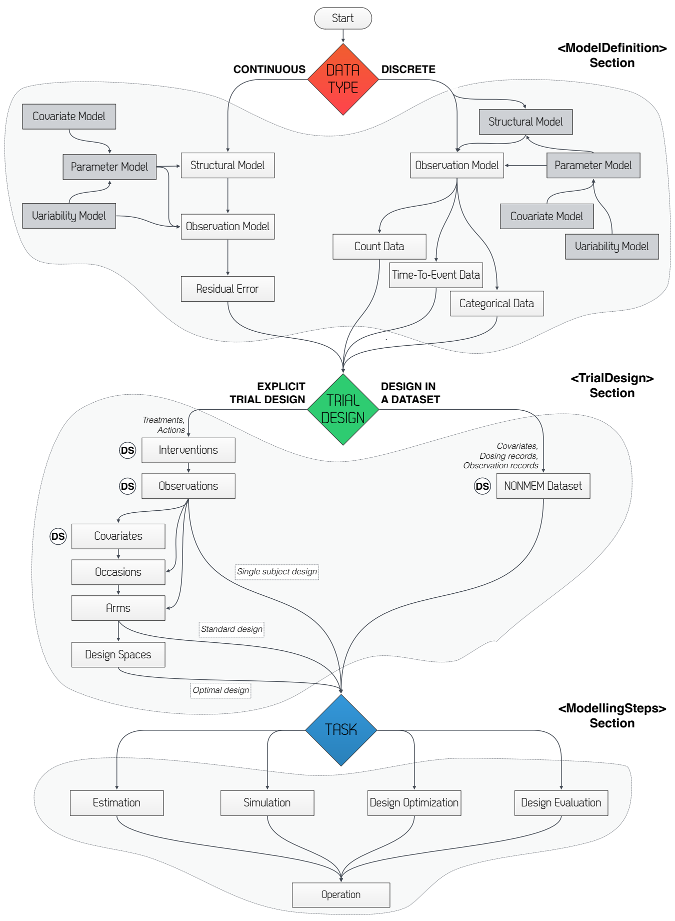
Internal Release

Extensions in PharmML 0.7.1

Authors:
Maciej J Swat
Sarala Wimalaratne
Niels Rode Kristensen



July 2, 2015

Contents

1 Overview **2**

1.1 Major changes/extensions in version 0.7 . . . 2

2 *ProbOnto* – Ontology/Knowledge Base of Probability Distributions **4**

2.1 Ontology versus Knowledge Base . . . 5

2.2 Limitations of the application of UncertML in PharmML . . . 5

2.3 ProbOnto Features . . . 6

2.3.1 Features under construction . . . 6

2.4 Working with ProbOnto . . . 7

2.4.1 New elements supporting ProbOnto . . . 7

2.5 Annotation of models with ProbOnto ontology . . . 7

2.5.1 Implementation in PharmML . . . 7

2.5.2 Annotation of PharmML . . . 8

2.5.3 Background Information is Contained in ProbOnto . . . 8

2.6 Alternative parameterisations – examples . . . 9

2.6.1 Negative binomial distribution . . . 9

2.6.2 Normal distribution . . . 9

2.6.3 Log-normal distribution . . . 10

2.7 ProbOnto examples . . . 15

2.7.1 Categorical data models . . . 15

2.7.2 Count data models . . . 17

2.7.3 Truncated distributions . . . 19

2.8 Independency of ProbOnto . . . 20

Chapter 1

Overview

This document describes extensions and changes in PharmML compared to version 0.6 released in January 2015.

1.1 Major changes/extensions in version 0.7

The following table summarises the major changes described in detail in following chapters. Please, note that this list is not exhaustive.

PharmML element or modelling aspect	version $\leq$ 0.6	version 0.7
	ProbOnto	
Probability distributions – annotation and encoding of statistical models, chapter 2	UncertML used for encoding of distributions but no support for annotations	NEW `<Distribution>` element with children `<ProbOnto>` and `<UncertML>` NEW *ProbOnto* - ontology and knowledge base of probability distribution – support for expressions as parameters – support for distribution functions and quantities – more then 50 discrete and continuous distributions and/or alternative parameterisations
	Models	
Covariates, parameters and observations model, chapter ??	*Distribution*-type not available	NEW *Distribution*-type model can be used with either UncertML or ProbOnto
Individual parameter model, section ??	3 types supported: – *Equation*-type – *Gaussian* with linear – *Gaussian* with nonlinear covariate model	4 types supported – MODIFIED *Structured*-type `<GaussianModel>` renamed to `<StructuredModel>` with linear/nonlinear covariate model – `<PopulationParameter>` element renamed to `<PopulationValue>` – *Equation*-type (no changes) – NEW *Distribution*-type model using either UncertML or ProbOnto NEW `<RandomEffectMapping>` to map variability levels in *Distribution* -type models
Population parameter, section ??	`<SimpleParameter>` used with no distribution support	NEW `<PopulationParameter>` element replaces `<SimpleParameter>` element
Observation model section ??	Gaussian and *Equation*-type	NEW *Distribution*-type

Covariate model, section ??	Transformation, interpolation and distribution of covariates supported	NEW Support for conditional distributions wrt covariates or design elements
section ??	not supported	`<CovariateModel>` has an additional option for creating new covariates out of exiting ones
Matrix/Vector operators, chapter ??	not supported basic types	NEW `<MatrixUniOp>` element with values *inverse, trace and transpose*
Transformation element, section ?? and ??	supported	new structure with attribute `type` to be assigned values such as *log*, *logit* etc. NEW *BoxCox*
Count data models, section ??		NEW `<NumberCounts>` element for variable *k*
	BAYESIAN INFERENCE & HIERARCHICAL MODELS	
Population parameter, chapter ??	not supported	NEW `<PopulationParameter>` with *Distribution*-type or *Equation*-type
	TRIAL DESIGN	
Structure, chapter ??	CDISC based	redesigned based on WP3 design proposal – all design elements are in `<TrialDesign>` – dataset reference `<ExternalDataSet>` relocated to trial section
Lookup table	Available in `<Administration>`	Moved to `<Observations>`
Simulation Step	reference to interventions not possible	`<InterventionsReference>` element NEW
	GENERAL	
Interval, section ??	not available	NEW `<Interval>` with left/right-endpoints of closed/open `type` attribute. `closed` is the default value.
Box-Cox transformation applied to observations and parameters section ??	not available	`<Transformation>` with new value *BoxCox* for the `type` attribute and `<Parameter>` child element for *lambda* parameter
Missing data, section ??	only *NA* supported in inline datasets	– Inline datasets – NEW elements `<NaN>`, `<minusInf>`, `<plusInf>`, `<ALQ>`, `<BLQ>` – External datasets – NEW elements `<MissingData>` with attributes `dataCode` and `missingDataType` with values: {*NA, NaN, plusInf, minusInf, BLQ, ALQ*}
Dataset headers, section ??	not supported	NEW elements in dataset – `<Definition>`: `<Header>` with attributes `name`, `headerType`, `rowNumber` – `<Table>`: `<HeaderRow>` with attribute `order`
Regressors, section ??	supported when using datasets and lookup tables	NEW attribute `regressor` with values *yes/no* added to `<Variable>` element
SO column types, section ??	no SO specific types supported	NEW values for the `columnType` introduced: {*indivParameter, popParameter, randEffect, residual, strataVariable, statPrecision, structParameter, varParameter*}

Table 1.1: Overview of major differences between versions 0.7 and 0.6

Chapter 2

ProbOnto – Ontology/Knowledge Base of Probability Distributions

Background When encoding probabilistic uncertainties using a parametric distribution its name and parameters are sufficient to specify it in an unambiguous way as in most cases such parameter set is unique. But, because for a number of cases two or more parameterisations exist, one needs to be precise what parameters are used when referring to a distribution, otherwise one might end up with a wrong model (see for an example Figure 2.1). For this purpose an external standard reference is very useful as it allows to considerably reduce the effort of declaring the required distribution in a language such as MDL or PharmML.

