Quick start guide
for BANSHEE

Dominik Paprotny, Oswaldo Morales-Nápoles, Daniël T. H. Worm, Elisa Ragno

20 November 2020

Contents

Preface

This short guide was written to enable users an easier start with the BANSHEE toolbox. It is not a step-by-step guide, but rather clarifies the main options for users to apply the toolbox in their MATLAB routines. We recommend to read it in combination with our paper "BANSHEE–A MATLAB Toolbox for Non-Parametric Bayesian Networks", published in *SoftwareX*, Vol. 12, 100588, https://doi.org/10.1016/j.softx.2020.100588.

Requirements

MATLAB with the Statistics and Machine Learning Toolbox installed. The toolbox was tested with more recent MATLAB versions (2018a, 2019b), but should work with some older versions as well.

Installation

In MATLAB, navigate the *Current Folder* window to the directory with BANSHEE.mtlbx BANSHEE and right-click on the file and select *Install*. In MATLAB version 2018b or newer, the following command line could also be used:

```
matlab.addons.toolbox.install('BANSHEE.mltbx')
```

On Windows systems, it is also possible to double-click on the BANSHEE.mtlbx BANSHEE outside MATLAB to install the toolbox.

Note that any previously installed versions of BANSHEE should be uninstalled in advance. This can be easily done (in MATLAB 2018a–2019b) by clicking on the *Add-Ons -> Manage Add-Ons* in the *Home* tab of MATLAB, right-clicking on BANSHEE in the *Add-On Manager* window, and choosing *Uninstall*. More options and information are accessed through the *View Details* window (see Figure 1). The window can be used to access all the functions by clicking on the *Open Folder* button.

BANSHEE version 1.0 by Oswaldo Morales Napoles, Dominik Paprotny
installed on 18 May 2020
A MATLAB Toolbox for Non-Parametric Bayesian Networks
Toolbox

Open Folder | Learn More ▼ | Uninstall...

Description

Bayesian Networks (BNs) are probabilistic, graphical models for representing complex dependency structures. They have many applications in science and engineering. This toolbox implements a particularly powerful variant Non-Parametric BNs. The software allows for quantifying the BN, validating the underlying assumptions of the model, visualizing the network and its corresponding rank correlation matrix, and finally making inference with a BN based on existing or new evidence. The toolbox also includes some applied BN models published in recent scientific literature.

▶ View File List

Figure 1: Details on BANSHEE in the Add-On Manager window (MATLAB 2019b).

Overview of the toolbox

BANSHEE has three components:

- Five function scripts for implementing and analyzing non-parametric Bayesian Networks (NPBN);

- Three scripts containing example applications of the NPBN functions, intended for learning how to use the toolbox and NPBN method in general;
- Five function scripts containing quantified NPBN models from literature, that could be applied to solve a specific problem, according to each model's set-up.

Each script has an extensive description in its header, which is also accessible through MATLAB's `help` function, e.g.:

```
help inference        % access help for function inference.
```

The ensuing help text presents all arguments and options of each BANSHEE function and provides background information.

Using the toolbox

When using the toolbox for the first time, it is recommended to run `example`, a script that highlights the use of all five function scripts. This section contains information on how to use each function illustrated by each step of the `example` script.

Creating DAGs of new NPBN models

NPBN models are largely expert knowledge-driven models. They are intended to recreate real-life interactions in a probabilistic framework. The structure could still be learned from the data using an iterative approach. In the first step, a simple rank correlation matrix can help to identify the first arc between a variable of interest and an explanatory variable (according to the user's designation of those). It can be generated with a standard MATLAB function `corr` for a data matrix `X`:

```
RHO = corr(X,'type','spearman','rows','complete');
```

Further arcs are added by the user (using expert knowledge or the unconditional correlations) and then running the `bn_rankcorr` function from BANSHEE. Plotting the BN rank correlation matrix and visualizing the directed acyclic graph (DAG) with the (conditional) correlations with `bn_visualize` allows identifying arcs for which the correlation is very low. These arcs could be removed and the correlation matrix recalculated to obtain new (conditional) correlations in the BN. The iterative process can be combined with the diagnostic tools of the toolbox (primarily `gaussian_distance`) and validation of the BN's predictions obtained through `inference` function, as shown in the `example` script. The most important feature of the DAG that has to be maintained is the lack of circular connections between variables (it has to be acyclic).

Example

The `example` script employs a default MATLAB dataset `cities`, which contains data on nine quality-of-life indicators in 329 cities in the United States. In the first step, data are loaded: the matrix `ratings` contains the numerical data, with nine variables in columns and a row of data per city, while cell array `categories` contains variable names, which is useful to have for visualizing the NPBN and other analyses (default names are assigned if names are not specified). The purpose of the model is to use different indicators of quality of life and predict the level of personal safety in American cities.

