Practicalities of Bayesian network modeling for nuclear data evaluation with the nucdataBaynet package

Georg Schnabel[1,*]

[1]NAPC-Nuclear Data Section, International Atomic Energy Agency, A-1040 Vienna, Austria

Abstract. Bayesian networks are a helpful abstraction in the modelization of the relationships between different variables for the purpose of uncertainty quantification. They are therefore especially well suited for the application to nuclear data evaluation to accurately model the relationships of experimental and nuclear models. Constraints, such as sum rules and the non-negativity of cross sections, can be rigorously taken into account in Bayesian inference within Bayesian networks. This contribution elaborates on the practical aspects of the construction of Bayesian networks with the *nucdataBaynet* package for the purpose of nuclear data evaluation.

1 Introduction

In nuclear data evaluation, experimental data and nuclear models are combined by means of a statistical procedure to obtain estimates and uncertainties for quantities of interest, such as cross sections.

The Generalized Least Squares (GLS) method, e.g., [1], implemented in many evaluation codes, such as GANDR [2] and GMA [3], provides a sound framework for the evaluation of nuclear data as long as relationships between variables can be in good approximation linearized. Many of such evaluation codes that are employed above the resonance region do not take into account experimental features, such as resolution broadening or uncertainties in the incident energy, because their influence on evaluation results is usually negligible.

In contrast, the statistical modules employed by R-Matrix codes, such as SAMMY [4], must consider these experimental features in order to properly forward propagate theoretical cross sections to quantities comparable with high-resolution experiments. Relationships in the resonance region are highly non-linear. Consequently the GLS method is not applicable and must be replaced by iterative procedures, which typically use the GLS method in each iteration to guide the optimization process.

The division between the resonance region and the fast energy range is first and foremost due to the physics model, i.e., R-matrix or optical model. From the statistical point of view, these two regions can be treated in the same framework if non-linear relationships are properly taken into account. Furthermore, even within the fast energy range, evaluators may employ different sets of assumptions, such as the inclusion of model defects on cross sections [5] or model parameters [6]. Therefore evaluation work can benefit from the availability of a flexible modeling and inference framework that can be applied over the entire energy range.

Bayesian networks, e.g., [7], are a perspective on Bayesian inference that emphasizes the relationship between variables and can be regarded as such a flexible inference framework. Variables are depicted as nodes and arrows indicate the relationships between those nodes. An extensive paper [8] was recently published that elaborates on the mathematics of Bayesian networks tailored to nuclear data evaluation and also includes evaluation examples.

In this contribution, we provide practical details of how Bayesian networks can be created and inference performed within them with the *nucdataBaynet* R package [9]. This package is hosted on GitHub and publicly accessible under the permissive MIT license. The information in this contribution refers to the version of the package at the time of writing, which is January 2022. The corresponding commit id on GitHub is *e9df5fd*. As an important remark, the aim of this contribution is to give the reader some basic understanding of how the package can be used rather than a comprehensive description of its features.

First, we explain how information about different variables is stored and how these variables or sets of variables are associated with nodes. Second, we elaborate on the creation of links between different nodes. Third, we discuss how inference in Bayesian networks can be performed. Fourth, we give a schematic evaluation example putting everything together. Finally, a summary section is provided.

2 Creating nodes

An essential step in the construction of a Bayesian network is the definition of the nodes. To this end, a data table [10] is defined with all the variables appearing in the Bayesian network. An example of such a data table is given in Ta-

*e-mail: g.schnabel@iaea.org

IDX	NODE	PRIOR	OBS	REAC	ENERGY	EXPID
1	truexs	1000	NA	(N,TOT)	1.0	NA
2	truexs	1000	NA	(N,TOT)	1.5	NA
3	truexs	1000	NA	(N,TOT)	2.0	NA
4	normerr	0	NA	(N,TOT)	NA	20733
5	exp	0	200	(N,TOT)	1.7	20733

Table 1. Structure of a data table with the definition of the variables of the Bayesian network

ble 1. Many convenience functions exist in the R language to create data tables. Python users can think of these functions as equivalents to the many functions the Python *pandas* package [11] provides for creating data frames.