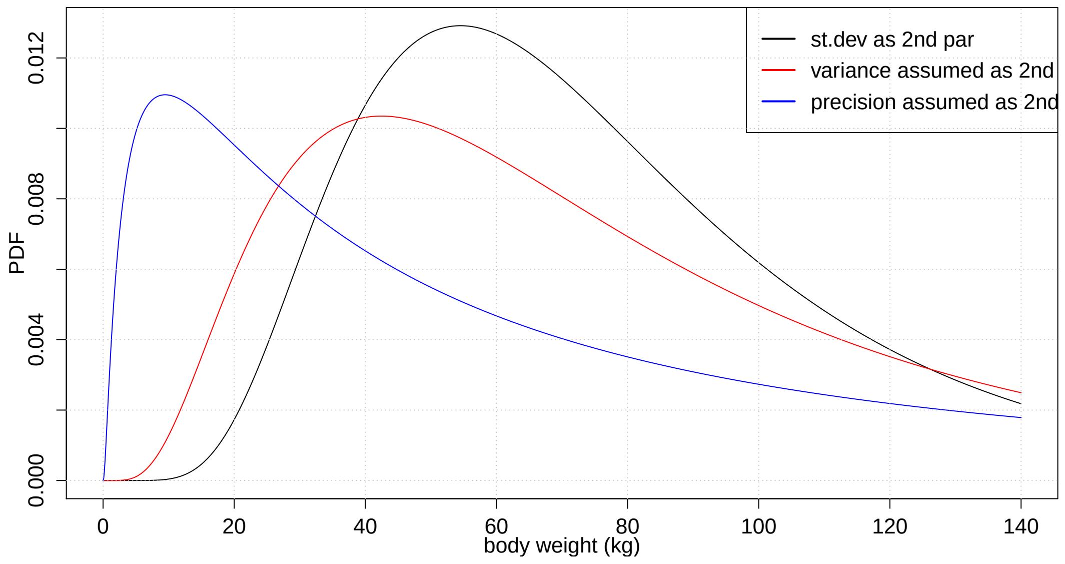


Figure 2.1: Illustration of possible model misspecification when using incorrect parameterisation. (1) The black curve corresponds to a log-normal distribution of body weight, $\mathcal{LN}(\mu = \log(70), \sigma = 0.5)$, the intended parameterisation. (2) Here it was mistakenly assumed that 0.5 corresponds to the variance and calculation of standard deviation as required by the R function *dlnorm* gives $\sigma = \sqrt{0.5} = 0.707$ (red). (3) Here the modeller assumed that the 2nd input value corresponds to the precision and calculated the standard deviation as $\sigma = 1/\sqrt{0.5} = 1.41$ (blue). Small numerical differences in σ, on the log-scale, result in significant differences on the natural scale.

Until now, we have been relying on the UncertML, [**?**], which provides means to encode in MDL/PharmML a range of continuous and discrete uni/multi-variate probability distributions. However, from the perspective of PharmML, it has several limitations as described in section 2.2.

Idea The initial motivation for *ProbOnto* was to create an ontology of probability distributions purely for annotation purposes. Many resources are available online and in printed format but no proper ontology exists so far[1]. Similarly, the databases of distributions available online come with analog issues[2]. It turns out that *ProbOnto* can be very helpful in designing a flexible alternative for UncertML with many additional features. It can be used e.g. in PharmML or other target tools/languages *both* as ontological resource for annotation purposes and as a knowledge base, see next section for their definitions, to provide the means to specify a wide range of distributions and distribution related functions and quantities.

In fact, such solution is indispensable in the face of requirements posed by models we would like to encode currently and in the foreseeable future.

2.1 Ontology versus Knowledge Base

Ontology is a formal representation of a domain of knowledge. It is an abstract entity defining the vocabulary for a domain and the relations between concepts. However, an ontology doesn't specify how that knowledge is stored (as physical file, in a database, or in some other form), and how the knowledge can be accessed.

Knowledge base is a physical artifact. It is a database, a repository of information that can be accessed and manipulated in some predefined fashion.

The knowledge in a knowledge base is modelled according to rules and relationships defined in an ontology.

2.2 Limitations of the application of UncertML in PharmML

Although very useful to a certain extent, there are limitations in the design and scope of UncertML making the encoding of some probability distributions cumbersome or even impossible, see examples below. Here some known limitations (in the order of severity):

- it doesn't support the assignment of expressions for distribution parameters or the specification of block references, which is required if the parameter in question is defined elsewhere in the model.
- it doesn't cover many distributions used in Pharmacometrics, e.g.
  - multivariate continues distributions such as Inverse-Wishart
  - discrete distributions such as Generalized Poisson, Zero-inflated Poisson etc.
  - or alternative parameterisations for distributions such as Negative Binomial, Log-Normal etc.
- `<degreesOfFreedom>` parameter element of the Wishart distribution doesn't support referencing a variable (required for Bayesian inference) – a known bug/limitation but with no solution for now.
- UncertML is a reference resource for distributions but does not provide mechanisms to retrieve programmatically related functions and quantities.

Other minor issues:

- the implementation of `<MultivariateNormalDistribution>` requires the specification of the `dimension` attribute of the covariance matrix – although this can be estimated it requires unnecessary calculations when translating models to PharmML.
- every extension requires changes in the already complex XML schema.
- doesn't support the precision parameter, τ, used in winBUGS rather then standard deviation or variance and precision matrix, T, instead of covariance matrix Σ, see tables 2.2 and 2.4.
- version 3.0 which we currently use is not yet released publicly, the UncertML website is not updated and 3.0 documentation is not available.

[1]For example, the Statistics Ontology, STATO, `http://stato-ontology.org/`, provides for most distributions merely a link to an external reference/definition. No parameters or related functions and quantities are defined in the ontology making their annotation impossible. Other ontologies, we have analysed number of them featured in the BioPortal, [?], suffer from equivalent limitations as they are designed in a similar way.

[2]Distributome, `http://www.distributome.org/`, comes with an impressive and well referenced collection of 90+ distributions but doesn't contain many of relevant for us types and/or parameterisations and is limited to univariate parametric ones.

UncertML extension A seemingly easy solution would be to extend UncertML but to do so, it would mean to introduce major extensions and changes to its current XML schema. Only the support of the most important missing features would de facto require to rewrite the entire standard, as UncertML doesn't possess the structure to encode even basic expressions. And because it would most certainly result in a different, compared to PharmML, mathematical notation we would be faced with inconsistent, layered and/or overlapping schemas difficult to handle and to process.

Suggested way forward ProbOnto offers an alternative solution allowing to avoid all the limitations listed above while providing number of additional features and means to build in a very flexible probability distribution support in MDL, PharmML and other languages/tools within DDMoRe and beyond.

2.3 ProbOnto Features

- General
  - Covers more then 50 distributions and alternative parameterisations.
  - Supports encoding of univariate mixture distributions and truncation bounds (open/closed).
  - Allows for easy encoding of distributions and related functions in target tools/languages thanks to its generic format.
  - Doesn't enforce specific implementation in target tools.
  - In PharmML only few extensions were required to provide flexible encoding support for all distributions and their features, see section 2.4.
  - Collection of supported distributions, see appendix **??** for some selected types and their essential features, is easily extendable without non or limited impact on the PharmML structure.
  - All mathematical functions and quantities are available in Latex and for a number of functions R-code is provided.
- ProbOnto as Ontology
  - It can be used to annotate statistical models based on supported probability distributions, e.g. their name, parameters, truncation bounds, their defining functions and quantities.
- ProbOnto as Knowledge Base
  - Provides for each distribution either PDF or PMF and in many cases also other distribution related functions such as CDF, hazard and survival functions – the level of coverage depends on the particular distribution.
  - Provides related quantities such as mean, median, mode, variance etc.
  - Provides other info about *support/range* and relationships to other distributions.