The data are prepared as follows:

```
load cities                    % a default Matlab dataset
Data = ratings;                % a matrix with data in columns
for i=1:9                      % variable names as a cell array
    Names{i}=strcat(num2str(i),'.',categories(i,:));
end
Names{4}='4.safety';           % variable 'crime' is renamed to 'safety'
```

After the data are ready, the structure of the NPBN is defined (the DAG). The DAG is constructed through expert knowledge supported by BN rank correlation matrix and the diagnostic tools. In the code, it is defined as a cell array ParentCell. Each cell of the array refers to one node (variable) of the DAG and lists all the parent nodes (empty if there are no parents). In our example, the DAG (Fig. 3) is as follows:

```
ParentCell{1} = [];         % climate (no parents)
ParentCell{2} = 3;          % arts (parent node: recreation)
ParentCell{3} = [4 1];      % recreation (parent nodes: economics, climate)
ParentCell{4} = [];         % economics (no parents)
ParentCell{5} = [2 3 4 1];  % safety (parents: all other variables)
```

BN Rank Correlation Matrix (bn_rankcorr)

Purpose

R = bn_rankcorr(PARENTCELL,DATA,ISDATA,PLOT,NAMES) computes the BN rank correlation matrix, which quantifies the strength of the (conditional) dependency between variables.

Input

PARENTCELL   A cell array containing the structure of the BN (directed acyclic graph – DAG). Each cell is a node of the BN and contains a list of the node's parents, defined as a vector with reference to particular cell(s). An example of the DAG is shown in the previous section, "Creating DAGs of new NPBN models".

DATA   A matrix containing data for quantifying the NPBN. Data for each node need to be located in columns in the same order as specified in **PARENTCELL**. The number of columns need to be at least equal the number of nodes specified in **PARENTCELL**. Surplus columns will be disregarded. Optionally, a cell array of rank correlations can be used, one conditional correlation per arc, following the same structure as **PARENTCELL**. This second option is intended for use in a User-Defined Random Model (UDRM). An example of a UDRM is shown in example_udrm script.

ISDATA   Specifies the input data type:

0 - cell array **DATA** contains rank correlations (for UDRM models);

1 - matrix **DATA** contains actual data.

PLOT (optional parameter). A plot of correlation matrix **R** can be displayed:

> 0 - do not create a plot (default);

> 1 - create a plot.

NAMES (optional parameter). A cell array containing names of the nodes for the plot; otherwise, default names are assigned.

Output

R An *n*-by-*n* matrix with the Spearman's (conditional) rank correlations, where *n* is the number of nodes, as specified in **PARENTCELL**.

Example

The DAG defined in `ParentCell` variable, the data matrix `Data` and (optionally) a cell array of variable names `Names` are inputs for the `bn_rankcorr` function. The inputs are composed as follows:

```
R = bn_rankcorr(ParentCell,... % structure of the BN
                Data,...        % matrix of data
                1,...           % matrix Data contains actual data
                1,...           % create a plot (0 = don't create plot)
                Names);         % names of variables
```

The value of `IsData` is 1, as we provide a matrix of actual data to quantify the model. Further, the value of `Plot` is also 1, as we want the function to generate a plot.

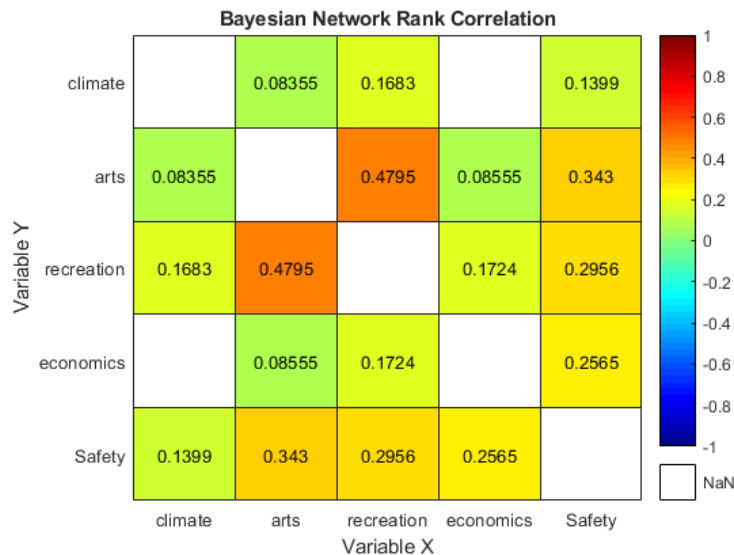Running the function returns a BN correlation matrix R and a plot (Fig. 2).

Figure 2: BN rank correlation matrix generated with bn_rankcorr after running example.m script.

Visualize the BN (bn_visualize)

Purpose

bn_visualize(DATA,R,NAMES) creates a directed digraph presenting the structure of nodes and arcs of the Bayesian Network (BN), also displaying the (conditional) rank correlations at each arc.

Input

PARENTCELL A cell array containing the structure of the BN (directed acyclic graph – DAG). Each cell is a node of the BN and contains a list of the node's parents, defined as a vector with reference to particular cell(s). An example of the DAG is shown in the previous section, "Creating DAGs of new NPBN models".