The columns that must always be present are *IDX*, *NODE*, *PRIOR* and *OBS*. The indices in the column *IDX* establish an order of the variables and are important for the Bayesian inference in the Bayesian network where vectors and covariance matrices need to be constructed. The column *NODE* determines the assignment of variables to nodes. As can be seen in the table, several variables can be associated with the same node. In this example, the variables assigned to the node *truexs* denote cross section values on a computational mesh. The column *OBS* indicates whether the value of a variable was observed or not. Unobserved variables have the special value NA (=Not Answered) whereas observed variables have the observed value in this column.

The meaning of the number in the column *PRIOR* depends on whether a variable has any ingoing links or not. The definition of links will be discussed in the next section, but we need to already introduce some aspects to clarify the meaning of the *PRIOR* column. To this end, let us consider the Bayesian network depicted in Figure 1. The nodes $\vec{x}_1$ and $\vec{x}_2$ do not contain any ingoing links. For instance, $\vec{x}_1$ could be *truexs* in Table 1. In contrast, the nodes $\vec{y}_1$ and $\vec{y}_2$ contain ingoing links. The *nucdataBaynet* package introduces the convention that every node with ingoing links, hence is a function of the values of some other nodes, must also be associated with a node $\vec{\varepsilon}_i$. Those extra nodes do not have any ingoing links and are only connected to exactly one node, such as $\vec{\varepsilon}_1$ to $\vec{y}_1$. Furthermore, the values in those nodes $\vec{\varepsilon}_i$ are an additive contribution for the values in $\vec{y}_i$. The reason for the introduction of those extra nodes by default and the stiff constraints on how they must be connected to other nodes is that it simplifies the implementation of Bayesian inference in the Bayesian network. These nodes are usually used to include statistical errors associated with experimental data.

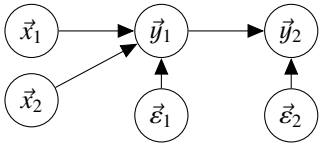

Figure 1. Simple Bayesian network

For nodes without any ingoing links—we may call them source nodes—such as $\vec{x}_1$ and $\vec{x}_2$, the number in *PRIOR* represents the prior best estimate. For all other nodes with ingoing links, such as $\vec{y}_1$ and $\vec{y}_2$, this number denotes the prior best estimate of the extra node, such as $\vec{\varepsilon}_1$ and $\vec{\varepsilon}_2$, respectively. In this construction, the values of the variables associated with nodes with ingoing links, e.g., $\vec{y}_1$ and $\vec{y}_2$ are deterministic functions of the values of all the nodes they depend on, hence also including the values of their $\vec{\varepsilon}_i$ node.

Regarding the node definition, only *IDX*, *NODE*, *PRIOR* and *OBS* are strictly required and all other columns dispensable. However, as will be discussed in the next section, the information in the extra columns may be essential for defining the links between nodes.

Finally, each node must also be endowed with a covariance matrix. For source nodes, such as $\vec{x}_1$ and $\vec{x}_2$, the associated covariance matrix reflects the prior knowledge on the variables in the node. For all other nodes, such as $\vec{y}_1$ and $\vec{y}_2$, the associated covariance matrix reflects the knowledge about values in the extra nodes $\vec{\varepsilon}_i$. In working with the *nucdataBaynet* package, these covariance matrix blocks are stored all together in a sparse matrix in compressed sparse row or column format, enabled by the *Matrix* package [12].

3 Establishing links

The second essential step is the definition of mappings between nodes. The definition of a mapping includes the specification of the source and the target variables, which is achieved by providing the associated indices given in the column *IDX* of the node data table.

