by a list of float ticks (see below),
- `IntegerVariable` : list of non-contiguous integer labels.



# Discrete variables as list of labels

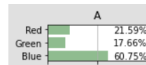
The kind of labels defines 4 different types of discrete variables :

- `LabelizedVariable` : list of generic labels (as string),
- `RangeVariable` : list of contiguous integer labels,
- `DiscretizedVariable` : list of labels defined by a list of float ticks (see below),
- `IntegerVariable` : list of non-contiguous integer labels.

`gum.LabelizedVariable`

`A{Red|Green|Blue}`

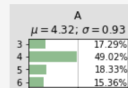
A		
Red	Green	Blue
0.2159	0.1766	0.6075



`gum.RangeVariable`

`A[3,6]`

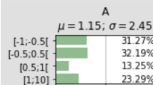
A			
3	4	5	6
0.1729	0.4902	0.1833	0.1536



`gum.DiscretizedVariable`

`A[-1, -0.5, 0.5, 1, 10]`

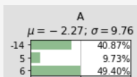
A			
[-1;-0.5[	[-0.5;0.5[	[0.5;1[	[1;10]
0.3127	0.3219	0.1325	0.2329



`gum.IntegerVariable`

`A{-14|5|6}`

A		
-14	5	6
0.4087	0.0973	0.4940



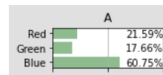
```
def aff(name, fastStx):
    bn=gum.fastBN(fastStx)
    return (f"<h2>{name}</h2>",
            f"<tt>{fastStx}</tt>",
            gnb.getPotential(bn.cpt(0)),
            gnb.getPosterior(bn,target=0,evs={}))

gnb.sideBySide(*aff("gum.LabelizedVariable", "A{Red|Green|Blue}"),
               *aff("gum.RangeVariable", "A[3,6]"),
               *aff("gum.DiscretizedVariable", "A[-1,-0.5,0.5,1,10]"),
               *aff("gum.IntegerVariable", "A{-14|5|6}"),
               ncols=4)
```

gum.LabelizedVariable

A{Red|Green|Blue}

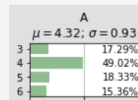
A		
Red	Green	Blue
0.2159	0.1766	0.6075



gum.RangeVariable

A[3,6]

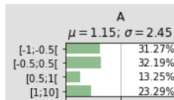
A			
3	4	5	6
0.1729	0.4902	0.1833	0.1536



gum.DiscretizedVariable

A[-1,-0.5,0.5,1,10]

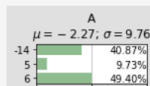
A			
[-1;-0.5[	[-0.5;0.5[	[0.5;1[	[1;10]
0.3127	0.3219	0.1325	0.2329



gum.IntegerVariable

A{-14|5|6}

A		
-14	5	6
0.4087	0.0973	0.4940



# A shared and simple API for all Discrete Variables

# A shared and simple API for all Discrete Variables

```
def apiDiscreteVar(variable,value,position):  
    print(f"{variable} : {value=}, {position=}")  
    print(f" + {variable.domainSize()=}")  
    print(f" + {variable[value]=}")  
    print(f" + {variable.index(value)=}")  
    print(f" + {variable.label(position)=}")
```

```
apiDiscreteVar(bn.variable("A"), "Green", 0)  
apiDiscreteVar(bn.variable("B"), "5", 0)  
apiDiscreteVar(bn.variable("C"), "0.75", 0)  
apiDiscreteVar(bn.variable("D"), "5", 0)
```

```
A: Labeled(<Red,Green,Blue>) : value='Green', position=0
```

```
 + variable.domainSize()=3  
 + variable[value]=1  
 + variable.index(value)=1  
 + variable.label(position)='Red'
```

```
B: Range([3,6]) : value='5', position=0
```

```
 + variable.domainSize()=4  
 + variable[value]=2  
 + variable.index(value)=2  
 + variable.label(position)='3'
```

```
C: Discretized(<[-1;-0.5],[ -0.5;0.5],[0.5;1],[1;10]>) : value='0.75', position=0
```

```
 + variable.domainSize()=4  
 + variable[value]=2  
 + variable.index(value)=2  
 + variable.label(position)='[-1;-0.5['
```

```
D: Integer(<-14,5,6>) : value='5', position=0
```

```
 + variable.domainSize()=3  
 + variable[value]=1  
 + variable.index(value)=1  
 + variable.label(position)='-14'
```

# Representation of graphs

Data structure : Directed | Mixed | Unoriented Graph

**goal** : represent a list of Arc  $(a, b)$  and Edge  $\{b, a\}$  between positive integers.

# Representation of graphs

Data structure : `Directed` | `Mixed` | `UnorientedGraph`

**goal** : represent a list of Arc  $(a, b)$  and Edge  $\{b, a\}$  between positive integers.

- Very compact definition of a graph : not even explicit set of nodes

# Representation of graphs

Data structure : `Directed` | `Mixed` | `UnorientedGraph`

**goal** : represent a list of Arc  $(a, b)$  and Edge  $\{b, a\}$  between positive integers.

- Very compact definition of a graph : not even explicit set of nodes
- Nodes, edges and arcs will be annotated for more complex structures based on graphs.

# Representation of graphs

Data structure : `Directed` | `Mixed` | `UnorientedGraph`

**goal** : represent a list of `Arc (a, b)` and `Edge {b, a}` between positive integers.

- Very compact definition of a graph : not even explicit set of nodes
- Nodes, edges and arcs will be annotated for more complex structures based on graphs.
- the nodes (`unsigned long`) are called `NodeId`



# Representation of graphs

Data structure : `Directed` | `Mixed` | `UnorientedGraph`

**goal** : represent a list of Arc  $(a, b)$  and Edge  $\{b, a\}$  between positive integers.

- Very compact definition of a graph : not even explicit set of nodes
- Nodes, edges and arcs will be annotated for more complex structures based on graphs.
- the nodes (unsigned long) are called `NodeId`

Several type of graphs :

- `DiGraph`
  - DAG (Directed Acyclic Graph)
- `UndiGraph` (and `CliqueGraph`)
- `MixedGraph`

# API for graphs

- node : `addNode`, `addNodeWithId(a)`, `addNodes(nbr)` , `eraseNode(a)`

# API for graphs

- node : `addNode`, `addNodeWithId(a)`, `addNodes(nbr)` , `eraseNode(a)`
- arcs, edges : `addEdge`, `eraseEdge`, etc.

# API for graphs

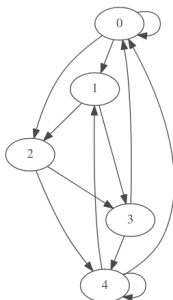
- node : `addNode`, `addNodeWithId(a)`, `addNodes(nbr)` , `eraseNode(a)`
- arcs, edges : `addEdge`, `eraseEdge`, etc.
- accessors : `parents`, `children`, `neighbour`, etc.
- algorithms : `topologicalOrder`, `moralGraph`, `connectedComponents`, etc.
- visualisation : `toDot()` (used by `pyAgrum.lib.notebook` for instance)

## Digraph

```
g=gum.DiGraph()  
g.addNodes(5) # returns the generated nodeId
```

```
{0, 1, 2, 3, 4}
```

```
for i in range(5):  
    g.addArc(i,(i+1)%5)  
    g.addArc(i,(i+2)%5)  
g.addArc(0,0)  
g.addArc(4,4)  
g
```



# API for graphs

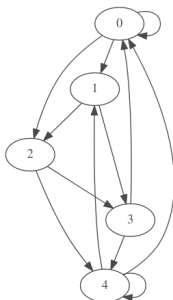
- node : `addNode`, `addNodeWithId(a)`, `addNodes(nbr)` , `eraseNode(a)`
- arcs, edges : `addEdge`, `eraseEdge`, etc.
- accessors : `parents`, `children`, `neighbour`, etc.
- algorithms : `topologicalOrder`, `moralGraph`, `connectedComponents`, etc.
- visualisation : `toDot()` (used by `pyAgrum.lib.notebook` for instance)

## Digraph

```
g=gum.DiGraph()  
g.addNodes(5) # returns the generated nodeId
```

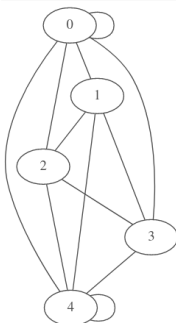
```
{0, 1, 2, 3, 4}
```

```
for i in range(5):  
    g.addArc(i,(i+1)%5)  
    g.addArc(i,(i+2)%5)  
g.addArc(0,0)  
g.addArc(4,4)  
g
```



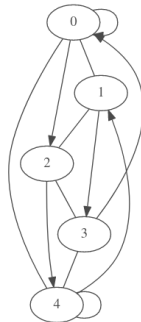
## UndiGraph

```
g=gum.UndiGraph()  
g.addNodes(5)  
for i in range(5):  
    g.addEdge(i,(i+1)%5)  
    g.addEdge(i,(i+2)%5)  
g.addEdge(0,0)  
g.addEdge(4,4)  
g
```



## Mixed Graph

```
g=gum.MixedGraph()  
g.addNodes(5)  
for i in range(5):  
    g.addEdge(i,(i+1)%5)  
    g.addArc(i,(i+2)%5)  
g.addArc(0,0)  
g.addEdge(4,4)  
g
```