R   An *n*-by-*n* matrix with the Spearman's (conditional) rank correlations, where *n* is the number of nodes, as specified in **PARENTCELL**. This variable should be generated with **bn_rankcorr** function (see section "BN Rank Correlation Matrix").

NAMES (optional parameter). A cell array containing names of the nodes for the plot; otherwise, default names are assigned.

Output

The function doesn't provide an output variable. Instead, it generates a plot presenting the DAG: nodes, variables and (conditional) rank correlations.

Example

The DAG from the cities example can be visualized using `bn_visualize` function. The only required arguments are `ParentCell` and `R` from the previous steps:

```
bn_visualize(ParentCell,...    % structure of the BN
             R,...             % the rank correlation matrix (function 1)
             Names)            % names of variables
```

Running the function returns a plot containing nodes (red dots with names of variables), the arcs with a defined direction, and the value of the (conditional) rank correlations on the arcs (Fig. 3).
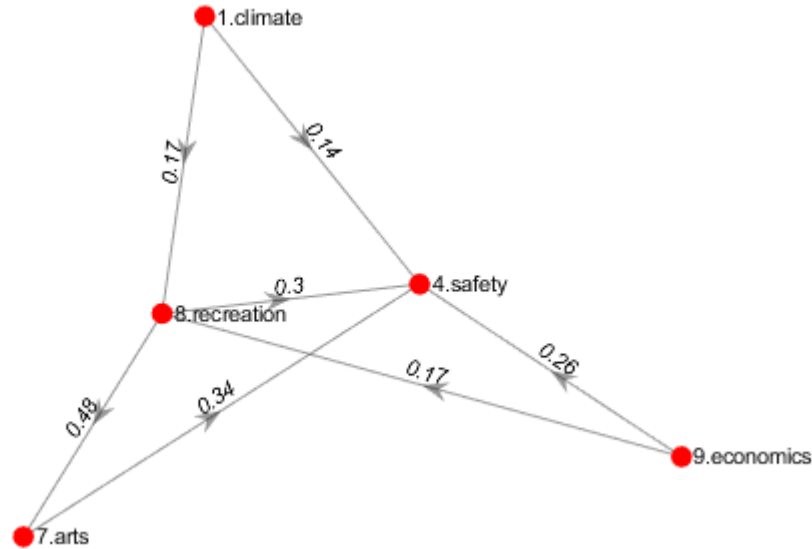
Figure 3: BN rank correlation matrix generated with bn_visualize after running example.m script.

Diagnostic test 1: Goodness-of-fit test for copulas (cvm_statistic)

Purpose

`M = cvm_statistic(DATA,PLOT,NAMES)` calculates the goodness-of-fit for pairs of variables, using Cramer-von Mises statistic.

Input

`DATA`   A matrix containing data for quantifying the NPBN. Each column is one variable.

`PLOT`   (optional parameter). A plot of highlighting the optimal copula per pair of variables can be displayed:

>   0 - do not create a plot (default);

>   1 - create a plot.

`NAMES`   (optional parameter). A cell array containing names of the nodes for the plot; otherwise, default names are assigned.

Output

`M`   A matrix containing the following columns:

>   [1]: first variable in the pair (column number in the matrix `DATA`);

>   [2]: second variable in the pair (column number in the matrix `DATA`);

>   [3]: Spearman's rank correlation between variables;

>   [4]: Cramer-von Mises statistic for Gaussian copula;

>   [5]: Cramer-von Mises statistic for Gumbel copula;

[6]: Cramer-von Mises statistic for Clayton copula;

[7]: Cramer-von Mises statistic for Frank copula.

The Cramer-von Mises statistic measures the sum of squared difference between the parametric and empirical copulas.

Example

The `cvm_statistic` can be used to analyse some of the underlying assumptions of the method. Firstly, the assumption that the bivariate dependencies between variables can be modelled with a Gaussian (normal) copula. The function `cvm_statistic` computes the sum of squared differences between the parametric and empirical copulas. The lower value of the resulting `M` metric, the better fit between the parametric and empirical copulas is achieved. The only required argument is the matrix of data used in the previous steps:

```
M = cvm_statistic(Data,...      % matrix of data
                  1,...         % create a plot (0 = don't create plot)
                  Names);       % names of variables
```

