ETH zürich

UQLab User Manual Statistical Inference

E. Torre, S. Marelli, B. Sudret

$Y = \sum_{\alpha \in A} y_\alpha \Psi_\alpha(\boldsymbol{X})$
$\int \exp\left(-\frac{(t-\mu)^2}{2\sigma^2}\right) dt$
$\Phi(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} \exp(-t^2/2)\, dt$
$\Phi\left(\frac{x-\mu}{\sigma}\right)$
$\beta_{HL} = \frac{\mu_R - \mu_S}{\sigma_R^2 + \sigma_S^2}$
ELEVATION

Chair of Risk, Safety and Uncertainty Quantification
Stefano-Franscini-Platz 5
CH-8093 Zürich

Risk, Safety &
Uncertainty Quantification

How to cite UQLab

S. Marelli, and B. Sudret, UQLab: A framework for uncertainty quantification in Matlab, Proc. 2nd Int. Conf. on Vulnerability, Risk Analysis and Management (ICVRAM2014), Liverpool, United Kingdom, 2014, 2554-2563.

How to cite this manual

E. Torre, S. Marelli, B. Sudret, UQLab user manual – Statistical inference, Report UQLab-V2.0-114, Chair of Risk, Safety and Uncertainty Quantification, ETH Zurich, Switzerland, 2022

BibTeX entry

```
@TechReport{UQdoc_20_114,
author = {Torre, E. and Marelli, S. and Sudret, B.},
title = {{UQLab user manual -- Statistical inference}},
institution = {Chair of Risk, Safety and Uncertainty Quantification, ETH Zurich,
Switzerland},
year = {2022},
note = {Report UQLab-V2.0-114}
}
```

Document Data Sheet

Document Ref.	UQLAB-V2.0-114
Title:	UQLAB user manual – Statistical inference
Authors:	E. Torre, S. Marelli, B. Sudret Chair of Risk, Safety and Uncertainty Quantification, ETH Zurich, Switzerland
Date:	01/02/2022

Doc. Version	Date	Comments
V2.0	01/02/2022	UQLAB V2.0 release • Minor fixes
V1.4	01/02/2021	UQLAB V1.4 release • Minor overall improvements
V1.3	19/09/2019	Initial release

Abstract

UQLAB supports the statistical inference of probabilistic models by established inference tools. In the context of UQ, inference allows one to fit different probabilistic models to input data, thereby selecting the model that best describes them. The input joint PDF is represented in UQLAB by the marginal distributions of the input variables and their copula (see the companion UQLAB User Manual – the INPUT module). Consequently, UQLAB offers tools to infer both marginal distributions and copulas.

This user manual includes a review of the methods that are used to perform inference for marginals and different classes of copulas. After introducing the theoretical aspects, an in-depth example-driven user guide is provided to help users to build INPUT objects by inference. Finally, a comprehensive reference list of the methods and functions available for inference in the UQLAB INPUT module is given at the end of the manual.

Keywords: Probabilistic Input Model, Marginals, Copula, Inference

Contents

Chapter 1

Theory

1.1 Introduction

In applied problems, the probability distribution of uncertain quantities is often not known and has to be inferred from data. In some cases, expert knowledge or theoretical considerations allow one to assume at least a parametric form of the distribution, and inference is limited to estimating its parameters. Otherwise, the full shape of the distribution needs to be assessed, using either parametric or non-parametric inference. Multivariate distributions pose the additional challenge that, beside the marginal distribution of each component, their dependence structure, or copula, may also be unknown. This user manual describes how statistical inference can be performed in UQLAB to select, among a broad class of probabilistic models of marginal and copula distributions, the model that best fits a given dataset.
To avoid confusion, we stress the distinction between inference and fitting. Fitting is a procedure by which the parameters of a *given* parametric probability distribution are determined based on the data, so as to maximize the plausibility of that distribution. Inference is a more general term that encompasses fitting several probability distributions to the data, and then selecting the most plausible one. The measures of plausibility used for parameter fitting and for model selection may differ, and will be introduced below.
This manual focuses on the inference process only, while an extensive description of the probabilistic models supported by the UQLAB INPUT module is available in the UQLAB User Manual – the INPUT module.

1.2 Distribution fitting

Statistical inference is a process that typically requires two main steps: fitting a number of plausible probabilistic models to available data, and comparing them to select the best fitting one. Here we describe the fitting problem.

1.2.1 Fitting by maximum likelihood

Consider M uncertain, real-valued quantities grouped into a random vector $\boldsymbol{X} = (X_1, \dots, X_M)$, for which a probability distribution $F_{\boldsymbol{X}}(\boldsymbol{x}; \boldsymbol{\theta})$ with density $f_{\boldsymbol{X}}$ is assumed. The distribution has k real parameters $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_k)$, $k \geq 1$, which have to be determined based on a set of n independent observations $\mathcal{X} = \{\hat{\boldsymbol{x}}^{(1)}, \ldots, \hat{\boldsymbol{x}}^{(n)}\}$ of $\boldsymbol{X}$.

Various ways exist to estimate the parameters $\boldsymbol{\theta}$. By far the most common maximum likelihood (ML) estimation. The ML estimator of $\boldsymbol{\theta}$ based on the data set $\mathcal{X}$ is defined as

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \prod_{h=1}^{n} f_{\boldsymbol{X}}(\hat{\boldsymbol{x}}^{(h)}; \boldsymbol{\theta}) \overset{\text{def}}{=} \arg\max_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}). \tag{1.1}$$

The rationale is to seek the parameters $\hat{\boldsymbol{\theta}}$ that maximize the value of the joint probability of the observations in $\mathcal{X}$, namely their *likelihood*. The term $\mathcal{L}(\boldsymbol{\theta})$ in Eq. (1.1) is a function of $\boldsymbol{\theta}$ only, rather than of $\boldsymbol{x}$, and is known as the *likelihood function*. Maximizing the likelihood in (1.1) is equivalent to solving

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \log\left(\mathcal{L}(\boldsymbol{\theta})\right) = \arg\max_{\boldsymbol{\theta}} \sum_{h=1}^{n} \log(f_{\boldsymbol{X}}(\hat{\boldsymbol{x}}^{(h)}; \boldsymbol{\theta})). \tag{1.2}$$

Working with the sum of logarithms is often easier, both analytically and numerically, due to the inherently small values of $f_{\boldsymbol{X}}$, as well as the widespread occurrence of exponential forms in commonly used PDFs, and therefore preferred. The quantity

$$\log(\mathcal{L}) = \sum_{h=1}^{n} \log(f_{\boldsymbol{X}}(\hat{\boldsymbol{x}}^{(h)}; \hat{\boldsymbol{\theta}})) \tag{1.3}$$

is the *total log-likelihood* of $f_{\boldsymbol{X}}(\cdot; \hat{\boldsymbol{\theta}})$ on the data $\mathcal{X}$. It is a relative measure of goodness of fit of $f_{\boldsymbol{X}}$ on the data, and it is often used in model selection to identify the distribution that best fits the data (see Section 1.3.2).

ML solutions to the problem (1.2) are available analytically for some distributions. In the general case, various numerical algorithms exist to find approximate solutions. For instance, gradient methods start from an initial guess $\boldsymbol{\theta}_0$ of $\boldsymbol{\theta}$ and then iteratively move away from it towards the direction where the sum of log-likelihoods in (1.3) increases.

Note that not all probability distributions admit a finite or even a unique solution. For instance, truncated univariate distributions (see the UQLAB User Manual – the INPUT module) do not in the general case, even when their support is known. In UQLAB, these cases are handled separately by imposing bounds on the parameter estimates. The bounds can be specified as specific values, or as functions of the inference data.