# Representation of multi-dimensionnal arrays

# Representation of multi-dimensionnal arrays

Data structure : Potential

**goal** : representation of multi-dimensional arrays (of float) without ambiguity on dimensions.

# Representation of multi-dimensionnal arrays

## Data structure : Potential

**goal** : representation of multi-dimensional arrays (of float) without ambiguity on dimensions.

## Implementing tensor algebra

$$f = g + h$$



# Representation of multi-dimensionnal arrays

## Data structure : Potential

**goal** : representation of multi-dimensional arrays (of float) without ambiguity on dimensions.

## Implementing tensor algebra

$$f(,,) = g(,,) + h(,,)$$

# Representation of multi-dimensionnal arrays

## Data structure : Potential

**goal** : representation of multi-dimensional arrays (of float) without ambiguity on dimensions.

## Implementing tensor algebra

$$f(,,) = g(,,) + h(,,)$$

[[[1. 2.] [3. 4.]] [5. 6.] [7. 8.]]  
 $g(,,)$

[[1. 3.] [5. 7.]]  
 $h(,,)$

?  
 $f(,,) = g(,,) + h(,,)$

# Representation of multi-dimensionnal arrays

## Data structure : Potential

**goal** : representation of multi-dimensional arrays (of float) without ambiguity on dimensions.

## Implementing tensor algebra

$$f(a, b, c) = g(b, a, c) + h(b, c)$$

[[[1. 2.] [3. 4.]] [[5. 6.] [7. 8.]]]  
g(, ,)

[[1. 3.] [5. 7.]]  
h(, )

?  
 $f(, ,) = g(, ,) + h(, ,)$

# Representation of multi-dimensionnal arrays

## Data structure : Potential

**goal** : representation of multi-dimensional arrays (of float) without ambiguity on dimensions.

## Implementing tensor algebra

$$f(a, b, c) = g(b, a, c) + h(b, c)$$

		A	
C	B	0	1
0	0	1.0000	2.0000
	1	3.0000	4.0000
1	0	5.0000	6.0000
	1	7.0000	8.0000

$g(a, b, c)$

B		
C	0	1
0	1.0000	3.0000
1	5.0000	7.0000

$h(b, c)$

		B	
A	C	0	1
0	0	2.0000	6.0000
	1	10.0000	14.0000
1	0	3.0000	7.0000
	1	11.0000	15.0000

$f(a, b, c) = g(a, b, c) + h(b, c)$

# API for Potential



Quite complex API.

# API for Potential



Quite complex API.

## Creation and operations on Potential

# API for Potential



Quite complex API.

## Creation and operations on Potential

```
a=gum.LabelizedVariable("A","descr of A",2)
b=gum.LabelizedVariable("B","descr of B",2)
c=gum.LabelizedVariable("C","descr of C",2)

g=gum.Potential().add(b).add(a).add(c).fillWith([1,2,3,4,5,6,7,8])
h=gum.Potential().add(b).add(c).fillWith([1,3,5,7])

f=g+h
```

# API for Potential



Quite complex API.

## Creation and operations on Potential

```
a=gum.LabelizedVariable("A","descr of A",2)
b=gum.LabelizedVariable("B","descr of B",2)
c=gum.LabelizedVariable("C","descr of C",2)

g=gum.Potential().add(b).add(a).add(c).fillWith([1,2,3,4,5,6,7,8])
h=gum.Potential().add(b).add(c).fillWith([1,3,5,7])

f=g+h
```

## Methods on Potential



# API for Potential



Quite complex API.

## Creation and operations on Potential

```
a=gum.LabelizedVariable("A","descr of A",2)
b=gum.LabelizedVariable("B","descr of B",2)
c=gum.LabelizedVariable("C","descr of C",2)

g=gum.Potential().add(b).add(a).add(c).fillWith([1,2,3,4,5,6,7,8])
h=gum.Potential().add(b).add(c).fillWith([1,3,5,7])

f=g+h
```

## Methods on Potential

```
gnb.sideBySide(f,f.sum(),f.margSumOut("B"),
  captions=['$$f(a,b,c)$$','$$\sum_{a,b,c} f(a,b,c)$$','$$\sum_{[b]} f(a,b,c)$$'])
```

		B	
A	C	0	1
0	0	2.0000	5.0000
	1	10.0000	13.0000
1	0	4.0000	7.0000
	1	12.0000	15.0000

$f(a, b, c)$

$$\sum_{a,b,c} 68.0$$

		C	
A		0	1
0		7.0000	23.0000
1		11.0000	27.0000

$$\sum_b f(a, b, c)$$

# API for Potential



Quite complex API.

## Creation and operations on Potential

```
a=gum.LabelizedVariable("A","descr of A",2)
b=gum.LabelizedVariable("B","descr of B",2)
c=gum.LabelizedVariable("C","descr of C",2)

g=gum.Potential().add(b).add(a).add(c).fillWith([1,2,3,4,5,6,7,8])
h=gum.Potential().add(b).add(c).fillWith([1,3,5,7])

f=g+h
```

## Methods on Potential

```
gnb.sideBySide(f,f.sum(),f.margSumOut("B"),
  captions=['$$f(a,b,c)$$','$$\sum_{a,b,c} f(a,b,c)$$','$$\sum_{[b]} f(a,b,c)$$'])
```

		B	
A	C	0	1
0	0	2.0000	5.0000
	1	10.0000	13.0000
1	0	4.0000	7.0000
	1	12.0000	15.0000

$f(a, b, c)$

$$\sum_{a,b,c} 68.0$$

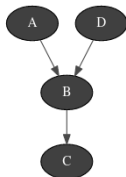
		C	
A		0	1
0		7.0000	23.0000
1		11.0000	27.0000

$$\sum_b f(a, b, c)$$

# Potential for probabilities

# Potential for probabilities

```
bn=gum.fastBN("A->B->C;D->B")  
gnb.sideBySide(bn,bn.cpt("A"),bn.cpt("B"),bn.cpt("C"),bn.cpt("D"))
```



A	
0	1
0.1495	0.8505

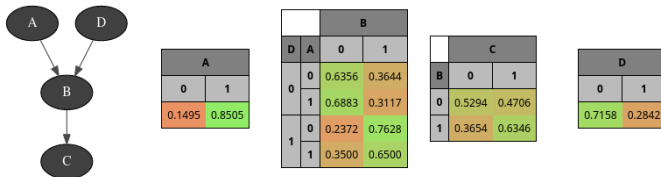
D	A	B	
		0	1
0	0	0.6356	0.3644
	1	0.6883	0.3117
1	0	0.2372	0.7628
	1	0.3500	0.6500

B	C	
	0	1
0	0.5294	0.4706
1	0.3654	0.6346

D	
0	1
0.7158	0.2842

# Potential for probabilities

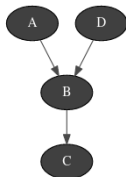
```
bn=gum.fastBN("A->B->C;D->B")
gnb.sideBySide(bn,bn.cpt("A"),bn.cpt("B"),bn.cpt("C"),bn.cpt("D"))
```



$$P(A, B, C, D) = P(A) * P(D) * P(B|A, D) * P(C|B)$$

# Potential for probabilities

```
bn=gum.fastBN("A->B->C;D->B")
gnb.sideBySide(bn,bn.cpt("A"),bn.cpt("B"),bn.cpt("C"),bn.cpt("D"))
```



A	
0	1
0.1495	0.8505

		B	
D	A	0	1
	0	0.6356	0.3644
1	0	0.6883	0.3117
	1	0.2372	0.7628
	1	0.3500	0.6500

	C	
B	0	1
0	0.5294	0.4706
1	0.3654	0.6346

D	
0	1
0.7158	0.2842

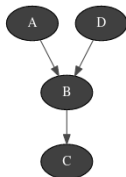
$$P(A, B, C, D) = P(A) * P(D) * P(B|A, D) * P(C|B)$$

```
pABCD=bn.cpt("A")*bn.cpt("B")*bn.cpt("C")*bn.cpt("D")
pABCD
```

			D	
A	C	B	0	1
0	0	0	0.0360	0.0053
		1	0.0143	0.0118
	1	0	0.0320	0.0047
		1	0.0247	0.0206
1	0	0	0.2218	0.0448
		1	0.0693	0.0574
	1	0	0.1972	0.0398
		1	0.1204	0.0997

# Potential for probabilities

```
bn=gum.fastBN("A->B->C;D->B")
gnb.sideBySide(bn,bn.cpt("A"),bn.cpt("B"),bn.cpt("C"),bn.cpt("D"))
```



A	
0	1
0.1495	0.8505

		B	
D	A	0	1
	0	0.6356	0.3644
1	0	0.6883	0.3117
	1	0.2372	0.7628
	1	0.3500	0.6500

	C	
B	0	1
0	0.5294	0.4706
1	0.3654	0.6346

D	
0	1
0.7158	0.2842

$$P(A, B, C, D) = P(A) * P(D) * P(B|A, D) * P(C|B)$$

```
pABCD=bn.cpt("A")*bn.cpt("B")*bn.cpt("C")*bn.cpt("D")
pABCD
```

			D	
A	C	B	0	1
0	0	0	0.0360	0.0053
		1	0.0143	0.0118
	1	0	0.0320	0.0047
		1	0.0247	0.0206
1	0	0	0.2218	0.0448
		1	0.0693	0.0574
	1	0	0.1972	0.0398
		1	0.1204	0.0997

## Potential for probabilities (2)



## Potential for probabilities (2)

$$P(D|C)$$

## Potential for probabilities (2)

$$P(D|C) = \frac{P(D, C)}{P(C)}$$

## Potential for probabilities (2)

$$P(D|C) = \frac{P(D, C)}{P(C)} = \frac{\sum_{A, B} P(A, B, C, D)}{\sum_{A, B, D} P(A, B, C, D)}$$

## Potential for probabilities (2)

$$P(D|C) = \frac{P(D, C)}{P(C)} = \frac{\sum_{A,B} P(A, B, C, D)}{\sum_{A,B,D} P(A, B, C, D)}$$

```
pABCD.margSumOut([ "A", "B" ])/pABCD.margSumIn("C")
```

	C	
D	0	1
0	0.7409	0.6943
1	0.2591	0.3057

## Potential for probabilities (2)

$$P(D|C) = \frac{P(D, C)}{P(C)} = \frac{\sum_{A,B} P(A, B, C, D)}{\sum_{A,B,D} P(A, B, C, D)}$$

```
pABCD.margSumOut([ "A", "B" ])/pABCD.margSumIn("C")
```