Optionally, two arguments can be added: to generate a plot (not created by default) and add variable names (as in the other functions). Running the script will result in the following graph (Fig. 4):
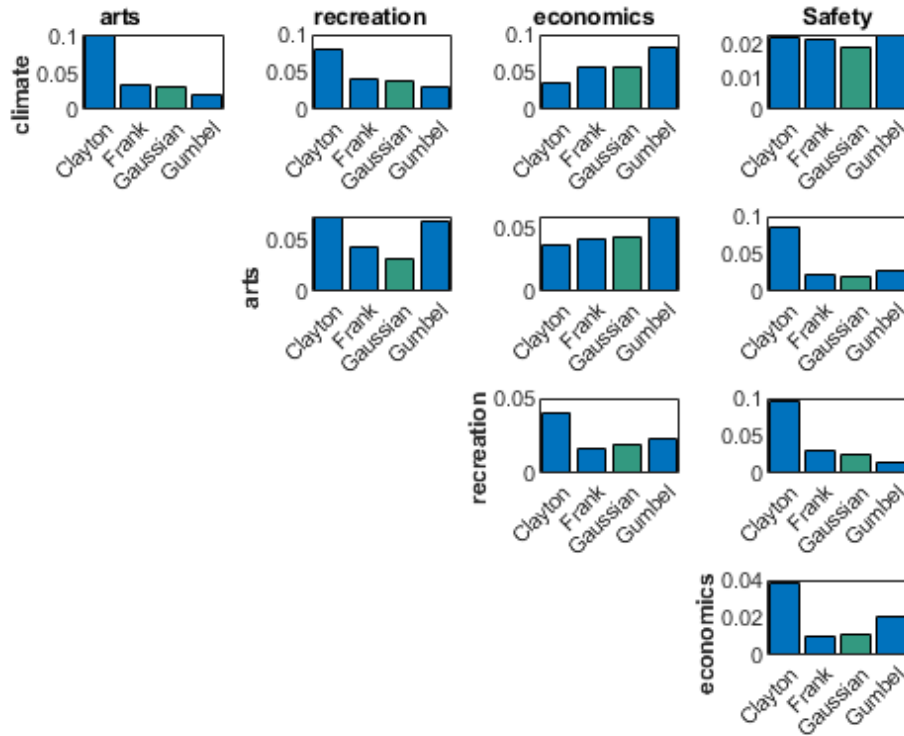

Figure 4: Diagnostic results for four copula types obtained using cvm_statistics after running example.m script.

The results of the diagnostic routines M highlight that the Gaussian copula is in the majority of cases the most suitable for representing the dependency between variables, especially for the variable of interest (safety).

Diagnostic test 2: Distance between Gaussian densities (gaussian_distance)

Purpose

`[D_ERC,B_ERC,D_BNRC,B_BNRC]=gaussian_distance(R,DATA,SAMPLESIZE,PLOT,TYPE)` computes the d-calibration score, which compares the distance between both the empirical and BN rank correlation matrices and the empirical normal rank correlation matrix.

Input

`R`   An *n*-by-*n* matrix with the Spearman's (conditional) rank correlations, where *n* is the number of nodes. This variable should be generated with `bn_rankcorr` function (see section "BN Rank Correlation Matrix").

`DATA`   A matrix containing data for quantifying the NPBN. Data for each node need to be located in columns in the same order as specified in `R`. The number of columns need to be equal to the number of nodes specified in `R`.

`SAMPLESIZE`   (optional parameter). A vector with the number of samples to be drawn in the resampling of the distributions in the test and, if two numbers is specified, number of iterations of computing the confidence interval of the determinant of the sampled random distribution. 1000 is the default.

`PLOT`   (optional parameter). A plot of the d-calibration scores can be displayed:

  0 - do not create a plot (default);

  1 - create a plot.

`TYPE`   (optional parameter). A string that sets the type of measure used to calculate the distance. Available methods are:

  `'H'`   Hellinger distance (default);

  `'KL'`   Symmetric Kullback–Leibler divergence;

  `'B'`   Bhattacharyya distance;

  `'G'`   G distance from Abou Moustafa et al. (2010)[1].

Output

`D_ERC`   Value of the d-calibration score for the empirical rank correlation matrix of `DATA`

---

[1] Abou Moustafa, K.T., De La Torre, F., and Ferrie, F. P. (2010). Designing a Metric for the Difference between Gaussian Densities. In: Angeles et al., "Brain, Body and Machine. Advances in Intelligent and Soft Computing." Berlin: Springer, 57-70.

D_BNRC Value of the d-calibration score for the Bayesian Network rank correlation matrix **R**.

B_ERC Quantile range (5th and 95th percentile) of the distribution of the determinant of the empirical distribution of **DATA** transformed to standard normal.

B_BNRC Quantile range (5th and 95th percentile) of the distribution of the determinant of the empirical distribution of the Bayesian Network.

The score is 1 if the matrices are equal and 0 if one matrix contains a pair of variables perfectly correlated, and the other one does not, and the score will be "small" as the matrices differ from each other elementwise.

Example

The second diagnostic test is done with the function `gaussian_distance`, which requires two arguments from the cities example, namely the rank correlation matrix `R` and the dataset `Data`.

```
[D_ERC,B_ERC,D_BNRC,B_BNRC] = gaussian_distance(R,...  % rank correlation matrix
                              Data,...      % matrix of data
                              SampleSize,...  % number of samples drawn
                              1,...         % create a plot
                              'H');         % type of distance metric
```