1.2.2 ML estimation in the copula formalism

In UQLAB, an M-variate distribution $F_{\boldsymbol{X}}$ is generically specified in terms of a set of M univariate distributions $F_{X_1}, \ldots, F_{X_M}$, which describe the marginal behavior of $X_1, \ldots, X_M$, and a copula $C_{\boldsymbol{X}}$ that defines their mutual dependence:

$$F_{\boldsymbol{X}}(\boldsymbol{x}) = C_{\boldsymbol{X}}(F_{X_1}(x_1), F_{X_2}(x_2), ..., F_{X_M}(x_M)) \tag{1.4}$$

More details can be found in Section 1.3 of the UQLab User Manual – the INPUT module. In particular, as discussed in Section 1.3.2 of the same document, $C_{\boldsymbol{U}}$ is also the joint distribution of the random vector

$$\boldsymbol{U} = (F_{X_1}(X_1), \dots, F_{X_M}(X_M)).$$

Fitting F_X on $\mathcal{X}$ thus involves the following steps:

1. Separately fit the parameters of F_{X_i} on $\mathcal{X}_i = \{x_i^{(1)}, \dots, x_i^{(n)}\}$ to obtain the fitted marginal $\hat{F}_{X_i}$, $i = 1, \dots, M$;

2. Transform the data set $\mathcal{X}$ into the set of *pseudo-observations* $\mathcal{U} = \{\hat{\boldsymbol{u}}^{(1)}, \dots, \hat{\boldsymbol{u}}^{(n)}\}$, where

$$\hat{\boldsymbol{u}}^{(h)} := (\hat{F}_{X_1}(x_1^{(h)}), \dots, \hat{F}_{X_M}(x_M^{(h)})); \tag{1.5}$$

3. Fit the parameters of $C_{\boldsymbol{X}}$ on $\mathcal{U}$ to obtain the fitted copula $\hat{C}_{\boldsymbol{X}}$.

1.2.3 ML estimation for vine copulas

A case that deserves a separate discussion is that of vine copulas (for more details, see UQLab User Manual – the INPUT module, Section 1.4.4). From a theoretical standpoint, vine copulas are multivariate distributions (Joe, 2015). As such, their parameters can be estimated by maximum likelihood numerically. Nevertheless, vine copulas typically comprise a large number of parameters: an M-dimensional vine is obtained as the tensor product of $M(M-1)/2$ pair copulas, each typically entailing one or more parameters. Solving the maximum likelihood maximization problem (1.2) in high dimension is impractical and time consuming.

The very structure of a vine copula, however, suggests a considerably faster way to find an approximate solution. Rather than globally optimize the vector $\boldsymbol{\theta}$ of all vine parameters, one can estimate the parameters of each pair copula separately, starting from the pair copulas in the first tree of the vine (the unconditional pair copulas). Specifically, if $\mathcal{U}$ is the set of pseudo-observations defined in (1.5), a sequential (rather than global) estimation approach for vine copula parameters encompasses the following steps:

1. For each unconditional pair copula C_{ij}, which couples X_i and X_j, estimate its parameters $\boldsymbol{\theta}_{ij}$ from $\mathcal{X}_{ij} = \{(\hat{x}_i^{(h)}, \hat{x}_j^h),\ h = 1, \dots, n\}$, thereby obtaining $\hat{C}_{ij}$;

2. For each conditional pair copula $C_{jl|i}$, which couples X_j and X_l conditioned on X_i, introduce the *simplifying assumption* that it depends on U_i only through the argument $F_{j|i}$. Under this assumption:

   (a) Derive the univariate conditional distributions $\hat{C}_{j|i}(u_j|u_i)$ and $\hat{C}_{l|i}(u_l|u_i)$, where

$$\hat{C}_{j|i}(u_j|u_i) := \frac{\partial \hat{C}_{ij}(u,v)}{\partial u}\Big|_{(u,v)=(u_i,u_j)}.$$

   $\hat{C}_{j|i}$ is the CDF of U_j conditioned on U_i;

(b) Obtain the set of conditional pseudo-observations $\mathcal{U}_{jl|i} = \{(\hat{u}^{(h)}_{j|i}, \hat{u}^{(h)}_{l|i}), h = 1, \ldots, n\}$, where

$$\hat{u}^{(n)}_{j|i} := \hat{C}_{j|i}(\hat{u}^{(h)}_j | \hat{u}^{(h)}_i);$$

(c) Fit the parameters of $C_{jl|i}$ on $\mathcal{U}_{jl|i}$ to obtain the fitted copula $\hat{C}_{jl|i}$.

3. Iterate to fit the pair-copulas in deeper trees of the vine, that is, with more conditioning variables.

This general procedure needs to be adapted to the specific set of pair copulas that populate the vine. Algorithms for ML estimation of C- and D-vines were originally proposed by Aas et al. (2009), and are implemented in UQLAB.
Note that in the general case sequential fitting does not yield the global solution to problem (1.2). Nevertheless, it typically provides a sufficiently close approximation thereof, at a considerably reduced computational cost.

1.3 Model selection

Inference of a probabilistic model from data typically has to deal with the problem that the latter are not associated to a prescribed parametric probability distribution. In such circumstance, the question arises about which distribution should be fitted to the data. In practice, this problem can be dealt with by various approaches, briefly discussed below.

1.3.1 Non-parametric inference

Non-parametric inference assigns to the data a parameter-free model, rather than fitting a parametric one. An example of non-parametric model is the empirical CDF

$$H_{\mathcal{X}}(\boldsymbol{x}) = \frac{1}{n} \sum_{h=1}^{n} \mathbb{1}_{\{\hat{\boldsymbol{x}}^{(h)} \leq \boldsymbol{x}\}}, \tag{1.6}$$

where $\boldsymbol{x}' \leq \boldsymbol{x}$ if and only if $x'_i \leq x_i$ for each $i = 1, \ldots, M$. In words, $H_{\mathcal{X}}$ has a jump of $+1/n$ at each observation in $\mathcal{X}$. The corresponding probability mass function is a normalized sum of Dirac delta functions, each centered at one observation:

$$h_{\mathcal{X}}(\boldsymbol{x}) = \frac{1}{n} \sum_{h=1}^{n} \delta_{\{\hat{\boldsymbol{x}}^{(h)} \leq \boldsymbol{x}\}}. \tag{1.7}$$

Another example of non-parametric inference is kernel density estimation (KDE), also referred to as kernel smoothing. KDE replaces the delta functions of the empirical PDF with a non-negative function (kernel) $k(\cdot)$. The expression for the univariate KDE, often used for inference of marginal distributions, reads

$$\hat{f}_{\mathcal{X}}(x) = \frac{1}{nw} \sum_{h=1}^{n} k\left(\frac{||\hat{x}^{(h)} - x||}{w}\right). \tag{1.8}$$

The kernel is a decaying function of the distance between the point x where the distribution is evaluated and the point $\hat{x} \in \mathcal{X}$ where the kernel is centered. The parameter w, called the bandwidth, dictates how fast $\hat{f}_{\mathcal{X}}$ decays with $||x - \hat{x}||$. A common choice for the kernel is the standard Gaussian distribution. When the size n of the data set is large enough(*e.g.*., $n \geq 100$), the choice of the kernel function has a negligible effect on the final KDE estimate, provided that the kernel bandwidth w is properly selected according to n. A common choice when using the Gaussian kernel is Silverman's rule (Silverman, 1996)

$$w = \left(\frac{4\hat{\sigma}^5}{3n}\right)^{\frac{1}{5}} \simeq 1.06\hat{\sigma}n^{-1/5}, \tag{1.9}$$

where $\hat{\sigma}$ is the sample standard deviation of the data. This choice minimizes the mean integrated square error, but should be used with care. Indeed, it may yield widely inaccurate (for instance, strongly over-smoothed) estimates if the data come from a strongly skewed or multimodal distribution. Finally, KDE generalizes to the multivariate case (Simonoff, 1996), which is here omitted for simplicity.