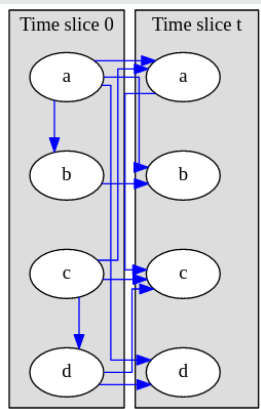
	C	
D	0	1
0	0.7409	0.6943
1	0.2591	0.3057

# Using these components to build new model in pyAgrum

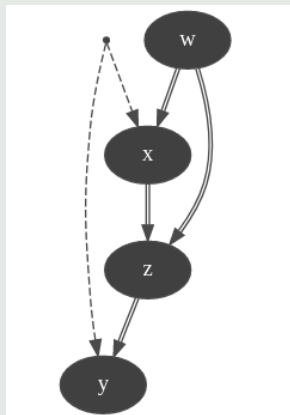
# Using these components to build new model in pyAgrum

Two models exist only in pyAgrum and has been developed mainly from those components :

## dynamic Bayesian Network



## Causal model



## dBN (dynamic BN)



## dBN (dynamic BN)

a dynamic Bayesian network is a Bayesian network with variables indexed by the time  $t$  and by  $i$  :  $\mathbf{X}^{(t)} = X_1^{(t)}, \dots, X_N^{(t)}$

## dBN (dynamic BN)

a dynamic Bayesian network is a Bayesian network with variables indexed by the time  $t$  and by  $i$  :  $\mathbf{X}^{(t)} = X_1^{(t)}, \dots, X_N^{(t)}$  for which those properties hold :

## dBN (dynamic BN)

a dynamic Bayesian network is a Bayesian network with variables indexed by the time  $t$  and by  $i$  :  $\mathbf{X}^{(t)} = X_1^{(t)}, \dots, X_N^{(t)}$  for which those properties hold :

- Markov property :

## dBN (dynamic BN)

a dynamic Bayesian network is a Bayesian network with variables indexed by the time  $t$  and by  $i$  :  $\mathbf{X}^{(t)} = X_1^{(t)}, \dots, X_N^{(t)}$  for which those properties hold :

- Markov property :

$$P(\mathbf{X}^{(t)} \mid \mathbf{X}^{(0)}, \dots, \mathbf{X}^{(t-1)}) = P(\mathbf{X}^{(t)} \mid \mathbf{X}^{(t-1)}),$$

- Homogeneity :

## dBN (dynamic BN)

a dynamic Bayesian network is a Bayesian network with variables indexed by the time  $t$  and by  $i$  :  $\mathbf{X}^{(t)} = X_1^{(t)}, \dots, X_N^{(t)}$  for which those properties hold :

- Markov property :

$$P(\mathbf{X}^{(t)} \mid \mathbf{X}^{(0)}, \dots, \mathbf{X}^{(t-1)}) = P(\mathbf{X}^{(t)} \mid \mathbf{X}^{(t-1)}),$$

- Homogeneity :

$$P(\mathbf{X}^{(t)} \mid \mathbf{X}^{(t-1)}) = \dots = P(\mathbf{X}^{(1)} \mid \mathbf{X}^{(0)}).$$

# dynamic Bayesian Networks

## dBN (dynamic BN)

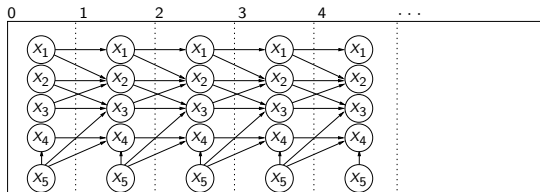
a dynamic Bayesian network is a Bayesian network with variables indexed by the time  $t$  and by  $i$  :  $\mathbf{X}^{(t)} = X_1^{(t)}, \dots, X_N^{(t)}$  for which those properties hold :

- Markov property :

$$P(\mathbf{X}^{(t)} \mid \mathbf{X}^{(0)}, \dots, \mathbf{X}^{(t-1)}) = P(\mathbf{X}^{(t)} \mid \mathbf{X}^{(t-1)}),$$

- Homogeneity :

$$P(\mathbf{X}^{(t)} \mid \mathbf{X}^{(t-1)}) = \dots = P(\mathbf{X}^{(1)} \mid \mathbf{X}^{(0)}).$$



## 2-TBN

A dynamic Bayesian network is defined by

## 2-TBN

A dynamic Bayesian network is defined by

- initial distributions ( $P(X^{(0)})$ ),



## 2-TBN

A dynamic Bayesian network is defined by

- initial distributions ( $P(X^{(0)})$ ),
- the transition between the variables at time  $t - 1$  and the same variables at time  $t$  (*timeslices*).

## 2-TBN

A dynamic Bayesian network is defined by

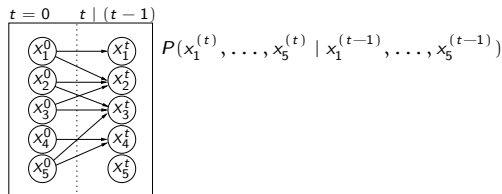
- initial distributions ( $P(X^{(0)})$ ),
- the transition between the variables at time  $t - 1$  and the same variables at time  $t$  (*timeslices*).

# dynamic Bayesian networks : 2-TBN

## 2-TBN

A dynamic Bayesian network is defined by

- initial distributions ( $P(X^{(0)})$ ),
- the transition between the variables at time  $t - 1$  and the same variables at time  $t$  (*timeslices*).

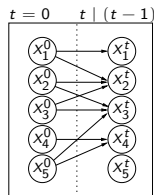


# dynamic Bayesian networks : 2-TBN

## 2-TBN

A dynamic Bayesian network is defined by

- initial distributions ( $P(X^{(0)})$ ),
- the transition between the variables at time  $t - 1$  and the same variables at time  $t$  (*timeslices*).



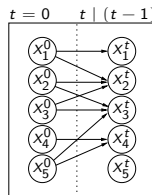
$$P(x_1^{(t)}, \dots, x_5^{(t)} \mid x_1^{(t-1)}, \dots, x_5^{(t-1)}) = P(x_1^{(t)} \mid x^{(t-1)})$$

# dynamic Bayesian networks : 2-TBN

## 2-TBN

A dynamic Bayesian network is defined by

- initial distributions ( $P(X^{(0)})$ ),
- the transition between the variables at time  $t - 1$  and the same variables at time  $t$  (*timeslices*).



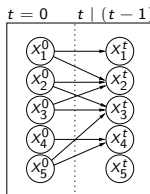
$$P(x_1^{(t)}, \dots, x_5^{(t)} \mid x_1^{(t-1)}, \dots, x_5^{(t-1)}) = P(x_1^{(t)} \mid x^{(t-1)}) \\ P(x_2^{(t)} \mid x_1^{(t-1)}, x_2^{(t-1)}, x_3^{(t-1)})$$

# dynamic Bayesian networks : 2-TBN

## 2-TBN

A dynamic Bayesian network is defined by

- initial distributions ( $P(X^{(0)})$ ),
- the transition between the variables at time  $t - 1$  and the same variables at time  $t$  (*timeslices*).



$$P(x_1^{(t)}, \dots, x_5^{(t)} \mid x_1^{(t-1)}, \dots, x_5^{(t-1)})$$

1024 versus  $4+16+16+8+2=46$ !!

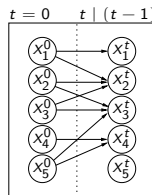
$$\begin{aligned} &= P(x_1^{(t)} \mid x^{(t-1)}) \\ &P(x_2^{(t)} \mid x_1^{(t-1)}, x_2^{(t-1)}, x_3^{(t-1)}) \\ &P(x_3^{(t)} \mid x_2^{(t-1)}, x_3^{(t-1)}, x_4^{(t-1)}) \\ &P(x_4^{(t)} \mid x_4^{(t-1)}, x_5^{(t-1)}) \end{aligned}$$

# dynamic Bayesian networks : 2-TBN

## 2-TBN

A dynamic Bayesian network is defined by

- initial distributions ( $P(X^{(0)})$ ),
- the transition between the variables at time  $t - 1$  and the same variables at time  $t$  (*timeslices*).



$$P(x_1^{(t)}, \dots, x_5^{(t)} \mid x_1^{(t-1)}, \dots, x_5^{(t-1)})$$

1024 versus  $4+16+16+8+2=46$ !!

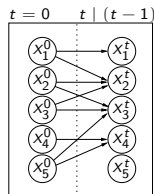
$$\begin{aligned} &= P(x_1^{(t)} \mid x^{(t-1)}) \\ &P(x_2^{(t)} \mid x_1^{(t-1)}, x_2^{(t-1)}, x_3^{(t-1)}) \\ &P(x_3^{(t)} \mid x_2^{(t-1)}, x_3^{(t-1)}, x_4^{(t-1)}) \\ &P(x_4^{(t)} \mid x_4^{(t-1)}, x_5^{(t-1)}) \\ &P(x_5^{(t)}) \end{aligned}$$

# dynamic Bayesian networks : 2-TBN

## 2-TBN

A dynamic Bayesian network is defined by

- initial distributions ( $P(X^{(0)})$ ),
- the transition between the variables at time  $t - 1$  and the same variables at time  $t$  (*timeslices*).



$$P(x_1^{(t)}, \dots, x_5^{(t)} \mid x_1^{(t-1)}, \dots, x_5^{(t-1)})$$

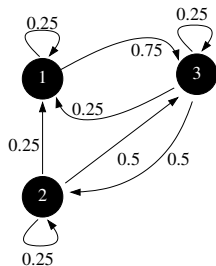
1024 versus  $4+16+16+8+2=46$ !!

$$\begin{aligned} &= P(x_1^{(t)} \mid x^{(t-1)}) \\ &P(x_2^{(t)} \mid x_1^{(t-1)}, x_2^{(t-1)}, x_3^{(t-1)}) \\ &P(x_3^{(t)} \mid x_2^{(t-1)}, x_3^{(t-1)}, x_4^{(t-1)}) \\ &P(x_4^{(t)} \mid x_4^{(t-1)}, x_5^{(t-1)}) \\ &P(x_5^{(t)}) \end{aligned}$$

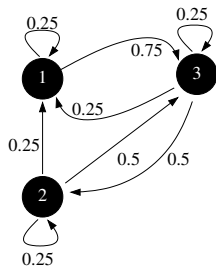
The representation a.k.a **2TBN** (2 timeslices BN) allow the modelisation of a virtually infinite Bayesian network which is the unrolled model from time 0.



