Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Software update

PyBanshee version (1.0): A Python implementation of the MATLAB toolbox BANSHEE for Non-Parametric Bayesian Networks with updated features

Paul Koot [a,d], Miguel Angel Mendoza-Lugo [a,*], Dominik Paprotny [b], Oswaldo Morales-Nápoles [a], Elisa Ragno [a], Daniël T.H. Worm [c]

[a] *Delft University of Technology, Department of Hydraulic Engineering, Stevinweg 1, 2628CN Delft, The Netherlands*
[b] *Potsdam Institute for Climate Impact Research, Research Department Transformation Pathways, Telegrafenberg, 14473 Potsdam, Germany*
[c] *TNO, Cyber Security & Robustness, Anna van Buerenplein 1, 2595DA The Hague, The Netherlands*
[d] *Nelen & Schuurmans B.V., Zakkendragershof 34-44 3511AE Utrecht, The Netherlands*

A R T I C L E   I N F O

Article history:
Received 19 October 2021
Received in revised form 9 November 2022
Accepted 21 November 2022

Keywords:
Non-Parametric Bayesian Networks
BANSHEE
Python library
Copulas

A B S T R A C T

In this paper we discuss PyBanshee, which is a Python-based open-source implementation of the MATLAB toolbox BANSHEE. PyBanshee constitutes the first fully open-source package to quantify, visualize and validate Non-Parametric Bayesian Networks (NPBNs). The architecture of PyBanshee is heavily based on its MATLAB predecessor. It presents the full implementation of existing tools and introduces new modules. Specifically, PyBanshee allows for: (i) choosing fully parametric one-dimensional margins, (ii) choosing different sample sizes for the model-validation tests based on the Hellinger distance, (iii) drawing user-defined sample sizes of the NPBN, (iv) sample-based conditioning sampling (similarly to the closed-source proprietary package UNINET by LightTwist Software) and (v) visualizing the comparison between the histograms of the unconditional and conditional marginal distributions. New detailed examples demonstrating new features are provided.

Code metadata

Current code version	PyBanshee v1.0, Paper v1.0
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-21-00202
Code Ocean compute capsule	–
Legal Code License	GNU General Public License
Code versioning system used	None
Software code languages, tools, and services used	Python, SciPy, NumPy, Matplotlib, PyCopula
Compilation requirements, operating environments & dependencies	Python version 3.6
If available Link to developer documentation/manual	https://github.com/mike-mendoza/py_banshee
Support email for questions	m.a.mendozalugo@tudelft.nl

Software metadata

Current software version	PyBanshee v1.0
Permanent link to executables of this version	https://github.com/mike-mendoza/py_banshee
Legal Software License	GNU General Public License
Computing platforms/Operating Systems	Microsoft Windows and macOS
Installation requirements & dependencies	Python version 3.6, including SciPy, NumPy, Matplotlib, PyCopula libraries
If available, link to user manual - if formally published include a reference to the publication in the reference list	https://github.com/mike-mendoza/py_banshee
Support email for questions	m.a.mendozalugo@tudelft.nl

DOI of original article: https://doi.org/10.1016/j.softx.2020.100588.
* Corresponding author.
E-mail addresses: paulkoot6@gmail.com (Paul Koot), m.a.mendozalugo@tudelft.nl (Miguel A. Mendoza-Lugo), paprotny@gfz-potsdam.de (Dominik Paprotny), o.moralesnapoles@tudelft.nl (Oswaldo Morales-Napoles), E.Ragno@tudelft.nl (Elisa Ragno), daniel.worm@tno.nl (Daniël T.H. Worm).