Depending on the mapping type, additional information may be required. For instance, one type of mapping is linear interpolation. It can be used if the source variables are the cross sections on a computational mesh and the target variables measured cross sections associated with energies in-between the computational mesh points. Therefore, in order to establish a linear interpolation mapping, the energies associated with the cross sections of the computational mesh and those associated with the measured values are required.

In working with the *nucdataBaynet* package, the creation of a specific mapping is a two step procedure. First, a *named list* needs to be created. From the point of usage, they can be regarded as roughly equivalent to Python dictionaries, even though there are differences in terms of space and time complexity and field names in named lists are restricted to character strings. How this list creation can be done for a linear interpolation mapping is illustrated in Listing 1.

The *maptype* field indicates the type of mapping, here linear interpolation. The *mapname* field associates a name with this specific mapping, and can be helpful for debugging. The always required fields *src_idx* and *tar_idx* to denote the indices of the source and target variables, respectively, follow next. R data tables (as data frames in Python) provide powerful query functionality to select a

```
mesh_xs_to_exp <- list(
  maptype = "linearinterpol_map",
  mapname = "truexs_to_experiment",
  src_idx = dt[NODE=="truexs", IDX],
  tar_idx = dt[NODE=="exp", IDX],
  src_x = dt[NODE=="truexs", ENERGY],
  tar_x = dt[NODE=="exp", ENERGY],
)
```

Listing 1. Example R code for creating a list with the definition of a linear interpolation mapping

```
comp_map_def <- list(
  maptype = "compound_map",
  mapname = "my_cool_compound_map",
  maps = list(mesh_xs_to_exp, ...)
)
```

Listing 2. Example R code for creating a list with the definition of a compound mapping

subset of the table. Such queries are utilized in this example to retrieve the indices associated with the source node *truexs* and the target node *exp*, again analogous to how this could be achieved with *pandas* in Python. The last two fields, *src_x* and *tar_x*, are specific for a linear interpolation mapping and contain the x-values associated with the source and target variables. Query operations on the node data table are used again to extract the energies associated with the source and target variables. For other types of mappings, other fields may be present instead.

All required mappings for establishing the link structure of the Bayesian network can be constructed analogously to how it is sketched for a linear interpolation mapping above. In the example mapping definition above, the list was also assigned to the variable name *mesh_xs_to_exp* so that it can be used as an abbreviation in later definitions.

Once all required mappings are defined, they need to be bundled to a so-called compound mapping. An example of a compound mapping definition is given in Listing 2. This mapping only requires the specification of the *maptype* field with value *compound_map* and a field *maps*, which contains a list of all the mappings present in the Bayesian network.

Finally, the compound mapping definition is used to create a compound mapping object in the following way:

```
comp_map <- create_map(comp_map_def)
```

This mapping object provides the two functions *propagate* and *jacobian*. The former function allows to forward propagate values associated with nodes throughout the network. The latter function provides the Jacobian matrix, also known under the name sensitivity matrix.

These two functions can be used to perform sensitivity studies to assess the impact of changes in the Bayesian network. They are also leveraged for Baysian inference in the network as explained in the next section.

4 Bayesian inference

Bayesian networks rely on Bayesian inference. First a brief recapitulation of the Bayesian inference is presented and afterwards the connection to Bayesian networks established.

Bayesian inference is performed with the Bayesian update formula, e.g., [13],

$$\pi(\vec{z}\,|\,\vec{o}) \propto \ell(\vec{o}\,|\,\vec{z})\,\pi(\vec{z})\,, \tag{1}$$

which combines the knowledge expressed as prior distribution $\pi(\vec{z})$ for quantities of interest $\vec{z}$ with the likelihood $\pi(\vec{o}\,|\,\vec{z})$ that connects probabilistically the quantities of interest with the observed values in $\vec{o}$.