The distribution collection and their features are based on probability distribution pages of the english Wikipedia[3], Forbes et al. 2010 [?], Leemis et al. 2008 [?], Song & Chen 2011 [?], and Wolfram MathWorld [?].

2.3.1 Features under construction

The following features are under construction and not available in the current release

- truncation bounds – supported already for all univariate distributions but an extension to multivariate distributions is needed.
- non-parametric distributions.

They are available to certain extend in UncertML, which can be used instead for the time being, if required.

[3]See the list of distributions on Wikipedia at `https://en.wikipedia.org/wiki/List_of_probability_distributions`

2.4 Working with ProbOnto

The subsequent chapters come with a number of examples of ProbOnto use but it is worth to point out two basic implementation rules

- The name of a distribution, encoded in the `<DistributionName>` tag, must be one of the 'Code names' assigned to each distribution in ProbOnto using the `name` attribute.
- The same holds of the parameters of a distribution, encoded in the `<Parameter>` tag. The parameter 'Code names' are specified using also a `name` attribute. The order of parameters doesn't matter.

To remain consistent with the nomenclature used so far in PharmML and MDL (which was based on UncertML vocabulary) the majority of parameter names is identical to those used in UncertML. For new distributions and their parameters we have defined the most common names used in the literature. In tables 2.3, 2.5 and 2.6 we have compiled the *code names*.

Example 1. The implementation of the negative binomial model illustrates how this works. There are two parametrisations for this distribution but the version with Poisson intensity, λ, and over-dispersion, τ, as parameters, with the code name, *NegativeBinomial2*, is frequently used in discrete data models.

```
<Distribution>
    <ProbOnto name="NegativeBinomial1">
        <Parameter name="rate">
            <ct:Assign>
                <ct:SymbRef blkIdRef="pm1" symbIdRef="rabbit"/>
            </ct:Assign>
        </Parameter>
        <Parameter name="overdispersion">
            <ct:Assign>
                <ct:SymbRef blkIdRef="pm2" symbIdRef="piggy"/>
            </ct:Assign>
        </Parameter>
    </ProbOnto>
</Distribution>
```

According to the rules, the names of the distributions and their parameters must be the code names defined by ProbOnto, see table 2.5. The user can then assign any symbols to the parameters, with *rabbit* for *rate*, defined in parameter model `pm1` and *piggy* for *overdispersion*, defined in parameter model `pm2`.

2.4.1 New elements supporting ProbOnto

The following elements are new in this version to support ProbOnto encoding

- `<ProbOnto>` tag with the `name` attribute for the distribution code names with children elements
  - `<Parameter>` with the `name` attribute for the parameter code names. It can be assigned any expression.
  - `<LowerTruncationBound>` and `<LowerTruncationBound>` to indicate the truncation bounds for univariate distributions with attribute `type` which can be either *closed* or *open*.
  - `<MixtureComponent>` with the `name` attribute for the code name of mixture component.

2.5 Annotation of models with ProbOnto ontology

2.5.1 Implementation in PharmML

The following code shows the typical Poisson model implementation

```
<PMF linkFunction="log">
    <Distribution>
        <ProbOnto id="X1" name="Poisson">
            <Parameter id="X2" name="rate">
                <ct:Assign>
                    <ct:SymbRef blkIdRef="pm1" symbIdRef="lambda"/>
                </ct:Assign>
            </Parameter>
        </ProbOnto>
    </Distribution>
</PMF>
```

2.5.2 Annotation of PharmML

Notice that in the above sniper of code the elements defining the distribution used, `<ProbOnto>`, and its parameter, `<Parameter>` are given identifiers, `id="X1"` and `id="X2"`, respectively. This allows us to annotate these elements so that we can make explicit their intended interpretation. Using the PharmML metadata annotation schema, we would record such interpretation using the property *has-interpreted-type*. In what follows, 'ps' abbreviates the namespace for this schema and 'probonto' abbreviates the namespace for the ProbOnto ontology, part of the stack of ontologies used in PharmML annotation.

# The distribution element is interpreted as an instantiation of the Poisson distribution.

X1 ps:has-interpreted-type probonto:0000111.

# The parameter element is interpreted as an instantiation of the parameter element, λ, of the Poisson distribution.

X2 ps:has-interpreted-type probonto:0000114.

In ProbOnto, *0000111* and *0000114* are the identifiers for the Poisson distribution and its (unique) parameter, respectively. The two statements above encode the interpretation of the PharmML code defining the distribution. Such statements can in principle be generated automatically after processing the PharmML code.

Annotating the actual PharmML model and the element to which the distribution applies would involve more or a variation upon the above to the effect that the ProbOnto distribution is identified.

2.5.3 Background Information is Contained in ProbOnto

Given the annotation of the PharmML code linking to ProbOnto, we can then use ProbOnto to make explicit all the information that is packed into these two very terse annotation statements.

Underlying accessible knowledge about Poisson distribution

We thus have access to the following regarding the distribution contained in the PharmML code (as much as is contained in the ProbOnto definition of the Poisson distribution):

name	Poisson (ID: 0000111)
type	discrete
variate	k, scalar
support	$k \in \{0, 1, 2, 3, \dots\}$

Additionally, we can obtain from ProbOnto the following type of information.

Underlying accessible knowledge about the related functions

PMF

$$\frac{\lambda^k}{k!} e^{-\lambda}$$

PMF in R

```
lambda^k/factorial(k) * exp(-lambda)
```

CDF

$$\frac{\Gamma(\lfloor k+1 \rfloor, \lambda)}{\lfloor k \rfloor !}$$

CDF in R

```
Igamma(floor(k+1), lambda, lower=F) / factorial(floor(k))
```

using *Igamma* from `http://cran.r-project.org/web/packages/zipfR/zipfR.pdf`.

Underlying accessible knowledge about the (rate) parameter

name	Poisson intensity (ID: 0000114)
type	scalar
symbol	λ
definition	$\lambda \in R, \lambda > 0$

The amount of information that may be encoded in ProbOnto is extensible. Thus, via a very simple and direct mechanism of annotation that amounts to linking a distribution and its parameter(s) in a piece of PharmML code, we can inherit and obtain all the background relevant information. This knowledge can be used either for our understanding and the validation of our PharmML encoding or, with adequate software support, for processing by tools.

Currently, such extensive software support is not available but is part of the development path for PharmML and its implementation of ProbOnto.

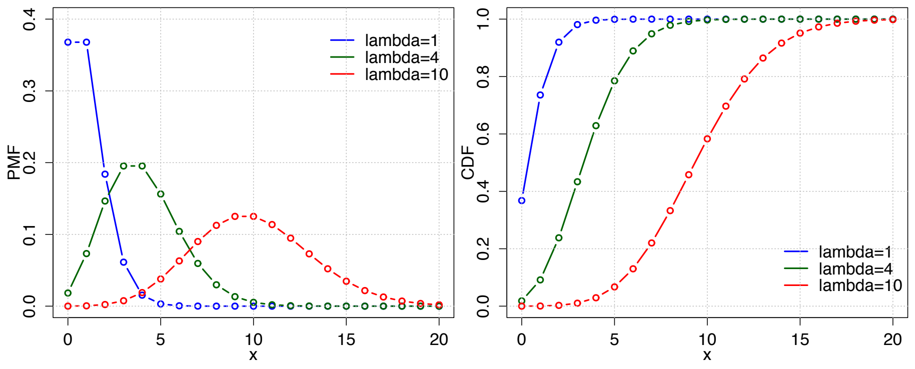


Figure 2.2: PMF and CDF of the Poisson distribution plotted using the R-code stored in ProbOnto.

2.6 Alternative parameterisations – examples

Providing alternative parameterisations is required for number of reasons, such as model type, application area, available data and target tool – e.g. BUGS using precision, τ, rather then standard deviation or variance for a number of distributions. (See also a discussion on parameters difference between BUGS and R, [?]). A few typical examples are given in next sections.