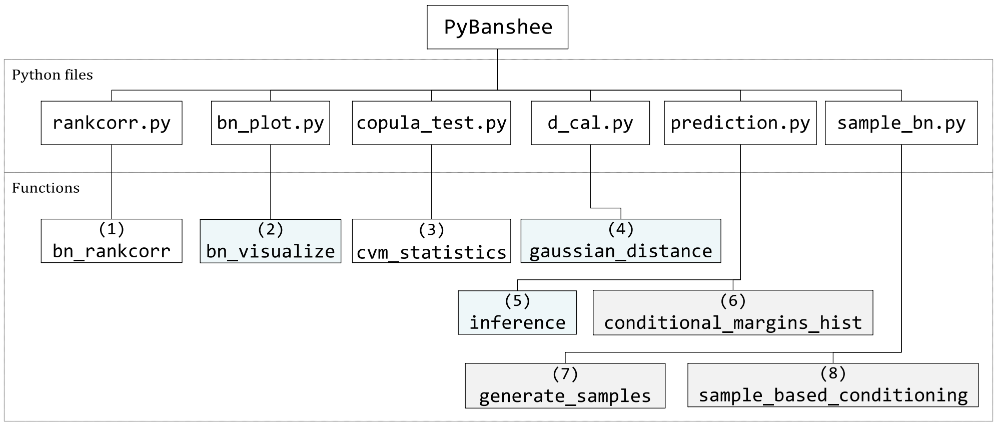

Fig. 1. PyBanshee structure. Functions (1) to (5) correspond to the main functions of the previous MATLAB release. Changes were made in functions (2), (4) and (5) (boxes filled in light-blue) to add feature (v), feature (ii) and feature (i) correspondingly. Functions (6), (7), and (8) (boxes filled in light-gray) correspond to the newly added functions (features (v), (iii) and (iv) respectively).

1. Motivation and significance

A MATLAB toolbox called BANSHEE implementing Non-Parametric Bayesian Networks (NPBNs) was presented in [1]. While BANSHEE is the first openly available tool for Non-Parametric Bayesian Networks it requires access to the commercial MATLAB environment. Hence, in order to further increase the availability of the tool, we introduce PyBanshee. PyBanshee is the Python-based version of MATLAB BANSHEE. In fact, our main motivation to introduce the Python version is to make our software fully open source so that it can be more accessible to non-MATLAB licensed users. In addition to the functionalities available in our first release (MATLAB BANSHEE), PyBanshee allows for the use of (i) fully parametric one-dimensional margins, (ii) user-defined sample sizes for model-validation tests based on the Hellinger distance, (iii) getting user-defined sample sizes of the NPBN, (iv) sample based conditioning and (v) advanced graphics for the underlying graph of the Bayesian Network and for the visualization of the comparison between the histograms of the unconditional and conditional marginal distributions.

Feature (i) is useful when the interest is in values that have not been observed in a particular sample. For example in civil engineering percentiles with a low probability of exceedance are often required for design purposes. These are obtained by extrapolation using a parametric one-dimensional margin. Although this feature would make the approach parametric, we still use the term NPBN to be consistent with previous literature [2]. Regarding (ii), since model validation is still in its infancy when it comes to NPBNs, it is often desired to estimate the minimum sample size at which the null hypothesis would not be rejected in order to give some guidance about statistical power. Feature (iii) is useful when statistically robust results are needed. This can be done by increasing the number of samples drawn from the NPBN. Sample based conditioning (feature (iv)) is implemented to allow for conditionalizing on intervals. This feature is also available in commercial software UNINET [3] and is often required in civil engineering applications. Advanced graphics (feature (v)) are implemented in order to facilitate further analysis for users.

2. Software description

Similarly to its MATLAB predecessor, the code of PyBanshee consists of a set of functions. These functions allow for quantifying the NPBN, analyzing the underlying assumptions of the model, visualizing the network and its corresponding rank correlation matrix, and making inferences with an NPBN based on existing observations or new evidence. In addition, new functions were added to: generate random samples of the NPBN, perform sample-based conditioning and visualize unconditional and conditional distributions. To show and better explain the features of PyBanshee, examples are provided as standalone scripts.

2.1. Software architecture

The PyBanshee package contains six Python files (`.py` extension files) in which eight main functions are located as can be seen in Fig. 1. The main functions are:

(1) `bn_rankcorr`, to compute the BN rank correlation matrix. It requires a defined directed acyclic graph (DAG, as a `list`) and a matrix of data (as a `DataFrame`). Additionally, it includes an option to compute the BN rank correlation without such data, which will be exemplified in Section 3;
(2) `bn_visualize`, to visualize the structure of the defined NPBN. In this release, an NPBN with marginal histograms inside of the nodes can be displayed (feature (v) presented in Section 1);
(3) `cvm_statistics`, to measure the degree of agreement of the Gaussian copula with the data based on the Cramer–von Mises statistic;
(4) `gaussian_distance`, to perform model validation based on the distance between the empirical (ERC) and Bayesian Network's rank correlation (BNRC) matrices and the empirical normal rank correlation matrix (NRC) using data. The distance is computed based on the d-calibration score [4,5]. PyBanshee allows users to define different sample sizes (feature (ii) presented in Section 1);
(5) `inference`, to compute the uncertainty distribution of nodes other than those that the user conditionalized on. In this release, parametric distributions can be an input for the `inference` function (feature (i) presented in Section 1);
(6) `conditional_margins_hist`, to visualize the comparison between unconditional and conditional histograms (feature (v) presented in Section 1) of the `inference` function output;
(7) `generate_samples`, to explicitly increase the number of unconditional samples of the NPBN (feature (iii) presented in Section 1);
(8) `sample_based_conditioning`, to conditionalize nodes on intervals (feature (iv) presented in Section 1).