Three additional arguments that can be specified. In this example,

```
SampleSize = [500;1000];
```

indicates that the function will draw 500 samples of the normal distribution, and then perform 1000 iterations to obtain the distribution of the d-calibration score. This option was added as the test is sensitive to the number of samples drawn as well as the number of iterations and is rather severe for large datasets. A plot can be generated (Fig. 5) and finally the distance metric can be specified out of four metrics implemented in the code. By default, Hellinger distance is used (`'H'`).
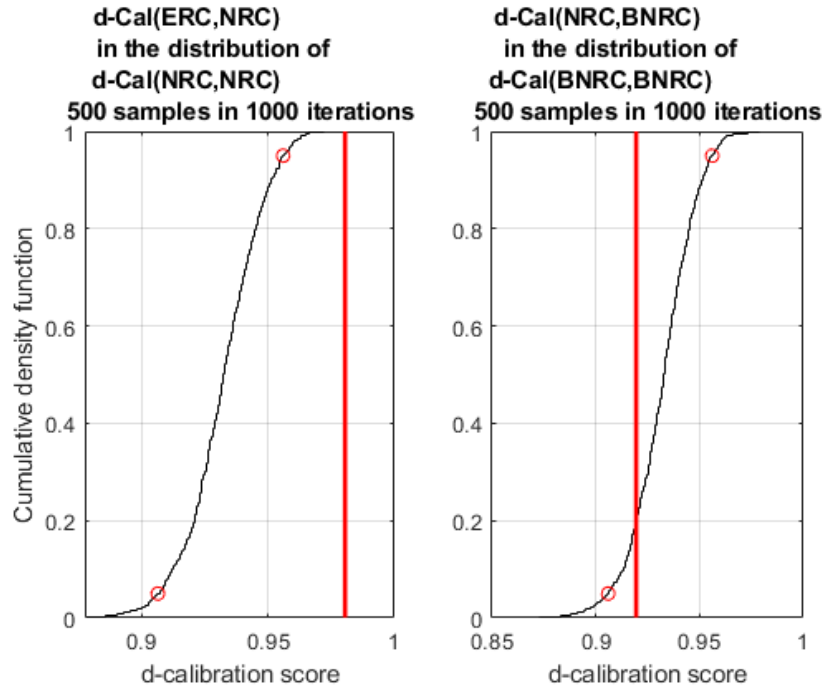
Figure 5: The d-calibration score of the BN model obtained using gaussian_distance after running example.m script.

The plot shows that the d-calibration score of the empirical rank correlation matrix (D_ERC) is outside the 90% confidence interval of the determinant of the empirical normal distribution (B_ERC), which is an unsatisfactory result (Fig. 5 left). However, the d-calibration score of the BN's rank correlation matrix (D_BNRC) is well within the 90% confidence interval of the determinant of the random normal distribution sampled for the same correlation matrix (B_BNRC; Fig. 5 right). This means that while the joint normal copula is not a good assumption for the whole dataset, it is valid for the particular configuration of the BN model.

Inference of the BN (inference)

Purpose

`F = inference(NODES,VALUES,R,DATA,OUTPUT,SAMPLESIZE,INTERP)` makes inference of the BN model, i.e. conditionalizes nodes of the model in order to obtain conditional distributions, which can be used as predictions for the values of the other nodes without observed values.

Input

NODES   A vector defining nodes to be conditionalized. The values of the vector define the variables in the same order as in matrix R. At least one node has to be left out from **NODES** in order to make inferences at least for this one node.

VALUES   A matrix containing data on which the inference will be based upon. Data for each node need to be located in columns in the same order as specified in **NODES** and **R**. The number of columns need to at least equal the number of nodes specified in **NODES** and **R**. Surplus columns will be disregarded.

R An *n*-by-*n* matrix with the Spearman's (conditional) rank correlations, where *n* is the number of nodes, as specified in **PARENTCELL**. This variable should be generated with **bn_rankcorr** function (see section "BN Rank Correlation Matrix").

DATA A matrix containing data for quantifying the NPBN. Data for each node need to be located in columns in the same order as specified in **NODES**, **VALUES** and **R**. The number of columns need to be equal the number of nodes specified in **NODES**, **VALUES** and **R**.

OUTPUT (optional parameter). A string setting the type of output of the function:

> `'full'` provides a cell array with the conditional empirical distributions (default).

> `'mean'` provides a matrix with the mean of the conditional empirical distributions.

> `'median'` provides a matrix with the median of the conditional empirical distributions.

SAMPLESIZE Number of samples drawn when conditionalizing the NPBN. 1000 is the default.

INTERP A string with the name of the interpolation method. The options are the same as in MATLAB's `interp1` function: `'linear'`, `'nearest'`, `'next'` (default), `'previous'`, `'spline'`, `'pchip'`, `'cubic'`, `'v5cubic'` or `'makima'`.

Output

F By default, provides a cell array with the conditional empirical distributions for each row in **VALUES** and each node not specified in **NODES**.

Example

Once the BN model was configured and analysed, it can be used to make inference. The function `inference` requires four arguments, of which two are used in the cities example: the rank correlation matrix `R` and the dataset `Data`. The argument `Nodes` defines which nodes are to be conditionalized, according to the numbering defined in the DAG by the `ParentCell` variable. Per each node conditionalized, a numeric value has to be provided in the variable `Values` in the same order as specified in `Nodes`. For example, if `R` is a 5-by-5 matrix,

```
Nodes = [1 3 5];
```

will conditionalize the BN using the first, third and fifth node and make inference of the second and fourth node.