2.6.1 Negative binomial distribution

The available parameterisations, among others, are

- NegativeBinomial1 $(r,\, p)$ with r – *number of failures* and p – *success probability*,

$$P(y = k; r, p) = \binom{k + r - 1}{k}(1 - p)^r p^k$$

- NegativeBinomial2 $(\lambda,\, \tau)$ with λ – *Poisson intensity* and τ – *over-dispersion*,

$$P(y = k; \lambda, \tau) = \frac{\Gamma(k + \frac{1}{\tau})}{k!\,\Gamma(\frac{1}{\tau})}\left(\frac{1}{1 + \tau\lambda}\right)^{\frac{1}{\tau}}\left(\frac{\lambda}{\frac{1}{\tau} + \lambda}\right)^k$$

with the latter being used in typical pharmacometric discrete data models, [?, ?]. See the Wikipedia article[4], explaining the reasons behind the various representations.

2.6.2 Normal distribution

Available parameterisations (see also Table 2.1 with indication about their coverage in target tools) are

- Normal1 $(\mu,\, \sigma)$ with μ – *mean* and σ – *standard deviation*,
- Normal2 $(\mu,\, v)$ with μ – *mean* and v – *variance*,
- Normal3 $(\mu,\, \tau)$ with μ – *mean* and τ – *precision* $(\tau = 1/\sigma^2)$

[4]`en.wikipedia.org/wiki/Negative_binomial_distribution`, section 'Alternative formulations'

Re-parameterisation formulas

In this case the recalculation between the representations are very simple but are given here for the completeness.

- $\mathbf{N1}(\mu,\,\sigma) \rightarrow \mathbf{N2}(\mu,\,v)\ :\ \mu \rightarrow \mu;\quad \sigma \rightarrow v = \sigma^2$
  $\mathbf{N2}(\mu,\,v) \rightarrow \mathbf{N1}(\mu,\,\sigma)\ :\ \mu \rightarrow \mu;\quad v \rightarrow \sigma = \sqrt{v}$
- $\mathbf{N1}(\mu,\,\sigma) \rightarrow \mathbf{N3}(\mu,\,\tau)\ :\ \mu \rightarrow \mu;\quad \sigma \rightarrow \tau = 1/\sigma^2$
  $\mathbf{N3}(\mu,\,\tau) \rightarrow \mathbf{N1}(\mu,\,\sigma)\ :\ \mu \rightarrow \mu;\quad \tau \rightarrow \sigma = 1/\sqrt{\tau}$
- $\mathbf{N2}(\mu,\,v) \rightarrow \mathbf{N3}(\mu,\,\tau)\ :\ \mu \rightarrow \mu;\quad v \rightarrow \tau = 1/v$
  $\mathbf{N3}(\mu,\,\tau) \rightarrow \mathbf{N2}(\mu,\,v)\ :\ \mu \rightarrow \mu;\quad \tau \rightarrow v = 1/\tau$

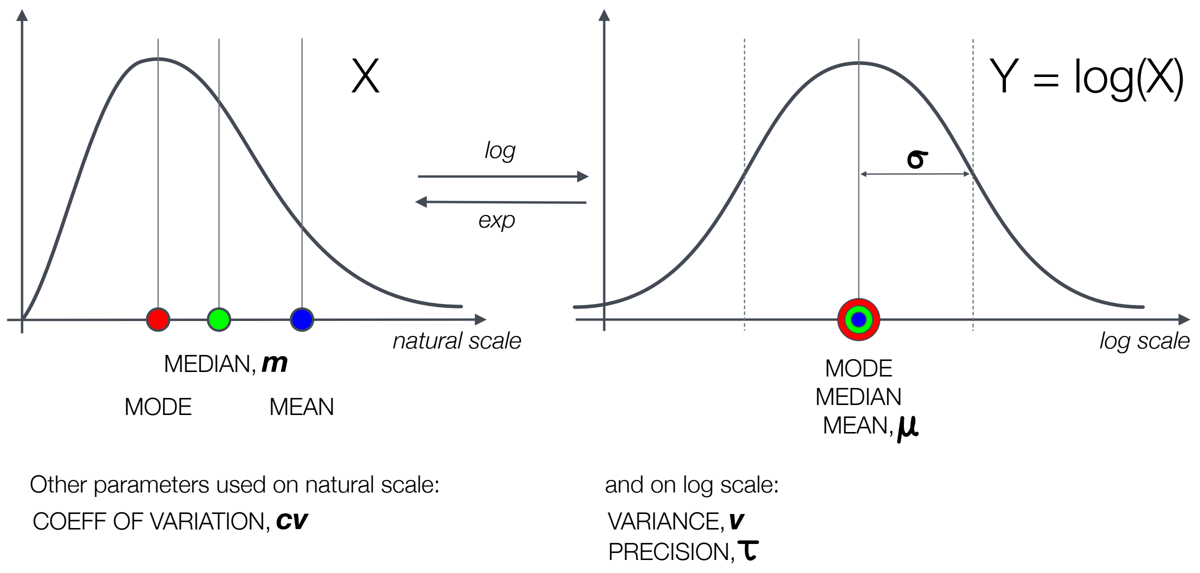


Figure 2.3: Schematic representation of the lognormaly distributed data on the natural (left) and logarithmic scale (right), see Figure 2.4 for real-life data example. **Bold** symbols stand for quantities commonly used to parameterise a log-normally distributed variable.

2.6.3 Log-normal distribution

The log-normal distribution is special in that not only different parameter sets exist but also because they are defined either on the natural or logarithmic scale. Interestingly, in one case the parameters are defined on two different scales, see Figure 2.3, for an overview.
Available parameterisations (also listed in Table 2.2 with indication about their coverage in target tools) are

- LogNormal1 $(\mu,\,\sigma)$ with *mean*, μ, and *standard deviation*, σ, both on the log-scale,
- LogNormal2 $(\mu,\,v)$ with *mean*, μ, and *variance*, v, both on the log-scale,
- LogNormal3 $(m,\,\sigma)$ with *median*, m, on the natural scale and *standard deviation*, σ, on the log-scale,
- LogNormal4 $(m,\,cv)$ with *median*, m, and *coefficient of variation*, cv, both on the natural scale,
- LogNormal5 $(\mu,\,\tau)$ with *mean*, μ, and *precision*, τ, both on the log-scale.

Re-parameterisation formulas

The recalculation between given parameterisations is error prone and should, whenever required, be taken over by the converters. The following equations might be useful when providing such translation support between target tools. For example when translating a model implemented for Monolix/NONMEM, which use either LN1 or LN2, with winBUGS as the target tool, which uses only LN5.

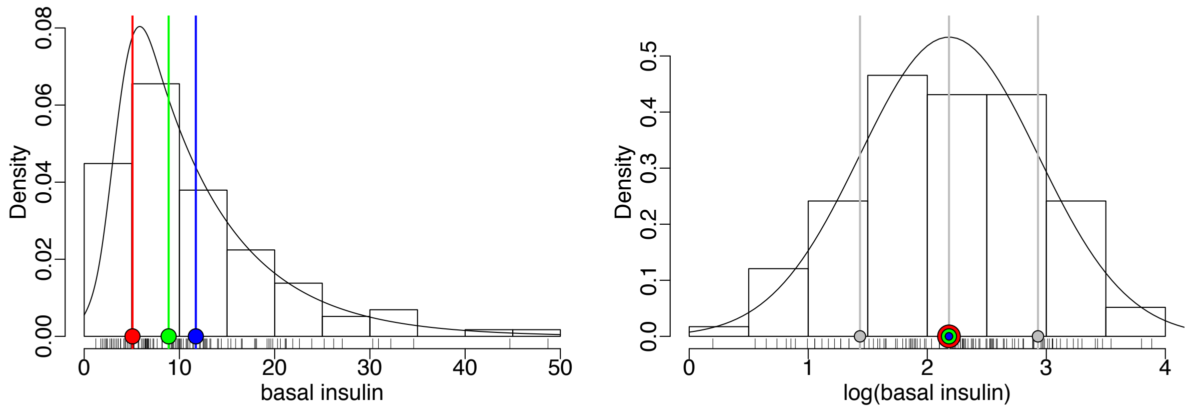


Figure 2.4: Representation of the lognormaly distributed basal insulin data in diabetic patients [?] on the natural scale (left) and on the logarithmic scale after *log*–transformation (right), colour code as in Figure 2.3. The density estimation for the data on the natural scale and its plotting was performed using the R package *logspline* [?].