Bayesian modeling with the *nucdataBaynet* package makes the assumption that both the prior distribution and likelihood are multivariate normal. The likelihood is therefore given by $\vec{z} \sim \mathcal{N}(\mathcal{M}(\vec{z}), \mathbf{\Sigma}_{\text{exp}})$, where $\mathcal{M}(\vec{z})$ is a (possibly non-linear) model to propagate values in $\vec{z}$ to values that are comparable with those in $\vec{o}$ up to a difference explained by the uncertainties and correlations in the experimental covariance matrix $\mathbf{\Sigma}_{\text{exp}}$. The prior distribution is given by $\pi(\vec{z}) \sim \mathcal{N}(\vec{z}_0, \mathbf{\Sigma}_0)$ with $\vec{z}_0$ being the prior best estimate and $\mathbf{\Sigma}_0$ the corresponding prior covariance matrix.

The GLS method addresses a special case where the model $\mathcal{M}(\vec{z})$ is linear. The Levenberg-Marquardt algorithm [14–16], an iterative algorithm for non-linear optimization, is employed to determine values $\vec{z} = \vec{z}_{\text{MAP}}$ associated with the maximum of the posterior distribution $\pi(\vec{z}_{\text{MAP}}\,|\,\vec{o})$. This type of estimate is called a *Maximum A Posteriori* (MAP) estimate.

Following, we explain how the concept of nodes and links of a Bayesian network is connected to the Bayesian update formula in eq. (1). Let us consider again the Bayesian network illustrated in Figure 1. Because the links denote functional relationships between variables, we can express the structure of the network as a set of equations:

$$\vec{y}_1 = f_1(\vec{x}_1) + f_2(\vec{x}_2) + \vec{\varepsilon}_1 \tag{2}$$

$$\vec{y}_2 = f_2(\vec{y}_1) + \vec{\varepsilon}_2 \tag{3}$$

Each mapping in the compound mapping object corresponds to one function f_i. If the values of nodes without incoming links $\vec{x}_1, \vec{x}_2, \vec{\varepsilon}_1, \vec{\varepsilon}_2$ should be forward propagated through the network to obtain the values in $\vec{y}_1$ and $\vec{y}_2$, it is important to first apply the functions f_1 and f_2 and only afterwards f_3. If we are only interested in $\vec{y}_2$, e.g., because it corresponds to the observed quantity $\vec{o}$, we may directly write

$$\vec{y}_2 = f_3(f_1(\vec{x}_1) + f_2(\vec{x}_2) + \vec{\varepsilon}_1) + \vec{\varepsilon}_2\,. \tag{4}$$

This equation clarifies that the structure of a Bayesian network defines a sequence of nested functions. If we are not interested in the nested structure of the functions, we can equally write

$$\vec{y}_2 = \mathcal{M}(\vec{x}_1, \vec{x}_2, \vec{\varepsilon}_1) + \vec{\varepsilon}_2 = \mathcal{M}(\vec{z}) + \vec{\varepsilon}_2\,, \tag{5}$$

where the mathematical model $\mathcal{M}$ summarizes the effect of the individual functions and the vector $\vec{z}$ is constructed

by stacking together the vectors $\vec{x}_1$, $\vec{x}_2$ and $\vec{\varepsilon}_1$. The prior covariance matrix $\mathbf{\Sigma}_0$ in the Bayesian update equation is constructed by creating a block diagonal matrix from the covariance matrices associated with $\vec{x}_1$, $\vec{x}_2$ and $\vec{\varepsilon}_1$. The experimental covariance matrix $\mathbf{\Sigma}_{\text{exp}}$ is constructed by the extra nodes $\vec{\varepsilon}_i$ connected to the observed nodes, hence in the example by the covariance matrix associated with $\vec{\varepsilon}_2$.

The convenience from the user point of view lies in the fact that $\mathcal{M}(\vec{z})$ can be automatically constructed from the individual mappings. Similarly, the overall Jacobian matrix (sensitivity matrix) that encompasses all nodes at the same time can be obtained by applying the chain rule to combine the Jacobian matrices of the individual mappings. The logistics and bookkeeping involved are handled by the compound mapping object.