Multiple rows of data (i.e. different observation records) in `Values` are possible. The configuration of those arguments in the cities example is:

```
Nodes  = 1:4;          % all variables except for safety (node 5)
Values = Data(:,1:4);  % data for conditionalization
```

The full `inference` function is written as follows:

```
F = inference(Nodes,...      % nodes that will be conditionalized
              Values,...     % information used to conditionalize the
                             % nodes of the BN
```

```
                  R,...          % the rank correlation matrix
                  Data,...       % matrix of data
                  'median',...   % type of output data
                  10000,...      % number of samples drawn
                  'nearest');    % interpolation method for the output ecdf
```

All optional parameters are used in this example. In this setting, the output variable `F` will contain only the median value of the predictions of the fifth node (safety), computed based on 10,000 samples of the BN, with the empirical marginal distributions interpolated using the nearest-neighbour method.

Computations made with inference can be e.g. used to compare prediction of the model with observations. It should be noted that the function, for larger dataset, will display the progress of the calculation and the display `Calculation complete` once all data in `Values` have been processed.

Using real-world example models

The toolbox contains five BN models presented previously in literature (`predict_floor_space`, `predict_coastal_erosion`, `predict_river_discharge`, `predict_residential_flood_damage`, `predict_commercial_flood_damage`). All three are used in the same way, similar to the `inference` function. The `predict_river_discharge` is highlighted in a wrapper script `example_hydro_simulation`. It computes and visualizes river discharges for an example dataset of Kingston river gauge in London, United Kingdom. All three models have example datasets for inference provided.

Model for estimating residential building floor space (predict_floor_space)

The model estimates useful floor space area of a given residential building [m²]. The model is implemented through `predict_floor_space` function:

```
FSA = predict_floor_space(Values,... % data to conditionalize the BN.
                          Nodes,...  % nodes to be conditionalized.
                          Output)    % type of output data.
```

The model estimates useful floor space area of a given residential building [m²]. The argument `Values` has a strictly defined order of variables and their units. The requirements are described in the header of the script, accessible also through MATLAB's `help` command. Here, the data used to run the function are:

[1] Population density per km², originally from Eurostat's GEOSTAT 2011 dataset at 1 km resolution.

[2] Building footprint area [m²], originally from OpenStreetMap (REQUIRED).

[3] Imperviousness (soil sealing) [%], originally from Imperviousness 2012 dataset from Copernicus Land Monitoring Service at 100 m resolution.

Of course, the important feature of BN models is its ability to make inference with only some of the data available. Therefore, at least one of the variables needs to be provided and identified through the `Nodes` variable. In the particular case of `predict_floor_space`, node 2 (building footprint area) needs to be always included for a different reason, namely that the BN model

itself estimates the height of buildings, which is then transformed into floor space based on the height, footprint area and two empirical coefficients. Optionally, the user can specify `Output` just as in the inference function. For example, if we want to know the mean estimate of useful floor space area for three buildings for which we have collected all three variables, we can specify the inputs as follows:

```
Values = [287 117 37; 1678 75 62; 545 53 88]; % data for predictions
Nodes  = [1 2 3];        % use all variables (default)
Output = 'mean';         % F returns only the mean of the uncertainty
                         % distribution of FSA
```

Running the function with those inputs will return a 3x1 numeric matrix FSA with the mean estimate of the posterior uncertainty distribution of floor space area [m²] of three buildings. If `Output = 'full'` is specified, the full uncertainty distribution will be returned as a cell array containing results per each sample drawn.

The example dataset, contained in the variable `szczecin`, include data on 70 random residential buildings located in the city of Szczecin, Poland, which was part of the validation dataset used in the original study that created the model. The columns in the dataset contain the three predictors described above and also an estimate of the building floor space derived from the number of floors and the footprint area obtained from cadastral data. The example data can be used as follows:

```
FSA = predict_floor_space(szczecin(:,1:3,1:3,'full')
```

Model for estimating extreme river discharges (predict_river_discharges)

The model estimates annual maximum of river discharge at a given location and year [m³/s]. The model is implemented through `predict_river_discharges` function:

```
QMAX = predict_river_discharges(Values,... % data to conditionalize the BN.
                                Nodes,...  % nodes to be conditionalized.
                                Output)    % type of output data
```

The argument `Values` has a strictly defined order of variables and their units. The requirements are described in the header of the script, accessible also through MATLAB's `help` command. Here, the data used to run the function are:

[1] Catchment steepness [m/km], from a digital elevation model (originally from EU-DEM).

[2] Catchment area [km²], from a hydrological map (originally from CCM2 database).

[3] Annual maximum of rainfall and snowmelt [mm], from meteorological data (originally from a regional climate model in the EURO-CORDEX framework).

[4] Runoff coefficient [-], which is total runoff in the catchment (from climate models) divided by annual maximum of rainfall and snowmelt, from climate or hydrological models (originally from a regional climate model in the EURO-CORDEX framework).