Non-parametric inference typically produces models that fit the training data $\mathcal{X}$ very well, in the sense that their total log-likelihood (1.3) is high. However, these models exhibit poor prediction power, because they concentrate all probability mass at (for the empirical CDF) or around (for KDE) the observed points, thereby assigning negligible probability elsewhere (overfitting).

1.3.2 Parametric inference and selection criteria

Parametric inference produces probability distributions that are less flexible than empirical or kernel smoothing ones, as they have a fixed analytical expression that is described by just a few parameters. The parameters of any such probability distribution are fitted to the data as described in Section 1.2. The question then becomes which of the many known probability distributions to fit to the data. At least in low dimension ($M = 1$ or 2), a plot of the data or other considerations may help to discard *a priori* some families of probability distributions and retain others. For instance, univariate data with a semi-finite support or with a multimodal histogram cannot have originated from a Gaussian distribution, while very skewed data cannot come from a symmetric distribution.

Once a set of possible families of parametric distributions has been determined, the family that best describes the data must be selected. Various selection criteria exist to this end. The following applies to both marginal distributions and copulas, unless stated otherwise.

The *maximum likelihood criterion* selects the fitted distribution with the highest log-likelihood score (see Eq. (1.3)). This choice, however, tends to favor distributions with a larger number of parameters, which more easily adapt to the data but possibly lead to overfitting. To avoid overfitting, a penalty term on the number of model parameters can be introduced.

The *Akaike Information Criterion* (AIC; Akaike, 1974) selects the distribution which minimizes the quantity

$$\mathrm{AIC} = 2k - 2\log(\mathcal{L}), \tag{1.10}$$

where k is the number of model parameters.

An even stronger penalization by the number of distribution parameters is the *Bayesian Information Criterion* (BIC; Schwartz, 1978), which minimizes

$$\mathrm{BIC} = \log(n)k - 2\log(\mathcal{L}), \tag{1.11}$$

where n is the number of data points used to fit the distribution.

Sometimes, maximizing the total likelihood (with or without penalization) produces a probability distribution with a substantial peak centered where most data points accumulate, and too little probability mass elsewhere. An example of this behavior is shown in Figure 1. A set of $1\,000$ data points is drawn from a Gaussian distribution truncated to the left ($\mu = 0$, $\sigma = 1$, support $[1, +\infty)$; true distribution and histogram of the data shown in the left panel). Five different families are then fitted to the data: Gaussian, truncated Gaussian, Weibull, Gamma, and Lognormal distributions. For the latter four, the truncation interval is specified as the true one, $[1, +\infty)$. The Gamma distribution yields the lowest AIC, despite visually exhibiting the largest deviation from the histogram of the data. In this case, one would intuitively want to select the distribution which most closely follows the data histogram.

This intuition is formalized by the *Kolmogorov-Smirnov distance* (K-S) criterion. The criterion selects the family whose cumulative distribution has the lowest maximum distance from the empirical CDF of the data (1.6), that is, which minimizes

$$d_{\mathrm{KS}} = \max_{\boldsymbol{x} \in \mathbb{R}^M} |F_{\boldsymbol{X}}(\boldsymbol{x}) - H(\boldsymbol{x})|. \tag{1.12}$$

In the example shown in Figure 1, right panel, the K-S criterion would select the Lognormal distribution (cyan) among those illustrated in the figure.

Currently, the K-S criterion in supported in UQLAB for univariate distributions only.

1.3.3 Pair-copula selection for vine copulas

Selecting the vine copula that best fits a data set $\mathcal{X}$, or its counterpart $\mathcal{U}$ defined in (1.5), among all existing vines of dimension M, involves the following steps:

1. Selecting a vine structure (for C- and D-vines: selecting the order of the nodes);
2. Selecting the parametric family of each pair copula;
3. Fitting the pair copula parameters to $\mathcal{U}$.

Steps 1-2 form the representation problem (for more details on vine copula representation in UQLAB, please refer to the UQLAB User Manual – the INPUT module). Concerning step 3, algorithms to compute the likelihood of C- and D-vines (Aas et al., 2009) and of R-vines (Joe, 2015) given a data set $\mathcal{U}$ exist, enabling parameter fitting based on maximum likelihood. In

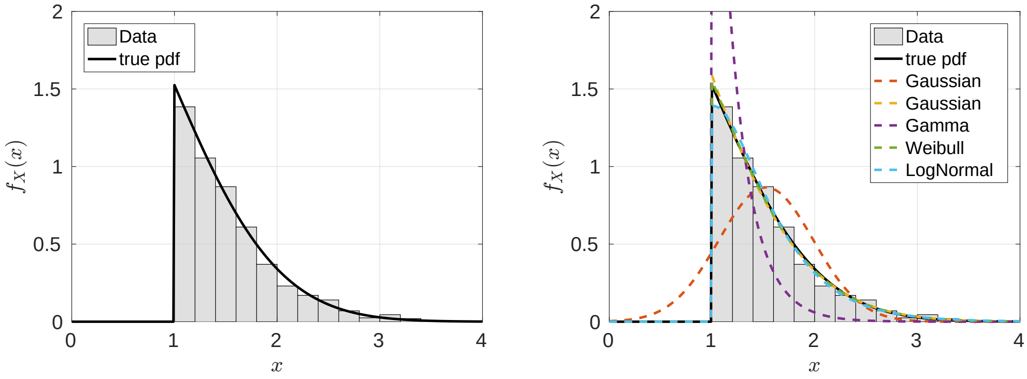

Figure 1: Example of inference on a dataset generated from a truncated Gaussian distribution. The left panel shows the normalized histogram of the data and the true PDF. The right panel shows different parametric distributions fitted to the data. The Lognormal distribution (cyan) would be selected by the Kolmogorov-Smornov (KS) criterion.

principle, the vine copula that best fits the data may be determined by iterating maximum likelihood fitting over all possible vine structures and all possible parametric families of comprising pair copulas. In practice, however, this approach is computationally infeasible in even moderate dimension M due to the large number of possible structures (Morales-Nápoles, 2011) and of pair copulas comprising the vine. Aas et al. (2009) therefore suggested a different, faster approach which first solves step 1 separately. The optimal vine structure is selected heuristically so as to capture first the pairs $(X_i,\, X_j)$ with the strongest dependence (which then fall in the upper trees of the vine). Kendall's tau (Stuart et al., 1999) is the measure of dependence of choice, defined by

$$\tau_{ij} = \mathbb{P}((X_i - \tilde{X}_i)(X_j - \tilde{X}_j) > 0) - \mathbb{P}((X_i - \tilde{X}_i)(X_j - \tilde{X}_j) < 0), \qquad (1.13)$$

where $(\tilde{X}_i,\, \tilde{X}_j)$ is an independent copy of $(X_i,\, X_j)$. If the copula of $(X_i,\, X_j)$ is C_{ij}, then

$$\tau_K(X_i, X_j) = 4 \int\int_{[0,1]^2} C_{ij}(u,\, v) dC_{ij}(u, v) - 1. \qquad (1.14)$$

For a C-vine, ordering the variables $X_1, \ldots, X_M$ in decreasing order of dependence corresponds to selecting the central node in tree T_1 as the variable X_{i_1} which maximizes $\sum_{j \neq i_1} \tau_{i_1 j}$, then the node of tree T_2 as the variable X_{i_2} which maximizes $\sum_{j \notin \{i_1, i_2\}} \tau_{i_2 j}$, and so on. For a D-vine, this means ordering the variables $X_{i_1}, X_{i_2}, \ldots,\, X_{i_M}$ in the first tree so as to maximize $\sum_{k=1}^{M-1} \tau_{i_k i_{k+1}}$, which can be solved as an open traveling salesman problem (OTSP; Applegate et al., 2006). UQLab solves the OTSP problem looping through all paths if $M \leq 9$, and stochastically using a genetic algorithm otherwise. An algorithm to find the optimal structure for more general regular vines has been proposed by Dißmann et al. (2013). For a selected vine structure, the corresponding pair copulas are inferred following a sequential approach analogous to the one outlined in Section 1.2.3: each pair copula is selected among the allowed families separately, thereby avoiding to compare all combinations. Any of the selection criteria illustrated in Section 1.3.2 can be used.