In the next section, an example is sketched that demonstrates the advantages of the Bayesian network framework in terms of computational and storage efficiency.

5 Benefits explained at an example

This section sketches a purely data-driven evaluation of the total cross section of ^{56}Fe between 1.0 and 1.1 MeV. A Gaussian process with a sparse structure will be assumed as mathematical fitting function. A good introduction to Gaussian processes is given in [17]. We consider one experimental data set of Cornelis et al. available in EXFOR [18, 19] under the accession number 22316001 and assume that the dataset is affected by statistical uncertainties and a normalization uncertainty. For the statistical uncertainties we adopt the values in the EXFOR entry and we assume a value of 0.05 barn for the normalization uncertainty. The code to reproduce the example analysis showcased in the following is also included in a tutorial, which is part of the *nucdataBaynet* package.

We may model this situation with the Bayesian network depicted in Figure 2. The node with the vector $\vec{\sigma}$ contains the cross section values on a computational mesh of incident energies. The mesh covers the range from 1.0 to 1.1 MeV with a spacing of 10 eV between mesh points. We can assign as prior best estimates a vector of zeros and a diagonal covariance matrix with very large diagonal elements (10^{30}) to implement a non-informative prior. If we are doing an evaluation in the fast energy region, we are expecting evaluated cross sections to have a certain degree of smoothness. Therefore, we introduce a new node

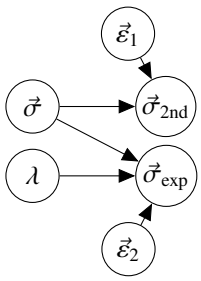

Figure 2. Bayesian network for the evaluation example. See the text for a description of the nodes and links.

$\vec{\sigma}_{\text{2nd}}$ containing the values of the second derivatives of the cross sections in $\vec{\sigma}$. The *nucdataBaynet* package already provides a mapping to go from a vector with values on an energy mesh to a vector with the second derivatives.

We assume that all the second derivates in $\vec{\sigma}_{\text{2nd}}$ were observed to be zero and that the associated covariance matrix (strictly speaking the prior covariance matrix associated with $\vec{\varepsilon}_2$) is a diagonal matrix with a single value α^2 repeated across the entire diagonal. We may call this observation a pseudo-observation because it does not correspond to any real measurement but is merely a mathematical trick to favor smooth solutions of the function encoded in a discretized way in the vector $\vec{\sigma}$. The value of α regulates the degree of smoothness. The construction of the prior knowledge for the cross sections in $\vec{\sigma}$ involving a pseudo-observation of the second derivatives in $\vec{\sigma}_{\text{2nd}}$ represents a Gaussian process that is in good approximation only opinionated on the degree of smoothness but not the range of the cross section values in $\vec{\sigma}$.

This construction is also attractive from a computational and storage efficiency point of view because all the covariance matrices associated with $\vec{\sigma}$ and $\vec{\sigma}_{\text{2nd}}$ are diagonal, which can be exploited in the Bayesian inference. For demonstration purposes, we will be doing the Bayesian inference two times, once with $\alpha = 10^4$ and another time with $\alpha = 10^7$ to show the impact on the evaluation result (the MAP estimate). As an aside, the *nucdataBaynet* package allows other Gaussian process constructions, e.g., based on pseudo-observations of first and third derivatives as well as based on user-provided covariance functions. At the time of writing, these options are however neither well documented in the package documentation nor demonstrated in the accompanying tutorials.

The measured values are summarized in the node $\vec{\sigma}_{\text{exp}}$. We assume that the prior best estimate of the values in $\vec{\varepsilon}_2$ is given by a zero vector and that the corresponding prior covariance matrix $\mathbf{D}$ is diagonal capturing the statistical uncertainties of the measured data points in $\vec{\sigma}_{\text{exp}}$. These statistical uncertainties are taken from the EXFOR entry and range from 0.04 to 0.06 barn. The normalization error λ is also associated with a node and linked to the measurement node $\vec{\sigma}_{\text{exp}}$ via a normalization mapping. The prior best estimate of λ is assumed to be zero and the corresponding prior variance η^2 (because a 1×1 covariance matrix is a single number—a variance) reflects the normalization uncertainty, which we assume to be 0.05 barn.