Table 1
One-dimensional fitted distributions.

Variable	Distribution	Shape	Location	Scale
W	Normal	–	192.52	29.33
AX1	Generalized extreme value	0.16	54.48	10.96
AX2	Normal	–	87.93	12.84
AX3	Generalized extreme value	0.085	41.53	8.95

For more detailed information on functions (1) to (5), the reader is referred to the original paper [1].

2.2. Software functionalities

In this paper we focus on the Python-based implementation, ensuring that users familiar with the MATLAB version will recognize the new features. The users can develop and validate an NPBN using the same workflow as in the previous MATLAB release i.e.: (I) load a dataset of interest (DATA); (II) define a DAG in PARENTCELL (list) based on the dependence between the variables; (III) compute the BN rank correlation matrix R with the function `bn_rankcorr` and (IV) obtain the joint distribution function of the variables and the conditional distribution of a variable given the remaining using the `inference` function. The following section will present illustrative examples. In their corresponding scripts, all functions are listed and detailed descriptions of each step of the procedure are included.

3. Illustrative examples

Example 1. Parametric one-dimensional margins.

The first script, `example_1`, estimates the distributions of individual axle loads for a hypothetical 400 kN three axle vehicle. We used the Weigh-in-Motion (WIM) dataset of the Dutch A12 motorway in the left driving direction presented in [6,7]. The WIM dataset contains data that describe the weight and length of the observed vehicles in April 2013. For this example, the variables of interest for a hypothetical three axle vehicle NPBN are: total vehicle weight (W), first axle load ($AX1$), second axle load ($AX2$) and third axle load ($AX3$). Each of the variables was fitted to a one-dimensional parametric distribution (see Table 1). The DAG corresponding to the NPBN is presented in Fig. 2(a). The (conditional) rank correlations between variables are the input of the `bn_rankcorr` function and this function is applied to compute an NPBN rank correlation matrix for the defined DAG (Fig. 2(b)). Once the NPBN has been built, estimation of the conditional distributions of individual axle loads given a total vehicle weight of 400 kN can be performed using the inference function (`inference`) passing the fitted parametric one-dimensional margins as an input (feature (i) presented in Section 1). A comparison between the unconditional and conditional marginal distributions (Figs. 2(c)–2(e)) can be presented, using `conditional_margins_hist` function (feature (v) presented in Section 1).

Example 2. Sample based conditioning.

To graphically show the accuracy of PyBanshee, the second example (script `example_2`) reproduces an NPBN which was built in the uncertainty analysis software package UNINET presented in [8]. The NPBN consists of 18 variables representing traffic loads, ground motion and bridge column structural response information. The name and description of each variable are presented in table S1. In particular, we analyzed the conditional case that represents a scenario with high vehicle loads and low material resistances during a "mid-intensity" ground motion.[1] For this example, empirical data is used. First, the structure of the NPBN is defined, and then the rank correlation matrix is computed via the `bn_rankcorr` function. Next, with the function `bn_visualize`, two kinds of DAG can be plotted: (i) an NPBN in which nodes are ellipses with the name of the variables and (ii) an NPBN with the marginals displayed inside of the nodes. Both DAGs display conditional rank correlations on the arcs (see figs. S1 and S2).

The function `cvm_statistic` tests the validity of the Gaussian copula assumption (see fig. S5). Figure S4 shows the d-calibration scores of the empirical rank correlation matrix (see fig. S3) and the NPBN's rank correlation matrix computed with `gaussian_distance` function. These scores support the assumptions of a joint normal copula used in the BN model. As stated in the introduction, model validation for NPBNs is still in its infancy. It should be noted that the tests proposed are sensitive to the number of samples drawn in each bootstrap step as well as the number of iterations (bootstraps). The tests are rather severe, especially for large datasets [2,4]. Unlike the previous MATLAB release, the number of drawn samples for each test can be different (feature (ii) presented in Section 1).

To obtain statistically robust results, we may increase the number of samples of the NPBN. This can be done by using the function `sample_bn` (feature (iii) presented in Section 1). Next, with the `sample_based_conditioning` function (feature (iv) presented in Section 1) we conditionalized on the corresponding intervals of the scenario under investigation (see table S2). The results are the sample based conditional empirical distributions of the output nodes. Finally, to graphically compare the output of PyBanshee with UNINET. Figure S6 shows the empirical cumulative distribution (ECDF) of the sample based conditional samples from UNINET and the sample based conditional samples computed by PyBanshee. As can be seen, the ECDFs are in good alignment. Slight differences can be observed in the output variables. However, these differences are within the uncertainty due to statistical fluctuation in the sampling methods.