Finally (and optionally), one can keep parametric expression of the vine determined through the sequential approach, and perform global likelihood maximization. Note that this step consists in the search for a maximum in a k-dimensional space, where k is the (usually large) total number of parameters of the vine, and may therefore be extremely time consuming while improving the quality of the fit only marginally.

1.4 Separating random variables into mutually independent subgroups (block independence)

A useful preprocessing step for data from a multivariate sample consists in determining groups of random variables that exhibit inter-group independence. That is, taken any pair (X_1, X_2) of random variables such that X_1 and X_2 belong to different groups, X_1 and X_2 are mutually independent. Once these groups, or blocks, have been identified by means of an appropriate block independence test, their copula can be expressed as the tensor product of the copulas of the individual subgroups (see the UQLAB User Manual – the INPUT module, Eq. (1.8)).

The copula of each subgroup can be then inferred from data as explained below. Of course, the block independence test can be skipped and an individual copula can be inferred from the full dataset for the entire input random vector. However, inferring lower-dimensional copulas for independent subgroups separately is typically computationally cheaper, and leads to more interpretable dependence structures.

Chapter 2

Usage

2.1 Inference of marginals

The following sections describe how to infer marginal distributions on available data. All examples refer to a univariate problem, which requires the user to specify a single marginal `iOpts.Marginals(1)`. Multivariate cases can be handled analogously by specifying the additional marginals `iOpts.Marginals(2)`, `iOpts.Marginals(3)`, and so on, consistently with the basic UQLAB INPUT syntax (see UQLAB User Manual – the INPUT module). All marginals are treated separately.

All UQLAB options to perform inference of marginal distributions on data are listed in Section 3.1 (see Table 4 and subsequent ones).

2.1.1 Fully automated inference of marginals

The most extensive (least informed) form of statistical inference consists in finding the distribution family that best represents the data, and the parameters of such distribution that best fit the data. Given a data set `X`, inferring the family of each marginal distribution across all families supported in UQLAB, and fitting its parameters, is possible with the code

```
iOpts.Inference.Data = X;
iOpts.Copula.Type = 'Independent';
InputHat = uq_createInput(iOpts);
```

For each column i in the matrix `X`, UQLAB loops across all supported univariate distributions, fits them all to the data in that column, and assigns the selected distribution to the i-th marginal of `InputHat`. In particular:

- The optimal parameters for each family are obtained by likelihood maximization, that is, by solving (1.1) or the equivalent problem (1.2)
- The distribution family that best fits the data is selected according to the default Akaike information criterion (AIC, see (1.11)). Different criteria can be specified through the field `iOpts.Inference.Criterion` (see Table 1);
- Only supported parametric families are considered by default (no KDE). For a list of supported families and their properties, see the companion UQLAB User Manual – the

INPUT module, Appendix A;

- UQLAB automatically discards those parametric families that are incompatible with the data (for instance, distributions with positive support for data with negative elements);
- In the example above, the copula is set to be the independence copula. If the copula is not specified, it is also inferred among all supported copula types (see Section 2.2).

The explicit, but in this case superfluous, code for automatic inference of the marginals on data `X` with `M` columns would be to set all marginal types to `'auto'`:

```
for ii = 1:M
    iOpts.Marginals(ii).Type = 'auto';
end
```

Note: For the inference of the marginals, UQLab uses the Matlab built-in function `fitdist`. During the fitting process, this function might generate warnings. This should not impact the quality of the resulting fit as long as the inferred distribution type did not generate any warnings.

2.1.2 Specifying inference options for each marginal separately

In the above examples, the (implicit) inference options specified in `iOpts.Inference` are identical for all marginals. Furthermore, the set of families considered for inference is `'auto'` for all marginals. UQLAB also allows one to specify each inference option separately for each marginal. For instance,

```
iOpts.Marginals(i).Inference.Criterion = 'KS';
```

overwrites the default inference criterion `'AIC'` with `'KS'` (Kolmogorov-Smirnov distance) for marginal `i`.

Analogously, the data to be used for inference can be assigned to each marginal field separately, as a one-dimensional array:

```
iOpts.Marginals(i).Inference.Data = x_new;
```

`iOpts.Marginals(i).Inference` accepts the same fields as `iOpts.Inference`, but the options specified in the former are specific to one marginal only, and have priority over those specified in the latter.

The distribution families to be tested can also be assigned explicitly for each marginal, as a `cell` of `char` strings (one per distribution family):

```
iOpts.Marginals(i).Type = {'Lognormal', 'Exponential', 'Weibull'};
```

All marginals for which a specific field is not assigned explicitly are assigned default values. Using the three lines of code above, each j-th marginal, $j \neq i$, will be inferred among all supported families (as with the implicit code `iOpts.Marginals(j).Type = 'auto'`) on the data `X(:,j)` specified in `iOpts.Inference.Data` (if any), using the AIC criterion (`iOpts.Marginals(j).Inference.Criterion = 'AIC'`).

Finally, inference works for all distributions, including user-defined ones (see Section 2.1.5) and truncated distributions. For instance, to specify that marginal 2 must be inferred as a truncated distribution in the interval $[a, b]$, write

```
iOpts.Marginals(2).Bounds = [a, b];
```

Infinite values for the bounds are accepted.

Note: Note that it is not currently possible in UQLAB to infer the support bounds of a truncated distribution. The only exception is the uniform distribution, the parameters of which are also its bounds.

2.1.3 Fitting or fixing parameters for a given family

The simplest form of parametric inference consists in fitting the parameters of a given probabilistic input model. The following code sets `Marginals(1)` as a Gaussian distribution with parameters obtained by likelihood maximization:

```
iOpts.Marginals(1).Type = 'Gaussian';
```

Again, all marginals can be assigned different `Type`(s). Those whose `Type` is not specified are assigned the default `'auto'` (inference performed among all supported parametric families). Following the same scheme, any marginal can be fully specified by assigning its `Parameters` (or alternatively its `Moments`), as in

```
iOpts.Marginals(1).Parameters = [0,1];
```

If so, no inference takes place for that marginal.

2.1.4 Non-parametric inference: kernel smoothing

Kernel smoothing (1.8) can be added to the list of marginal types to be considered for inference by

```
iOpts.Marginals(i).Type = {'ks', ...};
```

If kernel smoothing is the only distribution type desired, one can simply write

```
iOpts.Marginals(i).Type = 'ks';
```

In this case, inference reduces to setting the marginal's parameters to the observations, used as centers of the kernels. This can be done explicitly, thus avoiding inference, by setting

```
iOpts.Marginals(i).Parameters = x_i;
```

were `x_i` are the observations from marginal i (see also the UQLAB User Manual – the INPUT module, Section 2.1.1.2).