Noteworthy, the definition of the three nodes $\vec{\sigma}_{\text{exp}}$, λ and $\vec{\varepsilon}_2$ and the associated prior best estimates and covariance matrix blocks is fully equivalent to the specification that the experimental covariance matrix is given by $\mathbf{\Sigma}_{\text{exp}} = \mathbf{D} + \eta^2 \vec{1}\vec{1}^T$ where the diagonal matrix $\mathbf{D}$ reflects the statistical uncertainties and the second term the contribution of the normalization uncertainty. The notation $\vec{1}\vec{1}^T$ in the second term denotes a matrix of ones. This expression also shows that we are using an absolute normalization error—a shift—in this example rather than a relative one—a scaling factor—for the sake of simplicity. However, very often normalization errors are thought to be more adequately represented by scaling factors and this option is also supported by the *nucdataBaynet* package.

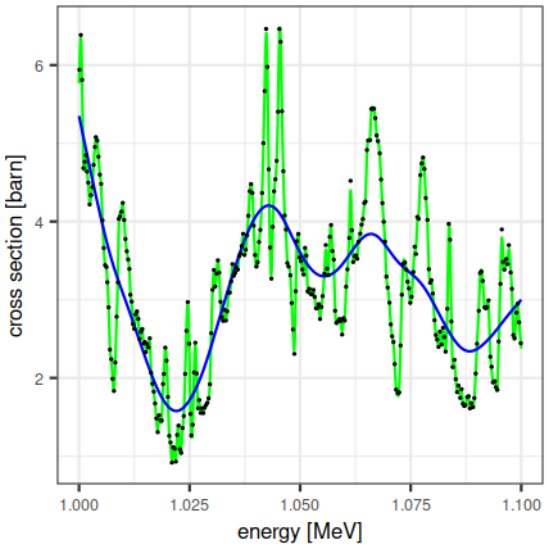

Figure 3. MAP estimate of the node $\vec{\sigma}$ resulting from two different prior uncertainty specifications of α. The blue line is the result for $\alpha = 10^4$ and the green line for $\alpha = 10^7$.

Regarding computational and storage efficiency, we can see here that the Bayesian network representation is very efficient. Instead of storing a dense matrix $\mathbf{\Sigma}_{\text{exp}}$ with all elements being non-zero, we can express identical uncertainty assumptions in the Bayesian network by relying on the nodes $\vec{\varepsilon}$ and λ with their diagonal prior covariance matrix blocks. This increased storage efficiency translates directly to an increased computational efficiency in the Bayesian inference because the inversion of a large and dense experimental covariance matrix can be avoided. The mathematical details regarding this statement can be found in [8].

Finally, we also need to connect the node $\vec{\sigma}$ to the measurement node $\vec{\sigma}_{\text{exp}}$, which we will do here by an interpolation mapping as described in section 3. As an aside, the *nucdataBaynet* package also offers other mappings types that can take into account a finite experimental energy resolution or uncertainty in incident energy calibration.

The MAP estimates based on all the modeling assumption described in this section are shown in Figure 3, where it can also be clearly seen that the value of α acts as a regularization parameter to control the smoothness of the solution. As all relationships are linear in in this network, the MAP estimate for the variables in this network coincides with the solution of the GLS method. Noteworthy, even though the computational mesh contains 10000 mesh points, the calculation of the MAP estimate takes less than one tenth of a second on the notebook of the author thanks to the sparse structure of the Bayesian network.

6 Summary

This contribution elaborated on the construction of a Bayesian network with the *nucdataBaynet* package [9], which is a two step procedure. The first step consists in the definition of nodes and the second step in the creation of mappings between the nodes.