- $\mathbf{LN1}(\mu,\,\sigma)\;\rightarrow\;\mathbf{LN2}(\mu,\,v)\;:\;\mu\rightarrow\mu;\quad\sigma\rightarrow v=\sigma^2$
  $\mathbf{LN2}(\mu,\,v)\;\rightarrow\;\mathbf{LN1}(\mu,\,\sigma)\;:\;\mu\rightarrow\mu;\quad v\rightarrow\sigma=\sqrt{v}$
- $\mathbf{LN1}(\mu,\,\sigma)\;\rightarrow\;\mathbf{LN3}(m,\,\sigma)\;:\;\mu\rightarrow m=\exp(\mu);\quad\sigma\rightarrow\sigma$
  $\mathbf{LN3}(m,\,\sigma)\;\rightarrow\;\mathbf{LN1}(\mu,\,\sigma)\;:\;m\rightarrow\mu=\log(m);\quad\sigma\rightarrow\sigma$
- $\mathbf{LN1}(\mu,\,\sigma)\;\rightarrow\;\mathbf{LN4}(m,\,cv)\;:\;\mu\rightarrow m=\exp(\mu);\quad\sigma\rightarrow cv=\sqrt{\exp(\sigma^2)-1}$
  $\mathbf{LN4}(m,\,cv)\;\rightarrow\;\mathbf{LN1}(\mu,\,\sigma)\;:\;m\rightarrow\mu=\log(m);\quad cv\rightarrow\sigma=\sqrt{\log(cv^2+1)}$
- $\mathbf{LN1}(\mu,\,\sigma)\;\rightarrow\;\mathbf{LN5}(\mu,\,\tau)\;:\;\mu\rightarrow\mu;\quad\sigma\rightarrow\tau=1/\sigma^2$
  $\mathbf{LN5}(\mu,\,\tau)\;\rightarrow\;\mathbf{LN1}(\mu,\,\sigma)\;:\;\mu\rightarrow\mu;\quad\tau\rightarrow\sigma=1/\sqrt{\tau}$
- $\mathbf{LN2}(\mu,\,v)\;\rightarrow\;\mathbf{LN3}(m,\,\sigma)\;:\;\mu\rightarrow m=\exp(\mu);\quad v\rightarrow\sigma=\sqrt{v}$
  $\mathbf{LN3}(m,\,\sigma)\;\rightarrow\;\mathbf{LN2}(\mu,\,v)\;:\;m\rightarrow\mu=\log(m);\quad\sigma\rightarrow v=\sigma^2$
- $\mathbf{LN2}(\mu,\,v)\;\rightarrow\;\mathbf{LN4}(m,\,cv)\;:\;\mu\rightarrow m=\exp(\mu);\quad v\rightarrow cv=\sqrt{\exp(v)-1}$
  $\mathbf{LN4}(m,\,cv)\;\rightarrow\;\mathbf{LN2}(\mu,\,v)\;:\;m\rightarrow\mu=\log(m);\quad cv\rightarrow v=\log(cv^2+1)$
- $\mathbf{LN2}(\mu,\,v)\;\rightarrow\;\mathbf{LN5}(\mu,\,\tau)\;:\;\mu\rightarrow\mu;\quad v\rightarrow\tau=1/v$
  $\mathbf{LN5}(\mu,\,\tau)\;\rightarrow\;\mathbf{LN2}(\mu,\,v)\;:\;\mu\rightarrow\mu;\quad\tau\rightarrow v=1/\tau$
- $\mathbf{LN3}(m,\,\sigma)\;\rightarrow\;\mathbf{LN4}(m,\,cv)\;:\;m\rightarrow m;\quad\sigma\rightarrow cv=\sqrt{\exp(\sigma^2)-1}$
  $\mathbf{LN4}(m,\,cv)\;\rightarrow\;\mathbf{LN3}(m,\,\sigma)\;:\;m\rightarrow m;\quad cv\rightarrow\sigma=\sqrt{\log(cv^2+1)}$
- $\mathbf{LN3}(m,\,\sigma)\;\rightarrow\;\mathbf{LN5}(\mu,\,\tau)\;:\;m\rightarrow\mu=\log(m);\quad\sigma\rightarrow\tau=1/\sigma^2$
  $\mathbf{LN5}(\mu,\,\tau)\;\rightarrow\;\mathbf{LN3}(m,\,\sigma)\;:\;\mu\rightarrow m=\exp(\mu);\quad\tau\rightarrow\sigma=1/\sqrt{\tau}$
- $\mathbf{LN4}(m,\,cv)\;\rightarrow\;\mathbf{LN5}(\mu,\,\tau)\;:\;m\rightarrow\mu=\log(m);\quad cv\rightarrow\tau=1/\log(cv^2+1)$
  $\mathbf{LN5}(\mu,\,\tau)\;\rightarrow\;\mathbf{LN4}(m,\,cv)\;:\;\mu\rightarrow m;\quad\tau\rightarrow cv=\sqrt{\exp(1/\tau)-1}$

The proof of the majority of the formulas is straightforward taking into account the definition of the parameters in question. The relationship between σ or τ (on the log scale) and cv (on the natural scale), essential for the re-calculation formulas involving LN4 parameterisation, is a bit more tricky to see. The proof starts with the known relationships for the mean, *mean*, and variance, *var*, on the natural scale, collected in table 2.1. Then the square of the coefficient of variation, cv, on the natural scale reads

$$cv^2=\frac{var}{mean^2}=\frac{(e^{\sigma^2}-1)\,e^{2\mu+\sigma^2}}{e^{(\mu+1/2\sigma^2)^2}}=(e^{\sigma^2}-1)\Leftrightarrow cv=\sqrt{e^{\sigma^2}-1}\quad\&\quad\sigma=\sqrt{\log(cv^2+1)}.$$