# Markov Chain and dynamic Bayesian network



# Markov Chain and dynamic Bayesian network



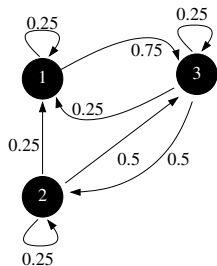
$$P(X^n | X^{n-1}) = \begin{pmatrix} 0.25 & 0 & 0.75 \\ 0.25 & 0.25 & 0.5 \\ 0.25 & 0.5 & 0.25 \end{pmatrix}$$

## Markov chain

- a discrete variable ( $X^n$ ) (at time  $n$ ).
- Parameters for this model :
  - Initial condition :  $P(X^0)$
  - transition probabilities :  $P(X^n | X^{n-1})$

Equivalent dynamic Bayesian network :

# Markov Chain and dynamic Bayesian network

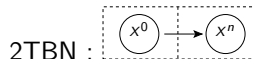
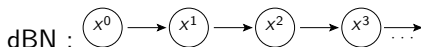


$$P(X^n | X^{n-1}) = \begin{pmatrix} 0.25 & 0 & 0.75 \\ 0.25 & 0.25 & 0.5 \\ 0.25 & 0.5 & 0.25 \end{pmatrix}$$

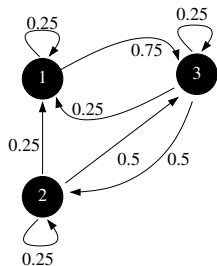
## Markov chain

- a discrete variable ( $X^n$ ) (at time  $n$ ).
- Parameters for this model :
  - Initial condition :  $P(X^0)$
  - transition probabilities :  $P(X^n | X^{n-1})$

Equivalent dynamic Bayesian network :



# Markov Chain and dynamic Bayesian network

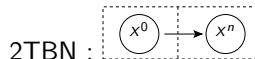
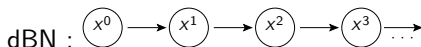


$$P(X^n | X^{n-1}) = \begin{pmatrix} 0.25 & 0 & 0.75 \\ 0.25 & 0.25 & 0.5 \\ 0.25 & 0.5 & 0.25 \end{pmatrix}$$

## Markov chain

- a discrete variable ( $X^n$ ) (at time  $n$ ).
- Parameters for this model :
  - Initial condition :  $P(X^0)$
  - transition probabilities :  $P(X^n | X^{n-1})$

Equivalent dynamic Bayesian network :



Could we do the same with continuous time ?

# Continuous-Time Markov Process

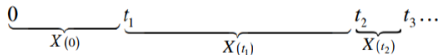
Dynamic processus dynamique verifying :

- a discrete variable

# Continuous-Time Markov Process

Dynamic processus dynamique verifying :

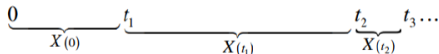
- a discrete variable
- a transition from a state to another can happen **any time**,



# Continuous-Time Markov Process

Dynamic processus dynamique verifying :

- a discrete variable
- a transition from a state to another can happen **any time**,



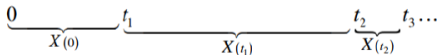
- Continuous time Markov property :

$$\forall s > r, \forall t > 0, P(X(s+t)|X(s), X(r)) = P(X(s+t)|X(s))$$

# Continuous-Time Markov Process

Dynamic processus dynamique verifying :

- a discrete variable
- a transition from a state to another can happen **any time**,



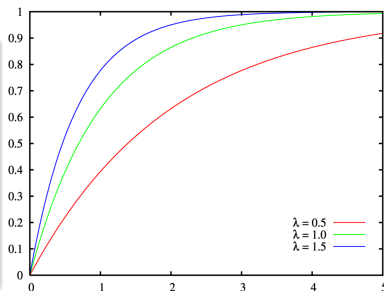
- Continuous time Markov property :

$$\forall s > r, \forall t > 0, P(X(s+t)|X(s), X(r)) = P(X(s+t)|X(s))$$

## Exponential distribution

$$D \sim \text{Exp}(\lambda)$$

- Cumulative distribution function  
 $\forall d > 0, F(d) = 1 - e^{-\lambda d}$
- $\mathbb{E}(D) = \lambda^{-1}$
- $\sigma(D) = \lambda^{-1}$





# Continuous Time Markov Chain (CTMC)

# Continuous Time Markov Chain (CTMC)

A CTMC is a continuous stochastic process in which, for each state, the process will change state according to an exponential random variable and then move to a different state as specified by the probabilities of a stochastic matrix.

# Continuous Time Markov Chain (CTMC)

A CTMC is a continuous stochastic process in which, for each state, the process will change state according to an exponential random variable and then move to a different state as specified by the probabilities of a stochastic matrix.

Minimum of independent exponential distributions

# Continuous Time Markov Chain (CTMC)

A CTMC is a continuous stochastic process in which, for each state, the process will change state according to an exponential random variable and then move to a different state as specified by the probabilities of a stochastic matrix.

## Minimum of independent exponential distributions

$$X \sim \text{Exp}(\lambda), Y \sim \text{Exp}(\mu), X \perp\!\!\!\perp Y \Rightarrow \min(X, Y) \sim \text{Exp}(\lambda + \mu)$$

# Continuous Time Markov Chain (CTMC)

A CTMC is a continuous stochastic process in which, for each state, the process will change state according to an exponential random variable and then move to a different state as specified by the probabilities of a stochastic matrix.

## Minimum of independent exponential distributions

$$X \sim \text{Exp}(\lambda), Y \sim \text{Exp}(\mu), X \perp\!\!\!\perp Y \Rightarrow \min(X, Y) \sim \text{Exp}(\lambda + \mu)$$

## Continuous Time Markov Chain

$(X_t \in x_1, \dots, x_n)_{t \geq 0}$  CdMTC is defined by

# Continuous Time Markov Chain (CTMC)

A CTMC is a continuous stochastic process in which, for each state, the process will change state according to an exponential random variable and then move to a different state as specified by the probabilities of a stochastic matrix.

## Minimum of independent exponential distributions

$$X \sim \text{Exp}(\lambda), Y \sim \text{Exp}(\mu), X \perp\!\!\!\perp Y \Rightarrow \min(X, Y) \sim \text{Exp}(\lambda + \mu)$$

## Continuous Time Markov Chain

$(X_t \in x_1, \dots, x_n)_{t \geq 0}$  CdMTC is defined by

- $P(X_0)$
- $\forall i, j \geq n, q_{i,j}$  tels que
  - 1  $q_{i,i} \in \mathbb{R}^-$
  - 2  $\forall i \neq j, q_{i,j} \in \mathbb{R}^+$
  - 3  $\forall i, \sum_j q_{i,j} = 0$

$$Q_X = \begin{pmatrix} -0.21 & 0.20 & 0.01 \\ 0.05 & -0.10 & 0.05 \\ 0.01 & 0.20 & -0.21 \end{pmatrix}$$

*intensity matrix*

$q_{i,j}$  is the parameter of the exponential distribution controlling the transition from state  $i$  to state  $j$   
 $-q_{i,i}$  is the paramter of the exponential distribution controlling a transition from state  $i$ .

# Properties

$(X_t \in x_1, \dots, x_n)_{t \geq 0}$  CTMC :

- $P(X_0)$
- $Q_X = (q_{i,j})_{i \leq n, j \leq n}$  intensity matrix

# Properties

$(X_t \in x_1, \dots, x_n)_{t \geq 0}$  CTMC :

- $P(X_0)$
- $Q_X = (q_{i,j})_{i \leq n, j \leq n}$  intensity matrix

## Properties

- $p_{i,j} = \frac{q_{i,j}}{-q_{i,i}}$  is the probability of transition from  $x_i$  to  $x_j$



# Properties

$(X_t \in x_1, \dots, x_n)_{t \geq 0}$  CTMC :

- $P(X_0)$
- $Q_X = (q_{i,j})_{i \leq n, j \leq n}$  intensity matrix

## Properties

- $p_{i,j} = \frac{q_{i,j}}{-q_{i,i}}$  is the probability of transition from  $x_i$  to  $x_j$
- $P(X_t) = P(X_0) \cdot \exp(Q_X t)$  with  $\exp(M) = \sum_{n=0}^{\infty} \frac{M^n}{n!}$

# Properties

$(X_t \in x_1, \dots, x_n)_{t \geq 0}$  CTMC :

- $P(X_0)$
- $Q_X = (q_{i,j})_{i \leq n, j \leq n}$  intensity matrix

## Properties

- $p_{i,j} = \frac{q_{i,j}}{-q_{i,i}}$  is the probability of transition from  $x_i$  to  $x_j$
- $P(X_t) = P(X_0) \cdot \exp(Q_X t)$  with  $\exp(M) = \sum_{n=0}^{\infty} \frac{M^n}{n!}$

## Convergence

With some good conditions (ergodicity),  $P(X_t) \xrightarrow[t \rightarrow \infty]{} P(X^*)$

# Properties

$(X_t \in x_1, \dots, x_n)_{t \geq 0}$  CTMC :

- $P(X_0)$
- $Q_X = (q_{i,j})_{i \leq n, j \leq n}$  intensity matrix

## Properties

- $p_{i,j} = \frac{q_{i,j}}{-q_{i,i}}$  is the probability of transition from  $x_i$  to  $x_j$
- $P(X_t) = P(X_0) \cdot \exp(Q_X t)$  with  $\exp(M) = \sum_{n=0}^{\infty} \frac{M^n}{n!}$

## Convergence

With some good conditions (ergodicity),  $P(X_t) \xrightarrow[t \rightarrow \infty]{} P(X^*)$

- Forward sampling :

$\text{draw}(\exp(-q_{i,i}))$  puis  $\text{draw}((p_{i,j}, j \neq i))$ .