The kernel type and bandwidth can be set explicitly in the `Marginals(i).Options` criteria. For instance, the code below uses a triangular kernel, with a bandwidth equal to 0.1:

```
iOpts.Marginals(i).Options.Kernel = 'Triangular';
iOpts.Marginals(i).Options.Bandwidth = 0.1;
```

The default kernel type is Gaussian, and the default kernel bandwidth is set using Silverman's rule (1.9). For a list of supported values, see Table 6.

2.1.5 Fitting a custom marginal distribution

While not supporting by default all known families of univariate parametric distributions, UQLAB allows the user to specify custom marginals (see the UQLAB User Manual – the INPUT module, Section 2.7). Performing inference on a custom marginal named 'myCustom' requires the following ingredients:

- a script uq_pdf_myCustom.m that computes the marginal PDF. The function must take as the first input a column array X of points where the PDF is calculated, and optionally as a second input the parameters of the distribution (as an array). More details are available in the companion UQLAB User Manual – the INPUT module, Section 2.7;
- the parameter bounds;
- an initial guess for each parameter.

While default values for the parameter bounds and the initial guess are known for standard families, they must be provided for custom distributions. Both can be provided either as fixed values (here, for a distribution 'myCustom' of 2 parameters):

```
iOpts.Marginals(1).Inference.Families = 'myCustom';
iOpts.Marginals(1).Inference.ParamBounds.myCustom = [-5, 5; 0.1 10];
iOpts.Marginals(1).Inference.ParamGuess.myCustom = [0; 1];
```

or as functions of the data:

```
iOpts.Marginals(1).Inference.Families = 'myCustom';
iOpts.Marginals(1).Inference.ParamBounds.myCustom = {...
    @(x) mean(x)-std(x), @(x) mean(x)+std(x); ...
    @(x) std(x)/4,       @(x) 4*std(x)};
iOpts.Marginals(1).ParamGuess.myCustom = {@mean; @std};
```

See also Table 7.

Note: ParamBounds and ParamGuess are structures that take as fields the names of the custom marginals to fit to the data. A different custom distribution named 'myNewCustom' would require the parameter bounds / initial guesses to be specified under ParamBounds.myNewCustom / ParamGuess.myNewCustom

2.1.6 Goodness of fit of inferred marginals

Once inference is completed and the corresponding INPUT object InputHat has been generated, a summary of the inference results for each marginal i can be found under InputHat.Marginals(i).GoF. The latter is a struct that contains various goodness of fit measures for each distribution that has been fitted to the data. For instance, if the exponential distribution was fitted to the data, then

```
InputHat.Marginals(i).GoF.Exponential
```

is a `struct` with the following fields:

- `.LL`: the distribution's log-likelihood;
- `.AIC`: the distribution's AIC;
- `.BIC`: the distribution's BIC;
- `.KSstat`: the distribution's K-S statistic;
- `.KSpvalue`: the p-value of the Kolmogorov-Smirnov test, which represents the approximate probability that the data come from the specified (in this case, exponential) distribution.

Note: `InputHat.Marginals(i).GoF` contains such information for each distribution that has been fitted during the inference process, not only for the distribution that was eventually selected as the best-fitting one.

2.2 Inference of copulas

UQLAB currently supports five types of copulas: the independent copula, pair copulas, the multivariate Gaussian copula, and two classes of vine copulas: the C-vine and the D-vine. More details on theoretical aspects and properties of these copulas can be found in the companion UQLAB User Manual – the INPUT module. All UQLAB options to perform copula inference are listed in Section 3.1 (see Table 8 and subsequent ones).
(See also: `uq_inferCopula`).

2.2.1 Fully automated inference of copula

Inference among all copulas (and marginals) supported in UQLAB can be done by

```
iOpts.Inference.Data = X;
InputHat = uq_createInput(iOpts);
```

The above code is equivalent to additionally specifying (before the `uq_createInput` command):

```
iOpts.Copula.Type = 'auto';
```

Specifically:

- if `X` is an array with only 1 column, no copula exists (the problem is univariate);
- if `X` has 3 or more columns, a block independence test is first performed to identify possible subgroups of random variables being mutually independent (see Section 1.4); a copula is fitted to each subgroup.

Note: The block independence test can be skipped; in this case, a single subgroup is identified, being equivalent to full input random vector.

For $M = 2$, the block independence test reduces to a pair-independence test

- for each subgroup `X_i`:
  - if `X_i` has 2 columns, a pair independence test is first performed. If the test is passed, the two variables are assigned the independence pair copula; otherwise, all other supported pair copula families are fitted to the corresponding data.
  - if `X_i` has three or more columns, pair copulas are excluded and all other copula types are fitted instead (for exactly three columns, the class of D-vines is also excluded as it is equivalent to the class of C-vines). For default inference options assigned to specific copula types, see the next sections.
  - the best fitting copula is retained.

2.2.2 Testing for block independence

By default, data with dimension 3 or larger are tested for block independence before parametric inference actually takes place. The test divides the data into inter-independent blocks, and copula inference is performed on each block separately. The final copula is the product of the individual copulas inferred for each block.

Two sets of random variables form two separate blocks if and only if each variable in one block is independent of each variable in the other block. Thus, block independence consists of a number of tests of pairwise independence. The code

```
iOpts.Inference.BlockIndepTest.Alpha = 0.05;
iOpts.Inference.BlockIndepTest.Type = 'Kendall';
iOpts.Inference.BlockIndepTest.Correction = 'none';
```

instructs each pair independence test to be performed using the Kendall's tau statistic, at a significance threshold $\alpha = 0.05$, using no statistical correction for the number of pairwise tests being performed in total. These are also the default test options. The list of supported values is illustrated in Table 3.

To turn off the block independence test, simply type

```
iOpts.Inference.BlockIndepTest.Alpha = 0;
```

Also notice that the block independence test options can be provided as copula inference options rather than general options, such as,

```
iOpts.Copula.Inference.BlockIndepTest.Alpha = 0.05;
```

If specified in this way, the options under `iOpts.Inference.BlockIndepTest` are ignored.

2.2.3 Specification of inference data for the copula

The data for copula inference can be specified through either one of the following fields:

- `iOpts.Inference.Data`. In this case, the data are used for inference of both the marginals and the copula. For the copula, they are transformed into pseudo-observations in $[0,1]^M$ through (1.5);
- `iOpts.Copula.Inference.Data`, equivalent to the former;
- `iOpts.Copula.Inference.DataU`, if the data are pseudo-observations in $[0,1]^M$, obtained from data in the physical space through (1.5).

The latter two are mutually exclusive and cannot be both assigned. If specified, they both take priority over the first option, which is then ignored if also provided.

2.2.4 Inference among a selected list of copula types

Inference among a selected list of copula types can be performed by specifying a `cell` of copula types, such as

```
iOpts.Copula.Type = {'DVine', 'CVine'};
```

Each copula type may involve additional mandatory or optional subfields of `.Inference`, as described in the next sections and summarized in Table 10-Table 11.
The code

```
iOpts.Copula.Type = 'DVine';
```

instead limits inference to a specific copula type (here, D-vines).