4. Final comments

We released PyBanshee, an open source Python porting of our previously-released MATLAB software package for Non-parametric Bayesian Networks with extended functionalities. The extensions made include the possibility to use parametric one-dimensional margins, to choose different sample sizes for the model-validation tests and a function for sample based conditioning. These features are also implemented in the MATLAB version.[2] The original MATLAB version included three 'real-life' models from the field of geosciences in the original release. Since then, in addition to the newly added functions, two scripts were added from recently published research namely flood damage models for residential and commercial assets. Further details can be found in [1,9,10]. We added new visualizations, which are only accessible in PyBanshee, recently used in [11].

The next step for PyBanshee is merging with our previous application for structured expert judgment ANDURYL (see [12–14]) in order to allow for the quantification of Non-parametric Bayesian Networks through structured expert judgments.

[1] According to [8], a "mid-intensity" ground motion is a ground motion with a maximum ground acceleration between 0.273 g and 0.783 g.

[2] The upgraded MATLAB BANSHEE (v1.3) is located at https://github.com/mike-mendoza/matlab_banshee.

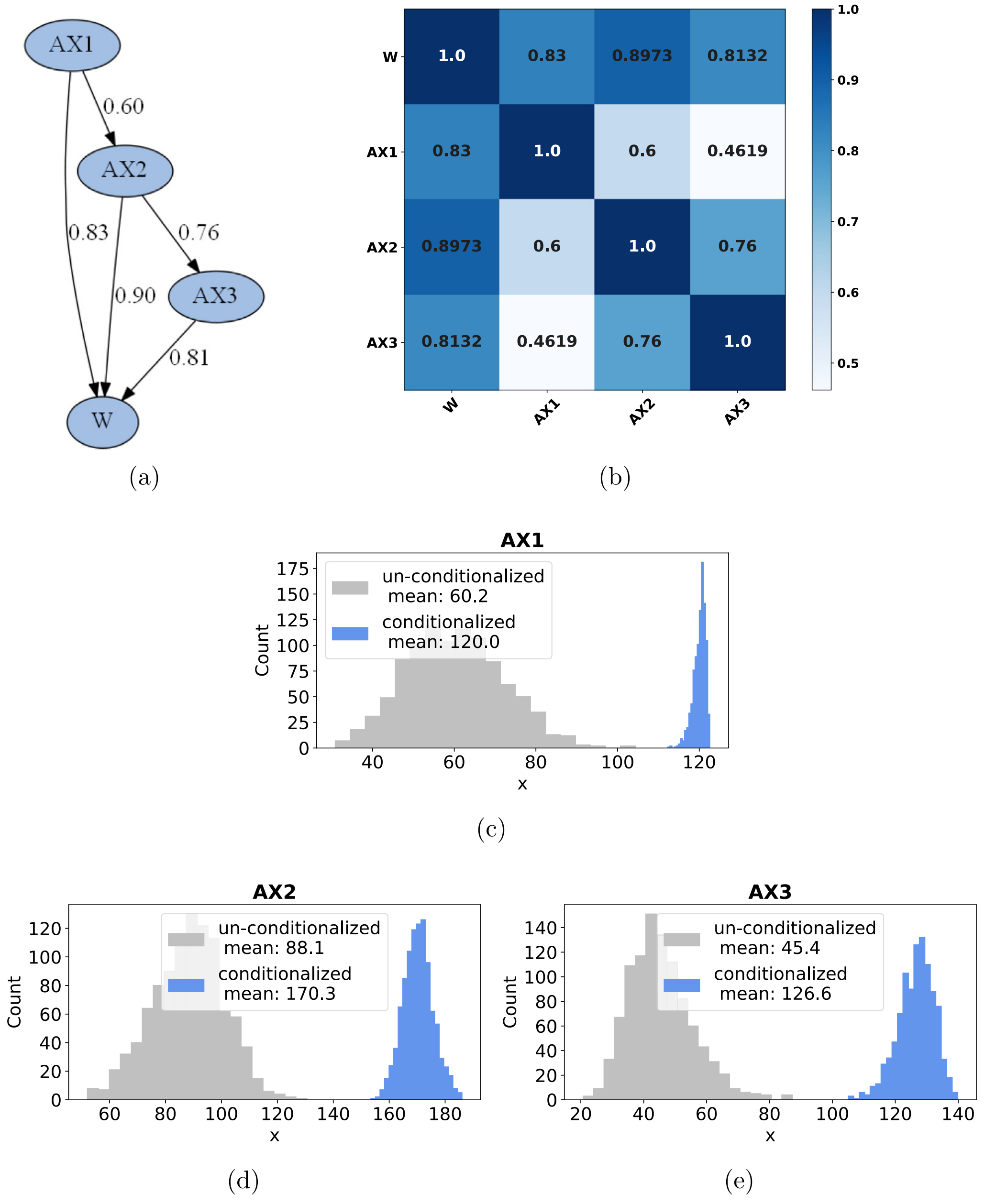

Fig. 2. Plots generated with the PyBanshee functions and WIM dataset: (a) BN structure (`bn_visualize`); (b) NPBN rank correlation matrix (`bn_rankcorr`); (c–e) unconditional and conditional marginal histograms comparison (`conditinal_margins_hist`).

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This research was partially supported by the Mexican National Council for Science and Technology (CONACYT) under project number 2019-000021-01EXTF-00564 CVU 784544 and also partially supported by Climate-KIC through project "SAFERPLACES", Task ID TC2018B_4.7.3-SAFERPL_P430-1 A KAVA2 4.7.3. The authors thank E. Aparicio Medrano and I. Lokhorst for their useful comments.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.softx.2022.101279.

References

[1] Paprotny Dominik, Morales-Nápoles Oswaldo, Worm Daniël TH, Ragno Elisa. BANSHEE–A MATLAB toolbox for non-parametric Bayesian networks. SoftwareX 2020;12:100588.

[2] Hanea Anca, Morales Napoles Oswaldo, Ababei Dan. Non-parametric Bayesian networks: Improving theory and reviewing applications. Reliab Eng Syst Saf 2015;144:265–84.

[3] LightTwist Software. uninet.