# Properties

$(X_t \in x_1, \dots, x_n)_{t \geq 0}$  CTMC :

- $P(X_0)$
- $Q_X = (q_{i,j})_{i \leq n, j \leq n}$  intensity matrix

## Properties

- $p_{i,j} = \frac{q_{i,j}}{-q_{i,i}}$  is the probability of transition from  $x_i$  to  $x_j$
- $P(X_t) = P(X_0) \cdot \exp(Q_X t)$  with  $\exp(M) = \sum_{n=0}^{\infty} \frac{M^n}{n!}$

## Convergence

With some good conditions (ergodicity),  $P(X_t) \xrightarrow[t \rightarrow \infty]{} P(X^*)$

- Forward sampling :  
 $\text{draw}(\exp(-q_{i,i}))$  puis  $\text{draw}((p_{i,j}, j \neq i))$ .
- Convergence of  $P(X_t) = P(X_0) \exp(Q_X t) \xrightarrow[t \rightarrow \infty]{} P^*$

CTBN

## CTBN

$(\mathbb{X}, G, (Q_X)_{X \in \mathbb{X}})$  Continuous-Time Bayesian Network if

## CTBN

$(\mathbb{X}, G, (Q_X)_{X \in \mathbb{X}})$  Continuous-Time Bayesian Network if

- $\mathbb{X} = (X_1, \dots, X_n)$  continuous-time Markov process

## CTBN

$(\mathbb{X}, G, (Q_X)_{X \in \mathbb{X}})$  Continuous-Time Bayesian Network if

- $\mathbb{X} = (X_1, \dots, X_n)$  continuous-time Markov process
- $G$  oriented graph on  $\mathbb{X}$



## CTBN

$(\mathbb{X}, G, (Q_X)_{X \in \mathbb{X}})$  Continuous-Time Bayesian Network if

- $\mathbb{X} = (X_1, \dots, X_n)$  continuous-time Markov process
- $G$  oriented graph on  $\mathbb{X}$  (not DAG)

## CTBN

$(\mathbb{X}, G, (Q_X)_{X \in \mathbb{X}})$  Continuous-Time Bayesian Network if

- $\mathbb{X} = (X_1, \dots, X_n)$  continuous-time Markov process
- $G$  oriented graph on  $\mathbb{X}$  (not DAG)
- $\forall X \in \mathbb{X}, Q_X$  conditional intensity matrix (CIM)

## CTBN

$(\mathbb{X}, G, (Q_X)_{X \in \mathbb{X}})$  Continuous-Time Bayesian Network if

- $\mathbb{X} = (X_1, \dots, X_n)$  continuous-time Markov process
- $G$  oriented graph on  $\mathbb{X}$  (not DAG)
- $\forall X \in \mathbb{X}, Q_X$  conditional intensity matrix (CIM)

$Q_X$  is a CIM  $\iff \forall \langle pa_X \rangle, Q_{X|\langle pa_X \rangle}$  intensity matrix.

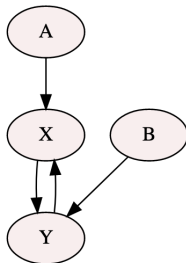
# factorized CTMC : CTBN

## CTBN

$(\mathbb{X}, G, (Q_X)_{X \in \mathbb{X}})$  Continuous-Time Bayesian Network if

- $\mathbb{X} = (X_1, \dots, X_n)$  continuous-time Markov process
- $G$  oriented graph on  $\mathbb{X}$  (not DAG)
- $\forall X \in \mathbb{X}, Q_X$  conditional intensity matrix (CIM)

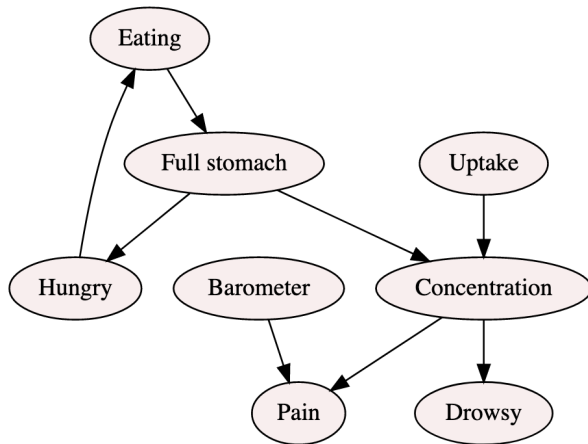
$Q_X$  is a CIM  $\iff \forall \langle pa_X \rangle, Q_{X|\langle pa_X \rangle}$  intensity matrix.



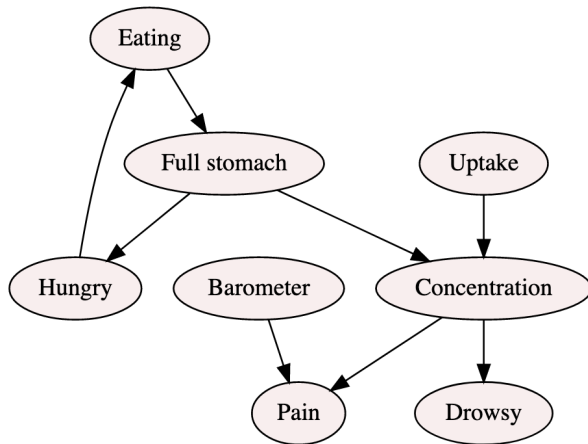
	A#j	
A#i	a0	a1
a0	-1.0000	1.0000
a1	2.0000	-2.0000

		X#j		
Y	A	X#i	x0	x1
y0	a0	x0	-1.0000	1.0000
		x1	2.0000	-2.0000
	a1	x0	-10.0000	10.0000
		x1	20.0000	-20.0000
y1	a0	x0	-5.0000	5.0000
		x1	6.0000	-6.0000
	a1	x0	-50.0000	50.0000
		x1	60.0000	-60.0000

# CTBN - properties

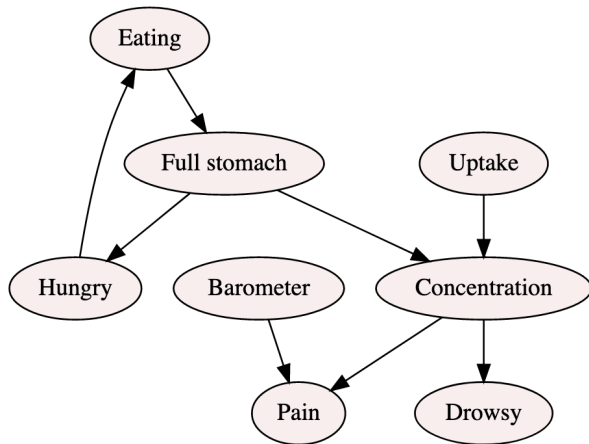


# CTBN - properties



- Arc **always temporal** (which allows 'cycle') :  
 $A \rightarrow B \iff P(B_{t+\delta_t} \mid A_t, \dots) = \dots$

# CTBN - properties

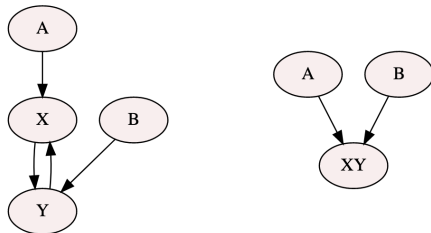


- Arc **always temporal** (which allows 'cycle') :  
$$A \rightarrow B \iff P(B_{t+\delta_t} | A_t, \dots) = \dots$$
- **A Continuous-Time Bayseian network is a joint continuous-time Markov process.**

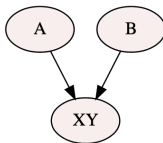
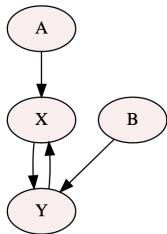
# From a CTBN to the Markov process : amalgamation



# From a CTBN to the Markov process : amalgamation



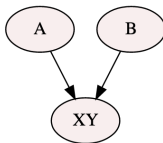
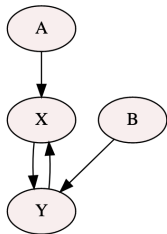
# From a CTBN to the Markov process : amalgamation



$$\mathbf{Q}_{X|a_0, y_0} = \begin{matrix} & x_0 & x_1 \\ x_0 & \begin{pmatrix} -1 & 1 \\ 2 & -2 \end{pmatrix} \end{matrix} \quad \mathbf{Q}_{Y|b_1, x_0} = \begin{matrix} & y_0 & y_1 \\ y_0 & \begin{pmatrix} -3 & 3 \\ 4 & -4 \end{pmatrix} \end{matrix}$$

$$\mathbf{Q}_{X|a_0, y_1} = \begin{matrix} & x_0 & x_1 \\ x_0 & \begin{pmatrix} -5 & 5 \\ 6 & -6 \end{pmatrix} \end{matrix} \quad \mathbf{Q}_{Y|b_1, x_1} = \begin{matrix} & y_0 & y_1 \\ y_0 & \begin{pmatrix} -7 & 7 \\ 8 & -8 \end{pmatrix} \end{matrix}$$