2.2.5 Testing for pair independence

The independence copula $C(u,v) = uv$ assigns two random variables statistical independence. One may therefore expect that inference would assign two independent random variables the independent copula. However, this is not necessarily the case. Due to randomness in the data, inference based solely on (penalized) likelihood maximization criteria may select a different parametric family. For `'Pair'` copulas and vines (`'CVine', 'DVine'`), it is possible to perform statistical testing to address this issue directly. This can be done by specifying the field `iOpts.Inference.PairIndepTest` or, equivalently, `iOpts.Copula.Inference.PairIndepTest` (the former is ignored if the latter is provided). For instance, the code

```
iOpts.Copula.Inference.PairIndepTest.Type = 'Kendall';
iOpts.Copula.Inference.PairIndepTest.Alpha = 0.05;
```

instructs UQLab to perform a classical independence test based on Kendall's tau prior to inferring the copula, setting the significance threshold to $\alpha = 0.05$. If the resulting p-value is larger than α, then the null hypothesis of independence is accepted and the independence copula is selected. Otherwise, inference proceeds as usual. For vines, it is also possible to correct the threshold by the total number of tests performed. For instance, Bonferroni correction sets the effective significance threshold of a set of T independent tests performed over the same data, each with threshold α, to α/T (Shaffer, 1995). It can be requested by

```
iOpts.Copula.Inference.PairIndepTest.Correction = 'Bonferroni';
```

The option is ignored for `'Pair'` copulas, for which a single test is performed.

For an extensive list of all the available options for tests of pair independence, see Table 2.

2.2.6 Inference of pair copulas

Pair copulas describe the dependence among two random variables. As such, their inference requires bivariate data to be performed on. In this section, the data `X` used for inference is an array with two columns. The resulting INPUT object is bivariate, with two marginals and a two-dimensional copula among them.

The following code infers the copula as a pair copula among all supported pair copula families (the marginals are also inferred, among all supported marginals, as described in Section 2.1.1):

```
iOpts.Inference.Data = X;
iOpts.Copula.Type = 'Pair';
InputHat = uq_createInput(iOpts);
```

The different lines of code instruct UQLAB that the input copula

1. is to be obtained by inference on data `X`,
2. as a pair copula,
3. selecting among all supported pair copula families. This can be made explicit by the (redundant) declaration (to be made before the `uq_createInput` command):

   ```
     iOpts.Copula.Inference.PCfamilies = 'auto';
   ```

4. trying any rotation of the pair copula density: 0, 90, 180 and 270° (or only 0 and 90° for families with a symmetric density with respect to the main diagonal, that is, such that $c(u,v) = c(v,u)$ for all $u, v \in [0,1]^2$).

A selected list of pair copula families can be specified as a `cell` of `char`s, one per family name, such as

```
iOpts.Copula.Inference.PCfamilies = {'Gaussian', 'Frank', 'Clayton'};
```

To infer the copula, UQLAB transforms the data $\mathcal{X}$ into pseudo-observations $\mathcal{U}$ in the unit hypercube $[0,1]^M$ through (1.5). This and alternative supported ways to specify inference data for copulas are illustrated in 2.2.3.

It is possible to infer the copula on data X_{new} that are not used for inference of the marginals, by typing

```
iOpts.Copula.Inference.Data = Xnew;
```

In this case, `iOpts.Inference.Data` is ignored for copula inference, if existing.
Similarly, pseudo-observations $\mathcal{U}$ can be directly provided for copula inference by

```
iOpts.Copula.Inference.DataU = U;
```

`X` and `U` are $n \times 2$ arrays: each column contains observations of one random variable.
The default selection criterion is the AIC. It can be changed to the BIC or the ML ones by

```
iOpts.Copula.Inference.Criterion = 'BIC'; % or 'ML'
```

Note that this will overwrite the criterion possibly specified in `iOpts.Inference.Criterion`

2.2.7 Inference of Gaussian copulas

The following code fits the parameters of a Gaussian copula:

```
iOpts.Copula.Type = 'Gaussian';
```

The parameters of the Gaussian copula are obtained as the Spearman's correlation coefficients between all pairs of random variables.

2.2.8 Inference of vine copulas

The following code infers a C-vine copula with fixed structure $3-1-2$ (that is, comprising pair copulas $C_{3,1}$, $C_{3,2}$ and $C_{1,2|3}$) on a 3-dimensional data set `X`. The pair copulas are inferred among the Frank, Gaussian, and t- families only:

```
iOpts.Copula.Type = 'CVine';
iOpts.Copula.Inference.CVineStructure = [3 1 2];
iOpts.Copula.Inference.PCfamilies = {'Frank', 'Gaussian', 't'};
```

The vine structure, in particular, determines which pair copulas are to be explicitly part of the vine, and therefore need to be inferred. In this case, the vine comprises the copulas $C_{3,1}$ between X_3 and X_1, $C_{3,2}$ between X_3 and X_2, and $C_{1,2|3}$ between $X_{1|3}$ and $X_{2|3}$. $C_{3,1}$ is inferred from the data `X(:,3)` and `X(:,1)`, and so on.
It is possible to retrieve this information before performing inference by using the function `uq_CopulaSummary` (see also Section 3.8.5 of the UQLab User Manual – the INPUT module):

```
uq_CopulaSummary('CVine', [3 1 2])
```

which returns the output

```
CVine, dimension 3, structure [3 1 2]. Pair copulas:
Index | Pair Copula
==================================
1     | C_3,1
2     | C_3,2
3     | C_1,2|3
```

A D-vine could be inferred analogously by specifying

```
iOpts.Copula.Type = 'DVine';
iOpts.Copula.Inference.DVineStructure = <array>;
```

As for the `'Pair'` copula case, vine copulas admit inference among a selected list of families, rather than all supported ones (see Table 10). For instance, one can specify

```
iOpts.Copula.Inference.PCfamilies ={'t', 'Frank', 'Gumbel'};
```

Typically, the vine structure is also unknown and needs to be inferred (see Section 1.3.3). This is done, for a C-vine, by specifying

```
iOpts.Copula.Inference.CVineStructure = 'auto';
```

and analogously for a D-vine. If the structure is not specified, it is set to `'auto'` by default. Finally, pair independence tests may be performed prior to solving the inference problem, as illustrated in Section 2.2.5.
Once a vine copula `myVine` has been inferred from data, its graphical structure can be visualized by the code

```
uq_drawVineCopula{myVine};
```

Information on the constituing pair copulas can be printed on screen by

```
uq_CopulaSummary{myVine};
```

2.2.9 Goodness of fit of inferred copulas

Once inference is completed and the corresponding INPUT object `InputHat` has been generated, a summary of the results can be found in the field `InputHat.Copula.GoF`. The latter is a `struct` that contains various goodness of fit measures for each copula that has been fitted to the data. For instance, if the `'Gaussian'` copula was fitted to the data, then

```
InputHat.Copula.GoF.Gaussian
```

is a `struct` object with the following fields:

- `.LL`: the copula's total log-likelihood over the data used for inference;
- `.AIC`: the copula's AIC;
- `.BIC`: the copula's BIC;

Note: `InputHat.Copula.GoF` contains such information for the best fitting copula (the one whose parameters maximize the likelihood function over the inference data) of each type selected for inference in `iOpts.Copula.Inference.Types`, not only for the type that was eventually selected as the overall best.