[5] Fraction of catchment covered by lakes [-], from land-use data (originally from CORINE Land Cover 2000).

[6] Fraction of catchment covered by marshes [-], from land-use data (originally from CORINE Land Cover 2000).

[7] Fraction of catchment covered by artificial surfaces [-], from land-use data (originally from CORINE Land Cover 2000).

Of course, the important feature of BN models is its ability to make inference with only some of the data available. Therefore, at least one of the variables needs to be provided and identified through the `Nodes` variable. Optionally, the user can specify `Output` just as in the inference function. For example, we want to know the median estimate of the annual maximum of extreme river discharge in a specific location and we have collected five variables. We don't have information on catchment steepness and the fraction covered by artificial surfaces. Then, we can specify the inputs variables in the right ordering and indicate that only nodes for which we have provided data are to be used for inference:

```
Values = [8900 54 0.67 0.03 0.07]; % data for predictions
Nodes  = [2:6];        % use only nodes 2-6
Output = 'median';     % F returns only the median of the uncertainty
                       % distribution of QMAX
```

Running the function with those inputs will return a 1x1 numeric matrix QMAX with the median estimate of the posterior uncertainty distribution of extreme river discharge [m³/s]. If `Output = 'full'` is specified, the full uncertainty distribution will be returned as a cell array containing results per each sample drawn.

The example dataset, contained in the variable `kingston`, includes 56 annual observations of discharge at Kingston river gauge in London, United Kingdom, together with data on all seven variables that can be used in the model to predict discharges. The example data can be used as follows:

```
QMAX = predict_river_discharges(kingston(:,1:7),1:7,'full')
```

Model for estimating storm-induced coastal cliff erosion (predict_coastal_erosion)

The model is more complex than the other two, as it contains 22 nodes, with 5 variables of interest. It estimates 5 different metrics of erosion of the cliff and the beach below it, as follows:

SHORE   shoreline (1 m above mean sea level) retreat [m].

BEACH   beach volume balance [m³].

FOOT   cliff foot retreat [m].

CLIFF   cliff volume balance [m³].

TOP   cliff top retreat [m].

Positive values of the output variables indicate erosion (retreat or loss of volume), while negative values indicate accretion (expansion or gain of volume).

The model is implemented through `predict_coastal_erosion` function:

```
[SHORE,BEACH,FOOT,CLIFF,TOP] = predict_coastal_erosion
                              (Values,... % data to conditionalize the BN.
                               Nodes,...  % nodes to be conditionalized.
                               Output)    % type of output data
```

The argument `Values` has a strictly defined order of variables and their units. The requirements are described in the header of the script, accessible also through MATLAB's `help` command.

15 different meteorological and hydrological variables can be used in the model. In the original study, storm surge heights were taken from tide gauge observations, wave and wind parameters from WAM wave model hindcast and other meteorological data from ERA5 climate reanalysis.

[1] Significant wave height, maximum [m] in the preceding period.

[2] Temperature, mean [°C] in the preceding period.

[4] Wind speed, 95th percentile [m/s].

[5] Water level above mean, 95th percentile [cm].

[6] Wave direction relative to the coast, average during storms (above +45 cm mean sea level) [degree].

[7] Peak wave period, average [s].

[9] Water level, average during storms [cm].

[10] Significant wave height, 95th percentile [m].

[12] Wave power, 95th percentile [kW/m].

[13] Mean wave period, average [s] in the preceding period.

[16] Accumulated excess storm energy, total [J/m²]

[17] Mean wave period, maximum [s]

[18] Water level above mean, maximum [cm]

[20] Precipitation, total [mm]

[21] Significant wave height, maximum [m]

The variables mostly pertain to the period between reference time points for which the erosion is computed. An exception are three variables (1,2 and 13) which pertain to the conditions in a preceding period (ending with the start of the reference period). In the original study, those 3 variables were used to predict the starting beach width and cliff slope, which then influenced shoreline and cliff foot erosion. Those variables can also be specified directly:

[3] Starting beach width (between 1 m above mean sea level and cliff foot) [m].

[14] Cliff slope (difference of height divided by length) [-].

In the original study, those variables were obtain from a digital elevation model constructed with LIDAR scanning.

Of course, the important feature of BN models is its ability to make inference with only some of the data available. Therefore, at least one of the variables needs to be provided and identified through the `Nodes` variable. Optionally, the user can specify `Output` just as in the inference function. For example, we want to know the estimate of cliff erosion in a specific location between two reference periods. We don't have information on wave conditions, but only sea level and meteorological conditions (6 nodes). Then, we can specify the inputs variables in the right ordering and indicate that only nodes for which we have provided data are to be used for inference:

```
Values = [5.5 11 84 56 131 38; 4.2 8 61 33 75.5 59]; % data for predictions
Nodes  = [2 4 5 9 18 20]; % use only nodes with sea level and
                          % meteorological parameters
Output = 'full';          % returns a cell array containing results per
                          % each sample drawn.
```

Running the function with those inputs will return five 1x2 numeric matrices with the full posterior uncertainty distributions of erosion for the two time periods specified.