# From a CTBN to the Markov process : amalgamation

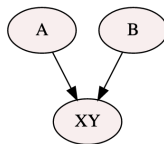
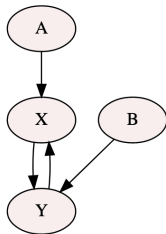


$$Q_{X|a_0, y_0} = \begin{matrix} & x_0 & x_1 \\ \begin{matrix} x_0 \\ x_1 \end{matrix} & \begin{pmatrix} -1 & 1 \\ 2 & -2 \end{pmatrix} \end{matrix} \quad Q_{Y|b_1, x_0} = \begin{matrix} & y_0 & y_1 \\ \begin{matrix} y_0 \\ y_1 \end{matrix} & \begin{pmatrix} -3 & 3 \\ 4 & -4 \end{pmatrix} \end{matrix}$$

$$Q_{X|a_0, y_1} = \begin{matrix} & x_0 & x_1 \\ \begin{matrix} x_0 \\ x_1 \end{matrix} & \begin{pmatrix} -5 & 5 \\ 6 & -6 \end{pmatrix} \end{matrix} \quad Q_{Y|b_1, x_1} = \begin{matrix} & y_0 & y_1 \\ \begin{matrix} y_0 \\ y_1 \end{matrix} & \begin{pmatrix} -7 & 7 \\ 8 & -8 \end{pmatrix} \end{matrix}$$

$$Q_{XY|a_0, b_1} = \begin{matrix} & (x_0, y_0) & (x_0, y_1) & (x_1, y_0) & (x_1, y_1) \\ \begin{matrix} (x_0, y_0) \\ (x_0, y_1) \\ (x_1, y_0) \\ (x_1, y_1) \end{matrix} & \begin{pmatrix} -4 & 3 & 1 & 0 \\ 4 & -9 & 0 & 5 \\ 2 & 0 & -9 & 7 \\ 0 & 6 & 8 & -14 \end{pmatrix} \end{matrix}$$

# From a CTBN to the Markov process : amalgamation



$$Q_{X|a_0, y_0} = \begin{matrix} & x_0 & x_1 \\ \begin{matrix} x_0 \\ x_1 \end{matrix} & \begin{pmatrix} -1 & 1 \\ 2 & -2 \end{pmatrix} \end{matrix} \quad Q_{Y|b_1, x_0} = \begin{matrix} & y_0 & y_1 \\ \begin{matrix} y_0 \\ y_1 \end{matrix} & \begin{pmatrix} -3 & 3 \\ 4 & -4 \end{pmatrix} \end{matrix}$$

$$Q_{X|a_0, y_1} = \begin{matrix} & x_0 & x_1 \\ \begin{matrix} x_0 \\ x_1 \end{matrix} & \begin{pmatrix} -5 & 5 \\ 6 & -6 \end{pmatrix} \end{matrix} \quad Q_{Y|b_1, x_1} = \begin{matrix} & y_0 & y_1 \\ \begin{matrix} y_0 \\ y_1 \end{matrix} & \begin{pmatrix} -7 & 7 \\ 8 & -8 \end{pmatrix} \end{matrix}$$

$$Q_{XY|a_0, b_1} = \begin{matrix} & (x_0, y_0) & (x_0, y_1) & (x_1, y_0) & (x_1, y_1) \\ \begin{matrix} (x_0, y_0) \\ (x_0, y_1) \\ (x_1, y_0) \\ (x_1, y_1) \end{matrix} & \begin{pmatrix} -4 & 3 & 1 & 0 \\ 4 & -9 & 0 & 5 \\ 2 & 0 & -9 & 7 \\ 0 & 6 & 8 & -14 \end{pmatrix} \end{matrix}$$

The amalgamation of all the CIMs of a CTBN produces the intensity matrix of the joint Markov process.

# Forward sampling dans un CTBN

# Forward sampling dans un CTBN

## CTBN Forward sampling

# Forward sampling dans un CTBN

## CTBN Forward sampling

$$\textcircled{1} (X_{t+1}, D_t) = (\textit{argmin}, \min)_{X \in \textit{CTBN}} \textit{Draw}(Q_X)$$

# Forward sampling dans un CTBN

## CTBN Forward sampling

- 1  $(X_{t+1}, D_t) = (\operatorname{argmin}, \min)_{X \in \text{CTBN}} \operatorname{Draw}(Q_X)$
- 2  $x_{t+1} = \operatorname{Draw}(Q_X)$



# Forward sampling dans un CTBN

## CTBN Forward sampling

- 1  $(X_{t+1}, D_t) = (\operatorname{argmin}, \min)_{X \in \text{CTBN}} \operatorname{Draw}(Q_X)$
- 2  $x_{t+1} = \operatorname{Draw}(Q_X)$
- 3 Note  $(X_t, D_t)$

# Forward sampling dans un CTBN

## CTBN Forward sampling

- 1  $(X_{t+1}, D_t) = (\text{argmin}, \min)_{X \in CTBN} \text{Draw}(Q_X)$
- 2  $x_{t+1} = \text{Draw}(Q_X)$
- 3 Note  $(X_t, D_t)$
- 4 loop until stop

# Forward sampling dans un CTBN

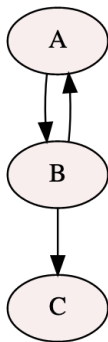
## CTBN Forward sampling

- 1  $(X_{t+1}, D_t) = (\text{argmin}, \min)_{X \in CTBN} \text{Draw}(Q_X)$
- 2  $x_{t+1} = \text{Draw}(Q_X)$
- 3 Note  $(X_t, D_t)$
- 4 loop until stop

# Forward sampling dans un CTBN

## CTBN Forward sampling

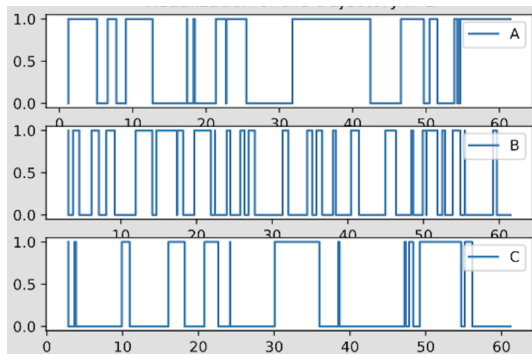
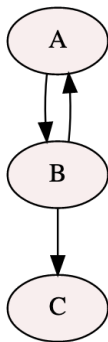
- 1  $(X_{t+1}, D_t) = (\text{argmin}, \min)_{X \in CTBN} \text{Draw}(Q_X)$
- 2  $x_{t+1} = \text{Draw}(Q_X)$
- 3 Note  $(X_t, D_t)$
- 4 loop until stop



# Forward sampling dans un CTBN

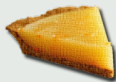
## CTBN Forward sampling

- 1  $(X_{t+1}, D_t) = (\operatorname{argmin}, \min)_{X \in \text{CTBN}} \text{Draw}(Q_X)$
- 2  $x_{t+1} = \text{Draw}(Q_X)$
- 3 Note  $(X_t, D_t)$
- 4 loop until stop





Implémentation (rapide)



# Goal1 : way to define a CTBN

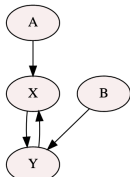
# Goal1 : way to define a CTBN

```
ctbn = Ctnb()
ctbn.add(gum.LabelizedVariable("A", "A", ["a0", "a1"]))
ctbn.add(gum.LabelizedVariable("B", "B", ["b0", "b1"]))
ctbn.add(gum.LabelizedVariable("X", "X", ["x0", "x1"]))
ctbn.add(gum.LabelizedVariable("Y", "Y", ["y0", "y1"]))

ctbn.addArc("A", "X")
ctbn.addArc("Y", "X")
ctbn.addArc("B", "Y")
ctbn.addArc("X", "Y")

ctbn.CIM("A")[:] = [[-1, 1],
                    [2, -2]]
ctbn.CIM("B")[:] = [[-3, 3],
                    [2, -2]]
ctbn.CIM("X") [{"A": "a0", "Y": "y0"}] = [[-1, 1],
                                           [2, -2]]
ctbn.CIM("X") [{"A": "a1", "Y": "y0"}] = [[-1, 1],
                                           [3, -3]]
ctbn.CIM("X") [{"A": "a0", "Y": "y1"}] = [[-5, 5],
                                           [6, -6]]
ctbn.CIM("X") [{"A": "a1", "Y": "y1"}] = [[-5, 5],
                                           [4, -4]]
ctbn.CIM("Y") [{"B": "b0", "X": "x0"}] = [[-2, 2],
                                           [5, -5]]
ctbn.CIM("Y") [{"B": "b1", "X": "x0"}] = [[-3, 3],
                                           [4, -4]]
ctbn.CIM("Y") [{"B": "b0", "X": "x1"}] = [[-3, 3],
                                           [5, -5]]
ctbn.CIM("Y") [{"B": "b1", "X": "x1"}] = [[-7, 7],
                                           [8, -8]]

gnb.sideBySide(ctbn, ctbn.CIM("A")._pot, ctbn.CIM("X")._pot)
```



	A#j	
A#i	a0	a1
a0	-1.0000	1.0000
a1	2.0000	-2.0000

		X#j	
Y	A	X#i	
y0	a0	x0	-1.0000
		x1	-2.0000
	a1	x0	-1.0000
		x1	3.0000
y1	a0	x0	-5.0000
		x1	6.0000
	a1	x0	-5.0000
		x1	8.0000