Log-normal distribution on the natural scale (NS)	Quantity	Normal distribution on the log-transformed scale (LS)
$LN1 : P(x; \boldsymbol{\mu}, \boldsymbol{\sigma}) = \frac{1}{x\sigma\sqrt{2\pi}} \exp\left[\frac{-(\log x - \mu)^2}{2\sigma^2}\right]$		
$e^{\mu + \frac{1}{2}\sigma^2}$	Mean	$\boldsymbol{\mu}$
$e^{2\mu+\sigma^2}[e^{\sigma^2} - 1]$	Variance	σ^2
$e^{\mu+\frac{1}{2}\sigma^2}\sqrt{e^{\sigma^2} - 1}$	Standard deviation	$\boldsymbol{\sigma}$
$e^{\mu-\sigma^2}$	Mode	μ
e^{μ}	Median	μ
$\sqrt{e^{\sigma^2} - 1}$	Coefficient of variation	σ/μ
$LN2 : P(x; \boldsymbol{\mu}, \boldsymbol{v}) = \frac{1}{x\sqrt{v}\sqrt{2\pi}} \exp\left[\frac{-(\log x - \mu)^2}{2v}\right]$		
$e^{\mu+\frac{1}{2}v}$	Mean	$\boldsymbol{\mu}$
$e^{2\mu+v}[e^{v} - 1]$	Variance	$\boldsymbol{v}$
$e^{\mu+\frac{1}{2}v}\sqrt{e^{v} - 1}$	Standard deviation	$\sqrt{v}$
$e^{\mu-v}$	Mode	μ
e^{μ}	Median	μ
$\sqrt{e^{v} - 1}$	Coefficient of variation	$\sqrt{v}/\mu$
$LN3 : P(x; \boldsymbol{m}, \boldsymbol{\sigma}) = \frac{1}{x\sigma\sqrt{2\pi}} \exp\left[\frac{-[\log(x/m)]^2}{2\sigma^2}\right]$		
$m\, e^{\frac{1}{2}\sigma^2}$	Mean	$\log(m)$
$m^2 e^{\sigma^2}[e^{\sigma^2} - 1]$	Variance	σ^2
$m\sqrt{e^{\sigma^2}(e^{\sigma^2} - 1)}$	Standard deviation	$\boldsymbol{\sigma}$
m/e^{σ^2}	Mode	$\log(m)$
$\boldsymbol{m}$	Median	$\log(m)$
$\sqrt{e^{\sigma^2} - 1}$	Coefficient of variation	$\sigma/\log(m)$
$LN4 : P(x; \boldsymbol{m}, \boldsymbol{cv}) = \frac{1}{x\sqrt{\log(cv^2+1)}\sqrt{2\pi}} \exp\left[\frac{-[\log(x/m)]^2}{2\log(cv^2+1)}\right]$		
$m\sqrt{cv^2 + 1}$	Mean	$\log(m)$
$m^2\,(cv^2 + 1)\,cv^2$	Variance	$\log(cv^2 + 1)$
$m\,cv\sqrt{(cv^2 + 1)}$	Standard deviation	$\sqrt{\log(cv^2 + 1)}$
$m/(cv^2 + 1)$	Mode	$\log(m)$
$\boldsymbol{m}$	Median	$\log(m)$
$\boldsymbol{cv}$	Coefficient of variation	$\sqrt{\log(cv^2 + 1)}/\log(m)$
$LN5 : P(x; \boldsymbol{\mu}, \boldsymbol{\tau}) = \sqrt{\frac{\tau}{2\pi}}\frac{1}{x} \exp\left[-\frac{\tau}{2}(\log x - \mu)^2\right]$		
$e^{\mu+\frac{1}{2\tau}}$	Mean	$\boldsymbol{\mu}$
$e^{2\mu+\frac{1}{\tau}}[e^{\frac{1}{\tau}} - 1]$	Variance	$1/\tau$
$e^{\mu+\frac{1}{2}\frac{1}{\tau}}\sqrt{e^{\frac{1}{\tau}} - 1}$	Standard deviation	$\sqrt{1/\tau}$
$e^{\mu-\frac{1}{\tau}}$	Mode	μ
e^{μ}	Median	μ
$\sqrt{e^{\frac{1}{\tau}} - 1}$	Coefficient of variation	$\sqrt{1/\tau}/\mu$
$1/\big(e^{2\mu+\frac{1}{\tau}}[e^{\frac{1}{\tau}} - 1]\big)$	Precision	$\boldsymbol{\tau}$

Table 2.1: The available parameterisations for the log-normal distribution and their characterising quantities as functions of the respective parameters. With $\boldsymbol{m}$ – median (NS), $\boldsymbol{cv}$ – coefficient of variation (NS), $\boldsymbol{\mu}$ – mean (LS), $\boldsymbol{\sigma}$ – standard deviation (LS), $\boldsymbol{v}$ – variance (LS), $\boldsymbol{\tau}$ – precision (LS). See Figure 2.3 and section 2.6.3 for the meaning of the symbols.

20 Small print: The re-parameterisation formulas, page 10, and expressions in this table are partially from literature, partially self calculated (by MJS), use them on your own risk.

ProbOnto 0.2	Parameters	**UncertML** 3.0	**WinBUGS** 1.4	**Monolix** 4.3	**NONMEM** 7.3
		Discrete Univariate			
Bernoulli	p	y	y	–	–
Binomial	$n,\ p$	y	y	–	–
Categorical ordered	$p_1, \dots, p_k$	y	y	–	–
Categorical unordered	$p_1, \dots, p_k$	y	y	–	–
Generalized Poisson	λ, δ	–	–	–	–
Geometric	p	y	–	–	–
Hypergeometric	$N,\ K,\ n$	y	–	–	–
Negative Binomial 1	$r,\ p$	y	y	–	–
Negative Binomial 2	λ, τ	–	–	–	–
Poisson	λ	y	y	–	–
Uniform 1 (discrete)	$a,\ b$	–	–	–	–
Uniform 2 (discrete)	$0,\ n$	–	–	–	–
Zero-inflated Poisson	λ, π	–	–	–	–
		Continuous Univariate			
Beta	α, β	y	y	y	–
Cauchy	x_0, γ	y	–	–	–
Chi-squared	k	y	y	y	–
Exponential	λ	y	y	y	–
F (aka Fisher-Snedecor)	d_1, d_2	y	–	y	–
Gamma	k, θ	y	y	y	–
Generalized Gamma	$a,\ d,\ p$	y	–	–	–
Gompertz	η, b	–	y	–	–
Gumbel (aka Extreme Value)	μ, β	–	y	–	–
Inverse-Gamma	α, β	y	–	–	–
Laplace 1 (Double-exponential 1)	μ, b	y	–	–	–
Laplace 2 (Double-exponential 2)	μ, τ	–	y	–	–
Log-Normal 1	μ, σ	y	–	y	–
Log-Normal 2	$\mu,\ v$	y	–	y	–
Log-Normal 3	$m,\ \sigma$	–	–	–	–
Log-Normal 4	$m,\ cv$	–	–	–	–
Log-Normal 5	$\mu,\ \tau$	–	y	–	–
Logistic	μ, s	y	y	–	–
Normal 1	μ, σ	y	–	y	y
Normal 2	μ, v	y	–	y	–
Normal 3	μ, τ	–	y	–	–
Normal-inverse-gamma	$\mu, \lambda, \alpha, \beta$	y	–	–	–
Pareto	x_m, α	y	y	–	–
Rayleigh	σ	–	–	y	–
Standard Normal	$\mu = 0, \sigma = 1$	y	–	y	y
Student's T	ν	y	y	y	–
Standard Uniform	$a = 0, b = 1$	–	–	y	y
Uniform	$a,\ b$	y	y	y	–
Weibull 1	λ, k	y	–	y	–
Weibull 2	λ, v	–	y	–	–

Table 2.2: Univariate distributions supported in ProbOnto. See Appendix **??** for the detailed description.

Distribution	**Parameters**			
Code name	Symbol	Code name	Symbol	Code name
Discrete Univariate				
`Bernoulli`	p	`probability`	–	–
`Binomial`	n	`numberOfFailures`	p	`probability`
`CategoricalOrdered`	$p_1, \ldots, p_k$	`categoryProb`	–	–
`CategoricalUnordered`	$p_1, \ldots, p_k$	`categoryProb`	–	–
`GeneralizedPoisson`	λ	`rate`	δ	`dispersion`
`Geometric`	p	`probability`	–	–
`Hypergeometric`	N	`populationSize`	K	`numberOfTrials`
			n	`numberOfSuccesses`
`NegativeBinomial1`	r	`numberOfFailures`	p	`probability`
`NegativeBinomial2`	λ	`rate`	τ	`overdispersion`
`Poisson`	λ	`rate`	–	–
`UniformDiscrete1`	a	`minimum`	b	`maximum`
`UniformDiscrete2`	0	`minimum`	n	`numberOfValues`
`ZeroInflatedPoisson`	λ	`rate`	π	`probabilityOfZero`
Continuous Univariate				
`Beta`	α	`alpha`	β	`beta`
`Cauchy`	x_0	`location`	γ	`scale`
`ChiSquared`	k	`degreesOfFreedom`	–	–
`Exponential`	λ	`rate`	–	–
`F`	d_1	`numerator`	d_2	`denumerator`
`Gamma`	k	`shape`	θ	`scale`
`GeneralizedGamma`	a	`scale`	d	`shape1`
			p	`shape2`
`Gompertz`	η	`shape`	b	`scale`
`Gumbel`	μ	`location`	β	`scale`
`InverseGamma`	α	`shape`	β	`scale`
`Laplace1`	μ	`location`	b	`scale`
`Laplace2`	μ	`location`	τ	`tau`
`LogNormal1`	μ	`meanLog`	σ	`stdevLog`
`LogNormal2`	μ	`meanLog`	v	`varLog`
`LogNormal3`	m	`median`	σ	`stdevLog`
`LogNormal4`	m	`median`	cv	`coefVar`
`LogNormal5`	μ	`meanLog`	τ	`precision`
`Logistic`	μ	`location`	s	`scale`
`Normal1`	μ	`mean`	σ	`stdev`
`Normal2`	μ	`mean`	v	`var`
`Normal3`	μ	`mean`	τ	`precision`
`NormalInverseGamma`	μ	`mean`	λ	`lambda`
	α	`alpha`	β	`beta`
`Pareto`	x_m	`scale`	α	`shape`
`Rayleigh`	σ	`scale`	–	–
`StandardNormal`	$\mu = 0$	`mean`	$\sigma = 1$	`stdev`
`StudentT`	ν	`degreesOfFreedom`	–	–
`StandardUniform`	$a = 0$	`minimum`	$b = 1$	`maximum`
`Uniform`	a	`minimum`	b	`maximum`
`Weibull1`	λ	`scale`	k	`shape`
`Weibull2`	λ	`lambda`	v	`shape`