Chapter 3

Reference List

How to read the reference list

Structures play an important role throughout the UQLab syntax. They offer a natural way to semantically group configuration options and output quantities. Due to the complexity of the algorithms implemented, it is not uncommon to employ nested structures to fine-tune the inputs and outputs. Throughout this reference guide, a table-based description of the configuration structures is adopted.
The simplest case is given when a field of the structure is a simple value or array of values:

Table X: Input			
●	.Name	String	A description of the field is put here

which corresponds to the following syntax:

```
Input.Name = 'My Input';
```

The columns, from left to right, correspond to the name, the data type and a brief description of each field. At the beginning of each row a symbol is given to inform as to whether the corresponding field is mandatory, optional, mutually exclusive, etc. The comprehensive list of symbols is given in the following table:

●	Mandatory
□	Optional
⊕	Mandatory, mutually exclusive (only one of the fields can be set)
⊞	Optional, mutually exclusive (one of them can be set, if at least one of the group is set, otherwise none is necessary)

When one of the fields of a structure is a nested structure, a link to a table that describes the available options is provided, as in the case of the Options field in the following example:

Table X: Input

●	.Name	String	Description
□	.Options	Table Y	Description of the Options structure

Table Y: Input.Options

●	.Field1	String	Description of Field1
□	.Field2	Double	Description of Field2

In some cases, an option value gives the possibility to define further options related to that value. The general syntax would be:

```
Input.Option1 = 'VALUE1' ;
Input.VALUE1.Val1Opt1 = ...;
Input.VALUE1.Val1Opt2 = ...;
```

This is illustrated as follows:

Table X: Input

●	.Option1	String	Short description
		'VALUE1'	Description of 'VALUE1'
		'VALUE2'	Description of 'VALUE2'
⊞	.VALUE1	Table Y	Options for 'VALUE1'
⊞	.VALUE2	Table Z	Options for 'VALUE2'

Table Y: Input.VALUE1

□	.Val1Opt1	String	Description
□	.Val1Opt2	Double	Description

Table Z: Input.VALUE2

□	.Val2Opt1	String	Description
□	.Val2Opt2	Double	Description

Note: In the sequel, double and doubles mean a real number represented in double precision and a set of such real numbers, respectively.

3.1 Create an Input with inferred marginals and/or copula

Syntax

```
InputHat = uq_createInput(iOpts)
```

The struct variable iOpts contains the configuration information for the input object whose marginals and/or copula are to be inferred from data.

Note: Inference and representation can be mixed: one may, for instance, declare some marginals to be inferred while others are known, or infer the marginals but not the copula, and so on. The inference options described in this chapter are applied only to the distributions to be inferred, and ignored for the others.

3.1.1 General inference options (valid for marginals and copula)

The list of general inference options valid for both marginals and copulas in provided in Table 1. Each of these options can also be provided for a specific marginal (resp. copula) under iOpts.Marginals(ii).Inference (resp. iOpts.Copula.Inference). The latter two (see Table 5 and Table 10) take priority and overwrite the first, in case both are provided.

Table 1: iOpts.Inference

●[1]	.Data	n-by-M array. n instances of dimension M	Inference data.
□	.Criterion	String default: 'AIC'	Selection criterion (Section 1.3.2)
		'ML'	Maximum likelihood
		'AIC'	Akaike information criterion
		'BIC'	Bayesian information criterion
		'KS'	Kolmogorov-Smirnov CDF distance. Valid only for marginals (by default set to 'AIC' for copulas instead)
□	.PairIndepTest	Struct	Specifications for the pair independence test (see Table 2). Concerns copula inference only
□	.BlockIndepTest	Struct	Specifications for the block independence test (see Table 3). Concerns copula inference only

Table 2: iOpts.Inference.PairIndepTest

□	.Alpha	scalar in $[0, 1]$ default: 0.1	Statistical threshold for the test of pair independence. Set to 0 or [] to disable the test.

[1] Does not need to be specified if and only if both iOpts.Marginals.Inference.Data and iOpts.Copula.Inference.Data or iOpts.Copula.Inference.DataU are provided instead

□	.Type	String default: 'Kendall'	Test statistic. One of 'Kendall', 'Spearman', or 'Pearson'
□	.Correction	String default: 'auto'	Correction for the number of pairwise tests. Applies to inference of vine copulas only:
		'none'	No statistical correction
		'fdr'	False discovery rate correction
		'Bonferroni'	Bonferroni correction
		'auto' or ''	No correction if $M \leq 8$, FDR correction otherwise

Table 3: iOpts.Inference.BlockIndepTest

□	.Alpha	scalar in $[0, 1]$ default: 0.1	Statistical threshold for the test of block independence. Set to 0 or [] to disable the test.
□	.Type	String default: 'Kendall'	Test statistic. One of 'Kendall', 'Spearman', or 'Pearson'
□	.Correction	String default: 'auto'	Correction for the number of pairwise tests:
		'none'	No statistical correction
		'fdr'	False discovery rate correction
		'Bonferroni'	Bonferroni correction
		'auto' or ''	No correction if $M \leq 8$, FDR correction otherwise

3.1.2 Options for inference of marginal distributions

The detailed list of options available for inference of marginal distributions is reported in the tables below. For a list of marginal distribution families currently supported in UQLAB, see the UQLAB User Manual – the INPUT module, Appendix A.

Table 4: iOpts.Marginals(ii)

□	.Type	char or cell of chars default: 'auto'	As a char, can be • 'auto': marginal inferred among all parametric distributions • the name of any supported or custom marginal distribution As a cell, contains the name of the marginal distributions among which to select the inferred one

□	`.Inference`	`struct`	Inference options for marginal ii, see Table 5. If provided, overwrites the corresponding options specified in `iOpts.Inference`
□	`.Bounds`	array $[a, b]$, $-\infty \leq a < b \leq +\infty$	Support of truncated distribution (Section 1.2.3 of the UQLab User Manual – the INPUT module)
□	`.Options`	struct	Kernel smoothing options (see Table 6)

Table 5: `iOpts.Marginals(ii).Inference`

□[2]	`.Data`	array $n \times 1$	Inference data for marginal ii
□	`.Criterion`	String default: `'AIC'`	Selection criterion (see Section 1.3.2)
		`'ML'`	Maximum likelihood
		`'AIC'`	Akaike information criterion
		`'BIC'`	Bayesian information criterion
		`'KS'`	Kolmogorov-Smirnov CDF distance

Table 6: `iOpts.Marginals(ii).Options`

□	`.Kernel`	Char default: `'Gaussian'`	Kernel shape
		`'Gaussian'` or `'Normal'`	
		`'Triangle'` or `'Triangular'`	
		`'Box'`	
		`'Epanechnikov'`	
□	`.Bandwidth`	Double default: set using Silverman's rule (1.9)	Kernel bandwith

Additional options for custom marginals

if `iOpts.Marginals(ii).Type` contains the name of one or more custom distributions `'myCustom'`, `iOpts.Marginals(ii).Inference` requires the following additional fields:

Table 7: `iOpts.Marginals(ii).Inference (For custom distributions)`

[2] Mandatory if `iOpts.Inference.Data` is not provided

●	.ParamBounds	struct	Parameter bounds
□	.ParamGuess	struct default: struct([])	Starting guess for optimal parameter search

where .ParamBounds.myCustom and .ParamGuess.myCustom are a $k \times 2$ and $k \times 1$ array, respectively (one row per parameter) whose elements can be:

- doubles (can be $\pm\infty$), *i.e.*, fixed values;
- inline functions, evaluated on the inference data.

3.1.3 Copula inference options

Options for inference of copulas

The detailed list of options available for inference of copulas is reported in the Tables below. For a list of copula types currently supported in UQLAB, see the UQLAB User Manual – the INPUT module, Section 1.4.
For a list of supported pair copula families, see in particular the UQLAB User Manual – the INPUT module, Table 1.