## Goal2 : implement amalgamation

# Goal2 : implement amalgamation

```
im=CIM()
for x in ctbn.names():
    im*=ctbn.CIM(x)
im.to_matrix()
```

```
array([[ -7.,   1.,   3.,   0.,   1.,   0.,   0.,   0.,   2.,   0.,   0.,
         0.,   0.,   0.,   0.,   0.],
       [  2.,  -8.,   0.,   3.,   0.,   1.,   0.,   0.,   0.,   2.,   0.,
         0.,   0.,   0.,   0.,   0.],
       [  2.,   0.,  -7.,   1.,   0.,   0.,   1.,   0.,   0.,   0.,   3.,
         0.,   0.,   0.,   0.,   0.],
       [  0.,   2.,   2.,  -8.,   0.,   0.,   0.,   1.,   0.,   0.,   0.,
         3.,   0.,   0.,   0.,   0.],
       [  2.,   0.,   0.,   0.,  -9.,   1.,   3.,   0.,   0.,   0.,   0.,
         0.,   3.,   0.,   0.,   0.],
       [  0.,   3.,   0.,   0.,   2., -11.,   0.,   3.,   0.,   0.,   0.,
         0.,   0.,   3.,   0.,   0.],
       [  0.,   0.,   2.,   0.,   2.,   0., -12.,   1.,   0.,   0.,   0.,
         0.,   0.,   0.,   7.,   0.],
       [  0.,   0.,   0.,   3.,   0.,   2.,   2., -14.,   0.,   0.,   0.,
         0.,   0.,   0.,   0.,   7.],
       [  5.,   0.,   0.,   0.,   0.,   0.,   0.,   0., -14.,   1.,   3.,
         0.,   5.,   0.,   0.,   0.],
       [  0.,   5.,   0.,   0.,   0.,   0.,   0.,   0.,   2., -15.,   0.,
         3.,   0.,   5.,   0.,   0.],
       [  0.,   0.,   4.,   0.,   0.,   0.,   0.,   0.,   2.,   0., -12.,
         1.,   0.,   0.,   5.,   0.],
       [  0.,   0.,   0.,   4.,   0.,   0.,   0.,   0.,   0.,   2.,   2.,
        -13.,   0.,   0.,   0.,   5.],
       [  0.,   0.,   0.,   0.,   5.,   0.,   0.,   0.,   6.,   0.,   0.,
         0., -15.,   1.,   3.,   0.],
       [  0.,   0.,   0.,   0.,   0.,   5.,   0.,   0.,   0.,   4.,   0.,
         0.,   2., -14.,   0.,   3.]])
```

## Goal3 : implement the 2 'inference'

# Goal3 : implement the 2 'inference'

```
ie=SimpleCtbnInference(ctbn)
ie.makeInference()
gnb.sideBySide(ie.posterior("A"),ie.posterior("B"),ie.posterior("Y"),ie.posterior("X"))
```

A	
a0	a1
0.6667	0.3333

B	
b0	b1
0.4000	0.6000

Y	
y0	y1
0.6100	0.3900

X	
x0	x1
0.6065	0.3935

```
ie=ForwardSampleCtbnInference(ctbn)
ie.makeInference()
gnb.sideBySide(ie.posterior("A"),ie.posterior("B"),ie.posterior("Y"),ie.posterior("X"))
```

A	
a0	a1
0.6651	0.3349

B	
b0	b1
0.3968	0.6032

Y	
y0	y1
0.6081	0.3919

X	
x0	x1
0.6060	0.3940

# Goal3 : implement the 2 'inference'

```
ie=SimpleCtbnInference(ctbn)
ie.makeInference()
gnb.sideBySide(ie.posterior("A"),ie.posterior("B"),ie.posterior("Y"),ie.posterior("X"))
```

A	
a0	a1
0.6667	0.3333

B	
b0	b1
0.4000	0.6000

Y	
y0	y1
0.6100	0.3900

X	
x0	x1
0.6065	0.3935

```
ie=ForwardSampleCtbnInference(ctbn)
ie.makeInference()
gnb.sideBySide(ie.posterior("A"),ie.posterior("B"),ie.posterior("Y"),ie.posterior("X"))
```

A	
a0	a1
0.6651	0.3349

B	
b0	b1
0.3968	0.6032

Y	
y0	y1
0.6081	0.3919

X	
x0	x1
0.6060	0.3940

# Classes to code

# Classes to code

- CIM

- CIM from `(gum.Potential)`



# Classes to code

- CIM from `(gum.Potential)`
- CTBN,

# Classes to code

- CIM from (`gum.Potential`)
- CTBN, (`gum.DiGraph`, `gum.DiscreteVariable`, `CIM`)

- CIM from (`gum.Potential`)
- CTBN, (`gum.DiGraph`, `gum.DiscreteVariable`, CIM)
- Convergence
  - 'Exact' method ( $\exp(IM)$ )
  - Sampling (Forward Sampling)





# CIM : wrapper de Potential

```
11 def __init__(self, pot=None):
12     if pot is None:
13         self._pot = gum.Potential()
14     else:
15         self._pot = gum.Potential(pot)
16     self._recordVars()
17
18 def add(self, v) → "CIM":
19     self._pot.add(v)
20     self._recordVars()
21     return self
22
23 def nbrDim(self) → int:
24     return self._pot.nbrDim()
25
26 def extract(self, ctxt) → "CIM":
27     return CIM(self._pot.extract(ctxt))
28
29 @property
30 def var_names(self):
31     return self._pot.var_names
32
33 def __getitem__(self, i):
34     return self._pot[i]
35
36 def __setitem__(self, i, v):
37     self._pot[i] = v
```

# CIM : Amalgamation

# CIM : Amalgamation

---

**Algorithm 2.2** Amalgamate two nodes of a CTBN.

---

*Amalgamate*( $X, Y$ )

```
1:  $\mathbf{Q}_{XY|\mathbf{Pa}(XY)} \leftarrow \emptyset$ 
2: for each  $\langle pa_{Y \setminus X} \rangle \in \mathbf{pa}_{X \setminus Y}$  and  $\langle pa_{X \setminus Y} \rangle \in \mathbf{pa}_{Y \setminus X}$ 
3:    $\mathbf{Q}^{XY} \leftarrow 0$ 
4:   for  $i, j = 1, \dots, |X|$  and  $l, k = 1, \dots, |Y|$ 
5:      $\mathbf{Q}^X \leftarrow \mathbf{Q}_{X|\langle pa_{X \setminus Y} \rangle, x_i}$ 
6:      $\mathbf{Q}^Y \leftarrow \mathbf{Q}_{Y|\langle pa_{Y \setminus X} \rangle, y_k}$ 
7:     if  $i = j \wedge k = l$ 
8:        $q_{(i,j),(k,l)}^{XY} \leftarrow q_{i,j}^X + q_{k,l}^Y$ 
9:     else if  $i = j \wedge k \neq l$ 
10:       $q_{(i,j),(k,l)}^{XY} \leftarrow q_{k,l}^Y$ 
11:     else if  $i \neq j \wedge k = l$ 
12:       $q_{(i,j),(k,l)}^{XY} \leftarrow q_{i,j}^X$ 
13:     end if
14:   end for
15:    $\mathbf{Q}_{XY|\langle pa_{XY} \rangle} \leftarrow \mathbf{Q}^{XY}$ 
16:    $\mathbf{Q}_{XY|\mathbf{Pa}(XY)} \leftarrow \mathbf{Q}_{XY|\mathbf{Pa}(XY)} \cup \{\mathbf{Q}_{XY|\langle pa_{XY} \rangle}\}$ 
17: end for
18: return  $\mathbf{Q}_{XY|\mathbf{Pa}(XY)}$ 
```

---



# CIM : Amalgamation

## Algorithm 2.2 Amalgamate two nodes of a CTBN.

*Amalgamate*( $X, Y$ )

```
1:  $\mathbf{Q}_{XY|\mathbf{Pa}(XY)} \leftarrow \emptyset$ 
2: for each  $\langle pa_{Y \setminus X} \rangle \in \mathbf{pa}_{X \setminus Y}$  and  $\langle pa_{X \setminus Y} \rangle \in \mathbf{pa}_{Y \setminus X}$ 
3:    $\mathbf{Q}^{XY} \leftarrow \mathbf{0}$ 
4:   for  $i, j = 1, \dots, |X|$  and  $l, k = 1, \dots, |Y|$ 
5:      $\mathbf{Q}^X \leftarrow \mathbf{Q}_X | \langle pa_{X \setminus Y} \rangle, x_i$ 
6:      $\mathbf{Q}^Y \leftarrow \mathbf{Q}_Y | \langle pa_{Y \setminus X} \rangle, y_k$ 
7:     if  $i = j \wedge k = l$ 
8:        $q_{(i,j),(k,l)}^{XY} \leftarrow q_{i,j}^X + q_{k,l}^Y$ 
9:     else if  $i = j \wedge k \neq l$ 
10:       $q_{(i,j),(k,l)}^{XY} \leftarrow q_{k,l}^Y$ 
11:    else if  $i \neq j \wedge k = l$ 
12:       $q_{(i,j),(k,l)}^{XY} \leftarrow q_{i,j}^X$ 
13:    end if
14:   end for
15:    $\mathbf{Q}_{XY|\langle pa_{XY} \rangle} \leftarrow \mathbf{Q}^{XY}$ 
16:    $\mathbf{Q}_{XY|\mathbf{Pa}(XY)} \leftarrow \mathbf{Q}_{XY|\mathbf{Pa}(XY)} \cup \{\mathbf{Q}_{XY|\langle pa_{XY} \rangle}\}$ 
17: end for
18: return  $\mathbf{Q}_{XY|\mathbf{Pa}(XY)}$ 
```

```
175 i = amal.instantiation()
176 iX = cimX.instantiation()
177 iY = cimY.instantiation()
178 i.setFirst()
179 while not i.end():
180     iX.setVals(i)
181     iY.setVals(i)
182     for v in pXinY:
183         iY.chgVal(v, iX[CIM.var_i(v.name())])
184     for v in pYinX:
185         iX.chgVal(v, iY[CIM.var_i(v.name())])
186     dX = True
187     for v in sX:
188         if iX[CIM.var_i(v)] != iX[CIM.var_j(v)]:
189             dX = False
190             break
191     dY = True
192     for v in sY:
193         if iY[CIM.var_i(v)] != iY[CIM.var_j(v)]:
194             dY = False
195             break
196     if dX and dY:
197         amal[i] = cimX[iX] + cimY[iY]
198     elif dY:
199         amal[i] = cimX[iX]
200     elif dX:
201         amal[i] = cimY[iY]
202     else:
203         amal[i] = 0
204
205     i.inc()
206
207 return amal
```

# Implementations

- CIM : 210 lines
- CTBN
- Inférence
  - Simple
  - Sampling

# Implementations

- CIM : 210 lines
- CTBN
- Inférence
  - Simple
  - Sampling

Mainly : synchronization of 3 very different objects :

Mainly : synchronization of 3 very different objects :

- oriented graph  
(`gum.DiGraph`),

Mainly : synchronization of 3 very different objects :

- oriented graph  
(`gum.DiGraph`),
- Discrete random variables in a dictionary  
(`gum.DiscreteVariable`)

Mainly : synchronization of 3 very different objects :

- oriented graph  
(`gum.DiGraph`),
- Discrete random variables in a dictionary  
(`gum.DiscreteVariable`)
- CIMs in a dictionary.