Table 2.3: Code names for distribution and parameter names of the univariate distributions supported in ProbOnto.

ProbOnto 0.2	Parameters	**UncertML** 3.0	**WinBUGS** 1.4	**Monolix** 4.3	**NONMEM** 7.3
Discrete Multivariate					
Multinomial	$n, p_1, \ldots, p_k$	y	y	–	–
Continuous Multivariate					
Dirichlet	$\alpha_1, \ldots, \alpha_K$	y	y	–	–
Inverse-Wishart	Ψ, ν	–	–	–	y
Multivariate Normal 1	μ, Σ	y	–	–	–
Multivariate Normal 2	μ, T	–	y	–	–
Multivariate (Student) T 1	μ, Σ, ν	y	–	–	–
Multivariate (Student) T 2	μ, T, k	–	y	–	–
Wishart 1	V, n	y	–	–	–
Wishart 2	R, k	–	y	–	–

Table 2.4: Multivariate distributions with overview of tool support. See the Appendix **??** for the detailed description of the most relevant distributions – full ProbOnto database will be released soon.

Distribution	**Parameters**			
Code name	Symbol	Code name	Symbol	Code name
Discrete Multivariate				
Multinomial	n	`numberOfTrials`	$p_1, \ldots, p_k$	`probabilityOfSuccess`
Continuous Multivariate				
`Dirichlet`	$\alpha_1, \ldots, \alpha_K$	`concentration`	–	–
`InverseWishart`	Ψ	`scaleMatrix`	ν	`degreesOfFreedom`
`MultivariateNormal1`	μ	`mean`	Σ	`covarianceMatrix`
`MultivariateNormal2`	μ	`mean`	T	`precisionMatrix`
`MultivariateStudentT1`	μ	`mean`	Σ	`covarianceMatrix`
			ν	`degreesOfFreedom`
`MultivariateStudentT2`	μ	`mean`	T	`precisionMatrix`
			k	`degreesOfFreedom`
`Wishart1`	V	`scaleMatrix`	n	`degreesOfFreedom`
`Wishart2`	R	`inverseScaleMatrix`	k	`degreesOfFreedom`

Table 2.5: Code names for the multivariate distributions and their parameters.

ProbOnto	Parameters		**UncertML**
0.2	Symbol	Code name	3.0
`MixtureDistribution`	$\pi_1, \ldots, \pi_k$	`weight`	y

Table 2.6: Mixture distribution - so far only for univariate distributions.

2.7 ProbOnto examples

2.7.1 Categorical data models

For categorical models we always first list in PharmML all categories

```
<ListOfCategories>
    <Category symbId="cat1"/>
    <Category symbId="cat2"/>
    <Category symbId="cat3"/>
</ListOfCategories>
```

which informs the user/target tool about the number and identifiers of the categories in question. The probabilities vector $p_i, i = 1, \ldots, k$ is the only parameter and the user has two options. One can declare either

- explicitly the probabilities for all k categories or
- the $k-1$ probabilities, with the last probability, p_k, known assuming $\Sigma_i p_i = 1$.

Note that the order of the categories in `<ListOfCategories>` and the `<Parameter>` (where the probability vector, p_i, is encoded) elements, must be preserved.

Example: Nominal categorical model

Consider the following *Observation model* for nominal categorical data, [**?**]:

- Type of observed variable – discrete/categorical
- Category variable: y
- Set of categories: $\{1, 2, 3\}$
- Probabilities for category '1' and '2'

$$p2 := P(y = 1) = a1/(a1 + a2 + a3)$$
$$p1 := P(y = 2) = a2/(a1 + a2 + a3)$$

The following code shows how to implement the model starting with the general information: parameters involved and equations for the probabilities

```
<CategoricalData ordered="no">
    <!-- can alternatively be defined as individual parameters with IIV etc.-->
    <PopulationParameter symbId="a1"/>
    <PopulationParameter symbId="a2"/>
    <PopulationParameter symbId="a3"/>

    <PopulationParameter symbId="p1">
        <ct:Assign>
            <math:Equation>
                <math:Binop op="divide">
                    <ct:SymbRef symbIdRef="a1"/>
                    <math:Binop op="plus">
                        <ct:SymbRef symbIdRef="a1"/>
                        <math:Binop op="plus">
                            <ct:SymbRef symbIdRef="a2"/>
                            <ct:SymbRef symbIdRef="a3"/>
                        </math:Binop>
                    </math:Binop>
                </math:Binop>
            </math:Equation>
        </ct:Assign>
    </PopulationParameter>
    <PopulationParameter symbId="p2">
        <ct:Assign>
            <math:Equation>
                <math:Binop op="divide">
                    <ct:SymbRef symbIdRef="a2"/>
                    <math:Binop op="plus">
                        <ct:SymbRef symbIdRef="a1"/>
                        <math:Binop op="plus">
                            <ct:SymbRef symbIdRef="a2"/>
                            <ct:SymbRef symbIdRef="a3"/>
                        </math:Binop>
                    </math:Binop>
                </math:Binop>
            </math:Equation>
        </ct:Assign>
    </PopulationParameter>
```

then listing the categories and specifying the category variable

```
    <ListOfCategories>
        <Category symbId="cat1"/>
        <Category symbId="cat2"/>
        <Category symbId="cat3"/>
    </ListOfCategories>
    <CategoryVariable symbId="y"/>
```

and eventually defining the unordered categorical distribution, *CategoricalNonordered* in ProbOnto, with the parameter

- `categoryProb` – event probabilities vector, $p_1, \ldots, p_k$

with the number of categories, here equal $k = 3$, which can be inferred from the length of the `<ListOfCategories>`. The PMF reads then

```
    <PMF linkFunction="identity">
        <Distribution>
            <ProbOnto name="CategoricalNonordered">
                <!-- category probabilities - a vector of length 2 (=k-1) -->
                <Parameter name="categoryProb">
                    <ct:Assign>
                        <ct:Vector>
                            <ct:VectorElements>
                                <ct:SymbRef symbIdRef="p1"/>
                                <ct:SymbRef symbIdRef="p2"/>
                            </ct:VectorElements>
                        </ct:Vector>
                    </ct:Assign>
                </Parameter>
            </ProbOnto>
        </Distribution>
    </PMF>
</CategoricalData>
```

Given that there are k categories, by default the specification of $k-1$ probabilities is sufficient assuming $\Sigma p_i = 1$. Note that alternatively, the expressions for $p1$ and $p2$ could be implemented directly as `<VectorElements>`.

2.7.2 Count data models

Zero-inflated Poisson – ZIP

ProbOnto simplifies the encoding of many discrete models significantly. So far unavailable distributions such as Generalized Poisson (GP), Zero-inflated Poisson (ZIP) and others frequently used in pharmacometrics, [?, ?], are now much more easy to encode. The following example illustrates that.