Table 8: iOpts.Copula

□	.Type	char or cell of chars default: 'auto'	As a char, can be • 'auto': copula inferred among all supported types • the name of any supported copula type As a cell, contains the copula types among which to perform inference
●	.Inference	struct	Inference options, see Table 10

Table 9: iOpts.Copula.Inference

⊕	.Data	array $n \times M$	Data in the physical space to use for copula inference
⊕	.DataU	array $n \times M$	Data in the unit hypercube to use for copula inference
□	.Criterion	String default: 'AIC'	Copula selection criterion (see Section 1.3.2)
		'ML'	Maximum likelihood
		'AIC'	Akaike information criterion
		'BIC'	Bayesian information criterion

□	.PairIndepTest	Struct	Specifications of test for pair independence (see Table 2)
□	.BlockIndepTest	Struct	Specifications of test for block independence (see Table 3)

If `iOpts.Copula.Type` includes any of `'Pair'`, `'CVine'` or `'DVine'`, or is the string `'auto'`, the following optional field is supported:

Table 10: `iOpts.Copula.Inference (For pair or vine copulas)`

□	.PCfamilies	String or cell of strings default: `'auto'`	Name(s) or copula types to choose from. If `'auto'`, try all supported parametric families

If `iOpts.Copula.Type` includes `'CVine'` or `'DVine'`, or is the string `'auto'`, the following optional fields are also accepted:

Table 11: `iOpts.Copula.Inference (for C- and D- vines)`

□	.CVineStructure	String `'auto'` or array (any permutation of `1:M`) default: `'auto'`	C-vine structure. If `'auto'`, it is inferred from data (Section 1.3.2)
□	.DVineStructure	String `'auto'` or array (any permutation of `1:M`) default: `'auto'`	D-vine structure. If `'auto'`, it is inferred from data (Section 1.3.2)

3.2 Working with inferred inputs

Using an input object whose marginals and/or copula were obtained by inference works analogously to using any input. Methods such as `uq_getSample`, `uq_print`, `uq_display`, `uq_enrichSobol`, etc., work as usual (see the UQLAB User Manual – the INPUT module).

The inference options specified for the individual marginals and/or for the copula can be retrieved into the subfield `InputHat.Marginals(ii).Inference` and `InputHat.Copula.Inference`. Similarly, various goodness-of-fit measures for the inferred distribution over the data used for inference are stored under `InputHat.Marginals(ii).GoF` and `InputHat.Copula.GoF`.

3.3 Additional functions

3.3.1 `uq_inferMarginals`

Routine called by `uq_createInput(iOpts)` when `iOpts.Marginals(ii).Type = 'auto'` for some `ii`.

Syntax

```
myMarginals = uq_inferMarginals(X, Marginals)
myMarginals = uq_inferMarginals(X, Marginals, ParamBounds, ParamGuess)
```

```
[myMarginals, GoF] = uq_inferMarginals(...)
```

Input

myMarginals = uq_inferMarginals(X, Marginals) returns a struct object MyMarginals which defines a set of marginals distributions obtained by inference over the data X:

- myMarginals has the same number of elements of the input argument Marginals, which is also the number of columns of X;
- for each ii such that Marginals(ii).Type = 'auto', myMarginals(ii) is obtained by inference; all other elements of myMarginals are copies of Marginals, and treated as known distributions;
- if any custom distribution has been included in the list of distribution families used for inference, that is, if there exists any ii such that Marginals(ii).Type = 'auto' and Marginals(ii).Inference.Families includes the name of a custom distribution, then the additional input arguments ParamBounds and ParamGuess need to be provided (as detailed in Table 7).

[myMarginals, GoF] = uq_inferMarginals(X, Marginals) additionally provides the struct with goodness-of-fit measures for each univariate family that has been fitted to data (see Section 2.1.6).

3.3.2 uq_inferCopula

Routine called by uq_createInput(iOpts) when iOpts.Copula.Type = 'auto'.

Syntax

```
myCopula = uq_inferCopula(U, Copula)
myCopula = uq_inferMarginals(U, Copula, options)
[myCopula, GoF] = uq_inferCopula(...)
```

Input

myCopula = uq_inferCopula(U, Copula) returns a struct object MyCopula which defines a copula obtained by inference over the data U:

- U must contain values in the interval $[0, 1]$. These values are obtained from data X in the physical space (original observations) by the probability integral transform (see the UQLAB User Manual – the INPUT module, Equation (1.20)). If X was drawn from an input with marginals MyMarginals, then U=uq_all_cdf(X, myMarginals);
- Copula must be specified as detailed in Table 8 and following;

`myCopula = uq_inferCopula(U, Copula, alpha, stat, correction)` additionally performs tests of pair independence for pair and vine copulas:

- `alpha` is the significance threshold of the test. The null hypothesis of independence is rejected if the test p-value is lower than `alpha`;
- `stat` specifies the statistic used for the test. It can be either `'Kendall'` (default), `'Pearson'`, or `'Spearman'`. The corresponding test is performed;
- For vine copulas, `corr = 'Bonferroni'` allows to correct the threshold `alpha` by the number `T` of pairwise tests ($T = M(M-1)/2$): each test is rejected if its p-value is lower than `alpha/T`. Default: `corr='none'` (no correction).

`[myMarginals, GoF] = uq_inferCopula(U, Copula)` additionally provides the `struct` with goodness-of-fit measures for each copula that has been fitted to data (see Section 2.2.9).

3.3.3 uq_inferVineStructure

Routine called by `uq_createInput(iOpts)` when a C-vine is to be inferred and `iOpts.Copula.Inference.CVineStructure = 'auto'` (or when a D-vine is to be inferred and `iOpts.Copula.Inference.DVineStructure = 'auto'`).

Syntax

```
S = uq_inferVineStructure(U, VineType)
[S, K] = uq_inferVineStructure(U, VineType)
```

Input

`S = uq_inferVineStructure(U, VineType)` returns a $1 \times M$ array `S` which infers the optimal structure of a vine copula of type `VineType` on data `U`:

- `U` must contain values in the interval $[0, 1]$. These values are obtained from data `X` in the physical space (original observations) by the probability integral transform (see the UQLab User Manual – the INPUT module, Equation (1.20)). If `X` was drawn from an input with marginals `MyMarginals`, then `U=uq_all_cdf(X, myMarginals);`
- `VineType` can be either `'CVine'` or `'DVine'`.

`[S, K] = uq_inferVineStructure(U, VineType)` additionally returns the sum `K` of empirical Kendall's tau of all pairs of variables explicitely coupled by a pair copula in the vine with the inferred structure `S`.

References

Aas, K., C. Czado, A. Frigessi, and H. Bakkend (2009). Pair-copula constructions of multiple dependence. *Insurance: Mathematics and Economics 44*(2), 182–198. 4, 6, 7

Akaike, H. (1974). A new look at the statistical model identification. *IEEE Transactions on Automatic Control 19*(6), 716–723. 5

Applegate, D. L., R. E. Bixby, V. Chvátal, and W. J. Cook (2006). *The Traveling Salesman Problem: A Computational Study*. NewJersey: Princeton University Press. 7

Dißmann, J., E. C. Brechmann, C. Czado, and D. Kurowicka (2013). Selecting and estimating regular vine copulae and application to financial returns. *Computational Statistics and Data Analysis 59*, 52–69. 7

Joe, H. (2015). *Dependence modeling with copulas*. CRC Press. 3, 6

Morales-Nápoles, O. (2011). Counting vines. In D. Kurowicka and H. Joe (Eds.), *Dependence Modeling: Vine Copula Handbook*, Chapter 9, pp. 189–218. World Scientific Publisher Co. 7

Schwartz, G. (1978). Estimating the dimension of a model. *Annals Stat. 6*(2), 461–464. 6

Shaffer, J. P. (1995). Multiple hypothesis testing. *Annual Review of Psychology 46*, 561–584. 16

Silverman, B. W. (1996). *Density estimation for statistics and data analysis*, Volume 26 of *Monographs on Statistics and Applied Probability*. Chapman & Hall. 5

Simonoff, J. S. (1996). *Smoothing Methods in Statistics.* Springer. 5

Stuart, A., K. Ord, and S. Arnold (1999). *Kendall's advanced theory of statistics Vol. 2A – Classical inference and the linear model* (6th ed.). Arnold. 7