Mainly : synchronization of 3 very different objects :

- oriented graph  
(gum.DiGraph),
- Discrete random variables in a dictionary  
(gum.DiscreteVariable)
- CIMs in a dictionary.

```
def __init__(self):
    self.graph = gum.DiGraph()
    self.cim = {}
    self.id2var = {}
    self.name2id = {}
```

```
def add(self, var: gum.DiscreteVariable) → NodeId:
    n = NodeId(self.graph.addNode())
    self.id2var[n] = var
    self.name2id[var.name()] = n

    self.bn0.add(var)

    v_i = var.clone()
    v_i.setName(CIM.var_i(var.name()))

    v_j = var.clone()
    v_j.setName(CIM.var_j(var.name()))

    self.cim[n] = CIM().add(v_j).add(v_i)

    return n
```

```
def addArc(self, val1: NameOrId, val2: NameOrId) → Tuple[NodeId, NodeId]:
    n1 = self._nameOrId(val1)
    n2 = self._nameOrId(val2)

    self.graph.addArc(n1, n2)

    self.cim[n2].add(self.id2var[n1])

    return (n1, n2)
```



# Implementations

- CIM : 210 lines
- CTBN : 190 lines
- Inférence
  - Simple
  - Sampling

# Implementations

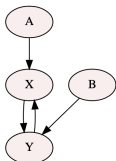
- CIM : 210 lines
- CTBN : 190 lines
- Inférence
  - Simple
  - Sampling

```
ctbn = Ctbn()
ctbn.add(gum.LabelizedVariable("A", "A", ["a0", "a1"]))
ctbn.add(gum.LabelizedVariable("B", "B", ["b0", "b1"]))
ctbn.add(gum.LabelizedVariable("X", "X", ["x0", "x1"]))
ctbn.add(gum.LabelizedVariable("Y", "Y", ["y0", "y1"]))

ctbn.addArc("A", "X")
ctbn.addArc("Y", "X")
ctbn.addArc("B", "Y")
ctbn.addArc("X", "Y")

ctbn.CIM("A")[:] = [[-1, 1],
                    [2, -2]]
ctbn.CIM("B")[:] = [[-3, 3],
                    [2, -2]]
ctbn.CIM("X")[{ "A": "a0", "Y": "y0" }] = [[-1, 1],
                                             [2, -2]]
ctbn.CIM("X")[{ "A": "a1", "Y": "y0" }] = [[-1, 1],
                                             [3, -3]]
ctbn.CIM("X")[{ "A": "a0", "Y": "y1" }] = [[-5, 5],
                                             [6, -6]]
ctbn.CIM("X")[{ "A": "a1", "Y": "y1" }] = [[-5, 5],
                                             [4, -4]]
ctbn.CIM("Y")[{ "B": "b0", "X": "x0" }] = [[-2, 2],
                                             [5, -5]]
ctbn.CIM("Y")[{ "B": "b1", "X": "x0" }] = [[-3, 3],
                                             [4, -4]]
ctbn.CIM("Y")[{ "B": "b0", "X": "x1" }] = [[-3, 3],
                                             [5, -5]]
ctbn.CIM("Y")[{ "B": "b1", "X": "x1" }] = [[-7, 7],
                                             [8, -8]]

gnb.sideBySide(ctbn, ctbn.CIM("A")._pot, ctbn.CIM("X")._pot)
```



	A#j	
A#i	a0	a1
a0	-1.0000	1.0000
a1	2.0000	-2.0000

		X#j	
Y	A	X#i	
y0	a0	x0	-1.0000 1.0000
		x1	2.0000 -2.0000
	a1	x0	-1.0000 1.0000
		x1	3.0000 -3.0000
y1	a0	x0	-5.0000 5.0000
		x1	6.0000 -6.0000
	a1	x0	-5.0000 5.0000
		x1	4.0000 -4.0000

# Inférence simple

# Inférence simple

```
26 class SimpleCtbnInference(CtbnInference):
27     """
28     Exact inference using amalgamation to compute the Intensity Matrix corresponding to the ctbn
29     (very bad for large models)
30     """
31
32     def __init__(self, cim: Ctbn):
33         super().__init__(cim)
34         self._joint = None
35
36     def makeInference(self):
37         q = CIM()
38         for nod in self._model.nodes():
39             q = q.amalgamate(self._model.CIM(nod))
40
41         q.from_matrix(expm(5000 * q.to_matrix()))
42
43         t0 = gum.Potential()
44         for n in q.var_names:
45             if n[-1] == "i":
46                 t0.add(q._pot.variable(n))
47             t0.fillWith(1).normalize()
48
49         self._joint = (t0 * q._pot).margSumOut(t0.var_names)
50
51     def posterior(self, name: str) → gum.Potential:
52         vj = CIM.var_j(name)
53         return gum.Potential().add(self._model.variable(name)).fillWith(self._joint.margSumIn(vj), [vj])
```

# Implémentations

- CIM : 210 lines
- CTBN : 190 lines
- Inférence
  - Simple : 23 lines
  - Sampling

# Forward Sampling

# Forward Sampling

---

**Algorithm 2.1** Forward sample CTBN.

---

*ForwardSample*( $\mathcal{N}$ )

```
1: for each  $X \in \mathcal{N}$ 
2:   choose  $X(0)$  by sampling from  $\mathcal{B}$ 
3: end for
4:  $t \leftarrow 0, \sigma \leftarrow \emptyset$ 
5: repeat until termination
6:   Append( $\sigma, \langle X, t \rangle$ )
7:   for each  $X \in \mathcal{N}$ 
8:     if  $\text{time}(X) \neq \text{null}$  then continue end for
9:      $\mathbf{A}_X \leftarrow \mathbf{A}_{X|\mathbf{Pa}(X)}$ 
10:     $i \leftarrow X(t)$ 
11:     $\Delta t \sim \text{Exponential}(a_{i,i})$ 
12:     $\text{time}(X) \leftarrow t + \Delta t$ 
13:   end for
14:    $X' \leftarrow \operatorname{argmin}_{X \in \mathcal{N}} (\text{time}(X))$ 
15:    $t \leftarrow \text{time}(X')$ 
16:    $X(t) \sim \text{Multinomial}(\mathbf{A}_{X'}, X(t))$ 
17:    $\text{time}(X') = \text{null}$ 
18:   for each  $Y \in \text{Ch}(X')$ 
19:      $\text{time}(Y) = \text{null}$ 
20:   end for
21: end repeat
22: return  $\sigma$ 
```

---

# Forward Sampling

---

**Algorithm 2.1** Forward sample CTBN.

---

*ForwardSample*( $\mathcal{N}$ )

```
1: for each  $X \in \mathcal{N}$ 
2:   choose  $X(0)$  by sampling from  $\mathcal{B}$ 
3: end for
4:  $t \leftarrow 0, \sigma \leftarrow \emptyset$ 
5: repeat until termination
6:   Append( $\sigma, \langle X, t \rangle$ )
7:   for each  $X \in \mathcal{N}$ 
8:     if  $\text{time}(X) \neq \text{null}$  then continue end for
9:      $\mathbf{A}_X \leftarrow \mathbf{A}_{X|\mathbf{Pa}(X)}$ 
10:     $i \leftarrow X(t)$ 
11:     $\Delta t \sim \text{Exponential}(a_{i,i})$ 
12:     $\text{time}(X) \leftarrow t + \Delta t$ 
13:   end for
14:    $X' \leftarrow \operatorname{argmin}_{X \in \mathcal{N}} (\text{time}(X))$ 
15:    $t \leftarrow \text{time}(X')$ 
16:    $X(t) \sim \text{Multinomial}(\mathbf{A}_{X'}, X(t))$ 
17:    $\text{time}(X') = \text{null}$ 
18:   for each  $Y \in \text{Ch}(X')$ 
19:      $\text{time}(Y) = \text{null}$ 
20:   end for
21: end repeat
22: return  $\sigma$ 
```

---



# Conclusion

- CIM : 210 lines
- CTBN : 190 lines
- Inférence
  - Simple : 23 lines
  - Sampling 90 lines

# Conclusion

- CIM : 210 lines
- CTBN : 190 lines
- Inférence
  - Simple : 23 lines
  - Sampling 90 lines

# Conclusion

- CIM : 210 lines
- CTBN : 190 lines
- Inférence
  - Simple : 23 lines
  - Sampling 90 lines

*Code based on the result of a student project (L2).*

# Conclusion

- CIM : 210 lines
- CTBN : 190 lines
- Inférence
  - Simple : 23 lines
  - Sampling 90 lines

*Code based on the result of a student project (L2).*

goals reached !

# Conclusion

- CIM : 210 lines
- CTBN : 190 lines
- Inférence
  - Simple : 23 lines
  - Sampling 90 lines

*Code based on the result of a student project (L2).*

## goals reached !

- model CTBN : compact representation of continuous-time Markov processes with a very large state space .

# Conclusion

- CIM : 210 lines
- CTBN : 190 lines
- Inférence
  - Simple : 23 lines
  - Sampling 90 lines

*Code based on the result of a student project (L2).*

## goals reached !

- model CTBN : compact representation of continuous-time Markov processes with a very large state space .
- pyAgrum : toolbox for the implementation of new graphical models.

# And now ?

- aGrUM/pyAgrum still a lab/academic tool. We will not stop maintaining & developing !
- Many users imply many responsibilities
  - Interaction  
gitlab issues, discord, gitter, linkedin, researchGate, what else ?
  - Structuration  
communaute ( ? ), consortium ( ? )
  - Scientific orientation ?
    - models
    - algorithms
    - scientific committee
    - ?
  - Development orientation ?
    - weaknesses, strengths
    - missing features
    - Ragrum, JSagrum
    - Steering committee
    - ?