The essential elements of the model is the following PMF

$$\begin{cases} \log(P(Y=k)) = \log(1-p0) - \lambda + k\log(\lambda) - \text{factln}(k) & \text{if } k > 0 \\ \log(P(Y=k)) = \log(p0 + (1-p0)\exp(-\lambda)) & \text{otherwise} \end{cases}$$

and the definition of model parameters, the Poisson intensity, λ, and the probability of extra zeros, $p0$. In the case of explicitly encoded PMF the model becomes lengthly. This comes always with a risk of encoding bugs/typos.

Explicitly encoded ZIP model The following explicit encoding of the Zero-inflated Poisson model was the only possibility in PharmML ≤ 0.6. Although the use of the model specific parameter elements, here `<IntensityParameter>` and `<ZeroProbabilityParameter>` is not mandatory, the complex conditional definition of this model within the `<PMF>` element, is still required, and reads

```
<ObservationModel blkId="om1">
    <Discrete>
        <CountData>
            <CountVariable symbId="y"/>
            <NumberCounts symbId="k"/>

            <IntensityParameter symbId="Lambda">
                <ct:Assign>
                    <ct:SymbRef blkIdRef="pm1" symbIdRef="lambda"/>
                </ct:Assign>
            </IntensityParameter>

            <ZeroProbabilityParameter symbId="P0">
                <ct:Assign>
                    <ct:SymbRef blkIdRef="pm1" symbIdRef="p0"/>
                </ct:Assign>
            </ZeroProbabilityParameter>

            <PMF linkFunction="log">
```

```
                <math:LogicBinop op="eq">
                    <ct:SymbRef symbIdRef="y"/>
                    <ct:SymbRef symbIdRef="k"/>
                </math:LogicBinop>
                <ct:Assign>
                    <math:Equation>
                        <math:Piecewise>
                        <!-- !!! 50 lines of code skipped here !!! -->
                        <!-- for encoding of the conditional PMF: -->
                        <!-- if (k > 0): log(1-p0)-lambda+k*log(lambda)-factln(k) -->
                        <!-- else: log(p0+(1-p0)*exp(-lambda)) -->
                        </math:Piecewise>
                    </math:Equation>
                </ct:Assign>
            </PMF>
        </CountData>
    </Discrete>
</ObservationModel>
```

Note that the explicit encoding of PMF's remains as an option in PharmML for user-defined or other distributions not covered by ProbOnto.

ZIP model encode using ProbOnto In contrast to the first option, encoding of models supported by ProbOnto becomes very easy as only the code names for the distribution and its parameters need to be specified as the following code snippet shows

```
<ObservationModel blkId="om1A">
    <Discrete>
        <CountData>
            <CountVariable symbId="y"/>
            <NumberCounts symbId="k"/>

            <PMF linkFunction="log">
                <math:LogicBinop op="eq">
                    <ct:SymbRef symbIdRef="y"/>
                    <ct:SymbRef symbIdRef="k"/>
                </math:LogicBinop>
                <Distribution>
                    <ProbOnto name="ZeroInflatedPoisson">
                        <Parameter name="rate">
                            <ct:Assign>
                                <ct:SymbRef blkIdRef="pm1" symbIdRef="Lambda"/>
                            </ct:Assign>
                        </Parameter>
                        <Parameter name="probabilityOfZero">
                            <ct:Assign>
                                <ct:SymbRef blkIdRef="pm1" symbIdRef="P0"/>
                            </ct:Assign>
                        </Parameter>
                    </ProbOnto>
                </Distribution>
            </PMF>
        </CountData>
    </Discrete>
</ObservationModel>
```

Poisson with mixtures – PMIX

Rather then creating specific types for Poisson, or other models, with mixture distributions for individual observations (PMIX), described in the PharmML 0.6 spec, [?],

$$P(y_{ij} = k; \pi, \lambda_1, \lambda_2) \quad = \quad \pi \frac{e^{-\lambda_1}\lambda_1^k}{k!} + (1-\pi)\frac{e^{-\lambda_2}\lambda_2^k}{k!}$$

with $\lambda_1, \lambda_2 > 0$ and $\pi \in [0, 1]$, the generic *MixtureDistribution* can be used as the following code shows

```
        <PMF linkFunction="log">
            <Distribution>
                <ProbOnto name="MixtureDistribution">
                    <!-- mixing weight -->
                    <Parameter name="weight">
                        <ct:Assign>
                            <ct:SymbRef symbIdRef="pi1"/>
                        </ct:Assign>
                    </Parameter>
                    <!-- lambda1 - Poisson intensity -->
                    <MixtureComponent name="Poisson">
                        <Parameter name="rate">
                            <ct:Assign>
                                <ct:SymbRef symbIdRef="lambda1"/>
                            </ct:Assign>
                        </Parameter>
                    </MixtureComponent>
                    <!-- lambda2 - Poisson intensity -->
                    <MixtureComponent name="Poisson">
                        <Parameter name="rate">
                            <ct:Assign>
                                <ct:SymbRef symbIdRef="lambda2"/>
                            </ct:Assign>
                        </Parameter>
                    </MixtureComponent>
                </ProbOnto>
            </Distribution>
        </PMF>
```

The `<MixtureComponent>` elements hold the mixtures in question, here Poisson with λ_1 and Poisson λ_2, and `<Parameter>`, π, the mixture probability or *weight*. The solution has the advantage to be extendable to any number of mixing components, m, [?]. The parameter π becomes a vector, π_i with $\pi_i \in [0, 1], i = 1, \dots, m$, and can be encoded as such using the `<Vector>` element.

Discussion The use of a generic *MixtureDistribution* seems well justified in this case. However, because reference literature exists for general Mixed Poisson regression models, [?], the introduction of such specialised mixture distribution could be considered for ProbOnto in future as well.

2.7.3 Truncated distributions

Truncation bounds can be set using `<LowerTruncationBound>` and `<UpperTruncationBound>` elements which accept any expression, such as $X \sim \mathcal{N}(\mu, \sigma, lower = \mu - 1.96\,\sigma, upper = \mu + 1.96\,\sigma)$ which in PharmML reads

```
<IndividualParameter symbId="pTruncated">
    <Distribution>
        <ProbOnto name="Normal1">
            <Parameter name="mean">
                <ct:Assign>
                    <ct:SymbRef symbIdRef="mu"/>
                </ct:Assign>
            </Parameter>
            <Parameter name="stdev">
                <ct:Assign>
                    <ct:SymbRef symbIdRef="sigma"/>
                </ct:Assign>
            </Parameter>
            <LowerTruncationBound>
                <ct:Assign>
                    <math:Equation>
                        <math:Binop op="minus">
                            <ct:SymbRef symbIdRef="mu"/>
                            <math:Binop op="times">
                                <ct:Real>1.96</ct:Real>
                                <ct:SymbRef symbIdRef="sigma"/>
                            </math:Binop>
                        </math:Binop>
                    </math:Equation>
                </ct:Assign>
            </LowerTruncationBound>
            <UpperTruncationBound>
                <ct:Assign>
                    <math:Equation>
                        <math:Binop op="plus">
                            <ct:SymbRef symbIdRef="mu"/>
                            <math:Binop op="times">
                                <ct:Real>1.96</ct:Real>
                                <ct:SymbRef symbIdRef="sigma"/>
                            </math:Binop>
```

```
                                </math:Binop>
                            </math:Equation>
                        </ct:Assign>
                    </UpperTruncationBound>
                </ProbOnto>
            </Distribution>
        </IndividualParameter>
```

The following Figure **??** illustrates few examples of truncated normal distribution on the interval [-2.5,1]

2.8 Independency of ProbOnto

It is worth to stress that ProbOnto ontology and knowledge base are fully independent from PharmML. ProbOnto doesn't enforce a certain way to implement it from a tool designer which allows it to be used across the DDMoRe target tools, languages and beyond.