It was also explained how Bayesian inference as implemented by the *nucdataBaynet* package is linked to the Bayesian update formula. Finally, a Bayesian network was created for a schematic evaluation of the total cross section of ^{56}Fe between 1.0 and 1.1 MeV.

This contribution provided a glimpse on the capabilities of Bayesian networks as a framework to create Bayesian models. The author believes that the mental abstractions of nodes and links are very beneficial in the creation of Bayesian models that truly reflect the assumptions of the evaluator. For instance, some quantities are known to be positive but this assumption may not be always incorporated in the evaluation process due to limitations of some evaluation codes or the GLS method.

The *nucdataBaynet* R package is available on the GitHub account of the Nuclear Data Section at the IAEA. As a final remark, the GMA code [3] used since the early 1980s for the evaluation of the neutron data standards [20] has been translated to a Python package *gmapy* [21] with new evaluation features. It is planned that also the gmapy package will support the creation of Bayesian models for nuclear data evaluation using the Bayesian network paradigm.

References

[1] T. Kariya, H. Kurata, *Generalized Least Squares*, 1st edn. (Wiley, 2007)
[2] D.W. Muir, A. Trkov, I. Kodeli et al., *The Global Assessment of Nuclear Data, GANDR* (EDP Sciences, 2007)
[3] W.P. Poenitz, S.E. Aumeier, Tech. Rep. ANL/NDM-139, Argonne National Laboratory, Argonne, Illinois (1997)
[4] N.M. Larson, Tech. Rep. ORNL/TM-9179/R4, Oak Ridge National Laboratory, Oak Ridge (1998)
[5] D. Neudecker, R. Capote, H. Leeb, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **723**, 163 (2013)
[6] P. Helgesson, H. Sjöstrand, Annals of Nuclear Energy **120**, 35 (2018)
[7] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, 1st edn. (Morgan Kaufmann, 2014)
[8] G. Schnabel, R. Capote, A. Koning et al., *Nuclear data evaluation with Bayesian networks* (2021), `arxiv:2110.10322`
[9] G. Schnabel, *IAEA-NDS/nucdataBaynet*, https://github.com/IAEA-NDS/nucdataBaynet (2021)
[10] M. Dowle, A. Srinivasan, *Data.table: Extension of 'data.frame'*, https://CRAN.R-project.org/package=data.table (2022)
[11] Wes McKinney, *Data Structures for Statistical Computing in Python*, in *Proceedings of the 9th Python in*

Science Conference, edited by S. van der Walt, Jarrod Millman (2010), pp. 56–61

[12] D. Bates, M. Maechler, *Matrix: Sparse and dense matrix classes and methods*, https://CRAN.R-project.org/package=Matrix (2021)

[13] D.S. Sivia, *Data Analysis: A Bayesian Tutorial* (Clarendon Press, 1996), ISBN 978-0-19-851889-1

[14] K. Levenberg, Quarterly of Applied Mathematics **2**, 164 (1944)

[15] D.W. Marquardt, Journal of the Society for Industrial and Applied Mathematics **11**, 431 (1963)

[16] P. Helgesson, H. Sjöstrand, Review of Scientific Instruments **88**, 115114 (2017)

[17] C.E. Rasmussen, C.K.I. Williams, *Gaussian Processes for Machine Learning* (MIT Press, Cambridge, Mass., 2006), ISBN 0-262-18253-X 978-0-262-18253-9

[18] V. Zerkin, B. Pritychenko, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **888**, 31 (2018)

[19] N. Otuka, E. Dupont, V. Semkova et al., Nuclear Data Sheets **120**, 272 (2014)

[20] A. Carlson, V. Pronyaev, R. Capote et al., Nuclear Data Sheets **148**, 143 (2018)

[21] G. Schnabel, *GitHub - IAEA-NDS/gmapy: Gmapy: A Python package for nuclear data evaluation*, https://github.com/IAEA-NDS/gmapy