[4] Morales-Nápoles O, Hanea A M, Worm DT H. Experimental results about the assessments of conditional rank correlations by experts: Example with air pollution estimates. In: Safety, reliability and risk analysis: beyond the horizon - proceedings of the European safety and reliability conference. 2014.

[5] Werner C, Bedford T, Cooke RM, Hanea AM, Morales-Nápoles O. Expert judgement for dependence in probabilistic modelling: A systematic literature review and future research directions. European J Oper Res 2017;258(3):801–19.

[6] Morales-Nápoles Oswaldo, Steenbergen Raphaël DJM. Analysis of axle and vehicle load properties through Bayesian networks based on weigh-in-motion data. Reliab Eng Syst Saf 2014;125:153–64, Special issue of selected articles from ESREL 2012.

[7] Morales-Nápoles O, Steenbergen RDJM. Large-scale hybrid Bayesian network for traffic load modeling from weigh-in-motion system data. J Bridge Eng 2015;20(1).

[8] Mendoza-Lugo Miguel Angel, Delgado-Hernández David Joaquín, Morales-Nápoles Oswaldo. Reliability analysis of reinforced concrete vehicle bridges columns using non-parametric Bayesian networks. Eng Struct 2019;188:178–87.

[9] Paprotny Dominik, Kreibich Heidi, Morales-Nápoles Oswaldo, Wagenaar Dennis, Castellarin Attilio, Carisi Francesca, et al. A probabilistic approach to estimating residential losses from different flood types. Nat Hazards 2021;105(3):2569–601.

[10] Paprotny Dominik, Kreibich Heidi, Morales-Nápoles Oswaldo, Castellarin Attilio, Carisi Francesca, Schröter Kai. Exposure and vulnerability estimation for modelling flood losses to commercial assets in Europe. Sci Total Environ 2020;737:140011.

[11] Mendoza-Lugo Miguel Angel, Morales-Nápoles Oswaldo, Delgado-Hernández David Joaquín. A non-parametric Bayesian network for multivariate probabilistic modelling of weigh-in-motion system data. Transp Res Interdiscip Perspect 2022;13:100552.

[12] Leontaris Georgios, Morales-Nápoles Oswaldo. ANDURIL — A MATLAB toolbox for analysis and decisions with Uncertainty: Learning from expert judgments. SoftwareX 2018;7:313–7.

[13] Hart Cornelis Marcel Pieter 't, Leontaris Georgios, Morales-Nápoles Oswaldo. Update (1.1) to ANDURIL — A MATLAB toolbox for analysis and decisions with UnceRtaInty: Learning from expert judgments. SoftwareX 2019;10:100295.

[14] Rongen Guus, Hart Cornelis Marcel Pieter 't, Leontaris Georgios, Morales-Nápoles Oswaldo. Update (1.2) to ANDURIL and ANDURYL: Performance improvements and a graphical user interface. SoftwareX 2020;12:100497.