The example dataset, contained in the variable `bansim`, include 12 observations of erosion at a profile of a cliff in the town of Bansim, in the German Baltic Sea coast, made between 2016 and 2018, together with data on all variables that can be used in the model to predict erosion. The example data can be used as follows:

```
[SHORE,BEACH,FOOT,CLIFF,TOP] = predict_coastal_erosion(bansim(:,[1 2 4:7 9
10 12 13 16:18 20 21]),[1 2 4:7 9 10 12 13 16:18 20 21],'mean')
```

Model for estimating residential flood damages (predict_residential_flood_damage)

The model estimates relative losses (damages) sustained by residential assets (building structure and household contents). The two output variables are:

BRLOSS   Relative loss to building structure (fraction).
CRLOSS   Relative loss to household contents (fraction).

The model is implemented through `predict_residential_flood_damage` function:

```
[BRLOSS,CRLOSS] = predict_residential_flood_damage
                (Values,...    % data to conditionalize the BN.
                 Nodes,...     % nodes to be conditionalized.
                 Output)       % type of output data
```

The argument `Values` has a strictly defined order of variables and their units. The requirements are described in the header of the script, accessible also through MATLAB's `help` command. Here, the data used to run the function are:

[1] Return period of the flood event [years].

[2] Water depth [cm].

[3] Net disposable income per person in the region [euro, 2015 price level].

[4] Flow velocity [m/s].

[5] Useful floor space area of the building [m²].

Of course, the important feature of BN models is its ability to make inference with only some of the data available. Therefore, at least one of the variables needs to be provided and identified through the `Nodes` variable. Optionally, the user can specify `Output` just as in the inference function. For example, we want to know the median estimate of building losses in a specific location and we have collected three variables. We don't have information on regional disposable income and flow velocity. Then, we can specify the inputs variables in the right ordering and indicate that only nodes for which we have provided data are to be used for inference:

```
Values = [70 105 278]; % data for predictions
Nodes  = [1:2 5];       % use only nodes 1, 2 and 6
Output = 'median';      % F returns only the median of the uncertainty
                        % distribution of flood losses
```

Running the function with those inputs will return two 1x1 numeric matrices BRLOSS and CRLOSS with the median estimate of the posterior uncertainty distribution of relative losses. If `Output = 'full'` is specified, the full uncertainty distribution will be returned as a cell array containing results per each sample drawn.

The example dataset, contained in the variable `Meuse_flood`, includes a selection of 100 Dutch households affected by the 1993 Meuse river flood together with data on all five variables that can be used in the model to predict flood losses. The example data can be used by simply referring to the dataset:

```
BRLOSS,CRLOSS] = predict_residential_flood_damage(Meuse_flood)
```

Model for estimating commercial flood damages (predict_residential_flood_damage)

The model estimates relative losses (damages) sustained by commercial assets (building structure and machinery/equipment). The two output variables are:

BRLOSS   Relative loss to building structure (fraction).
ERLOSS   Relative loss to machinery and equipment (fraction).

The model is implemented through `predict_commercial_flood_damage` function:

```
[BRLOSS,CRLOSS] = predict_commercial_flood_damage
                (Values,...    % data to conditionalize the BN.
                Nodes,...      % nodes to be conditionalized.
                Output)        % type of output data
```

The argument `Values` has a strictly defined order of variables and their units. The requirements are described in the header of the script, accessible also through MATLAB's `help` command. Here, the data used to run the function are:

[1] Water depth [cm].

[2] Inundation duration [hours]

[3] Gross fixed capital formation per person employed in the region [euro, 2015 price level],

[4] Gross value added per capita in the region [euro, 2015 price level],

[5] Precautionary measures ratio (number of measures used divided by number of applicable measures) [fraction].

Of course, the important feature of BN models is its ability to make inference with only some of the data available. Therefore, at least one of the variables needs to be provided and identified through the `Nodes` variable. Optionally, the user can specify `Output` just as in the inference function. For example, we want to know the estimate of building losses for three specific locations and we have collected two variables: water depth and inundation duration. We also want a full uncertainty distribution of the predictions. Then, we can specify the inputs variables in the right ordering and indicate that only nodes for which we have provided data are to be used for inference:

```
Values = [122 43; 65 16; 17 35]; % data for predictions
Nodes  = [1:2];        % use only nodes 1 and 2.
Output = 'full';       % returns a cell array containing results per
                       % each sample drawn.
```

Running the function with those inputs will return two 3x1 cell arrays the full posterior uncertainty distributions of the posterior uncertainty distribution of relative losses.

The example dataset, contained in the variable `german_floods`, includes synthetic data of 50 companies, which are based on German post-disaster surveys of flooded enterprises, but (for data protection reasons) do not represent any actual records from the surveys. The example data can be used by simply referring to the dataset:

```
[BRLOSS,ERLOSS] =
predict_commercial_flood_damage(german_floods,[1:5},'mean')
```