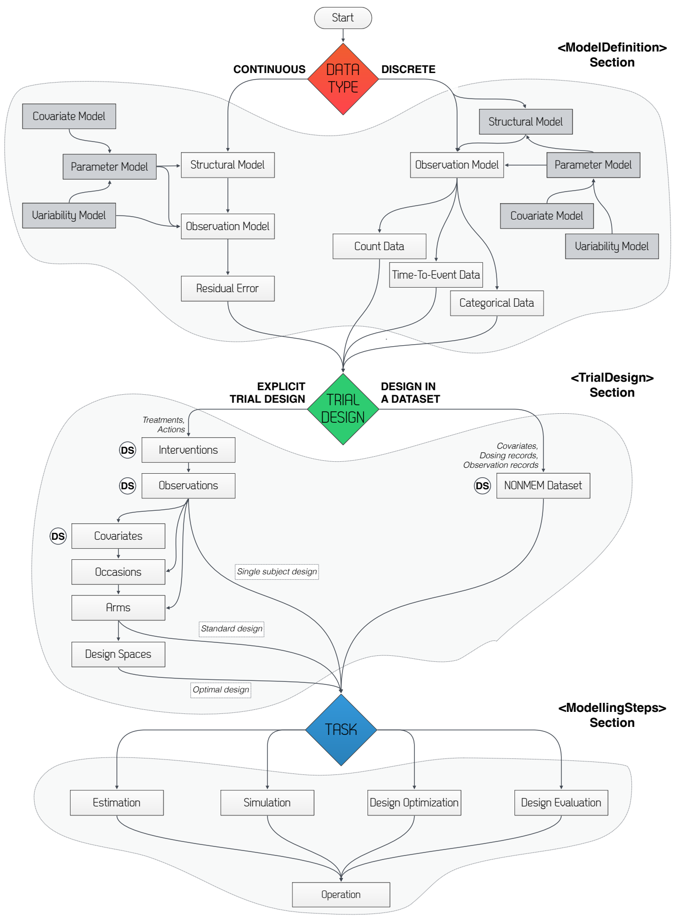
Internal Release

Extensions in PharmML 0.7

Authors:
Maciej J Swat
Pierre Grenon
Florent Yvon
Sarala Wimalaratne
Niels Rode Kristensen

with contributions from:
Emmanuelle Comets
Paolo Magni
Lorenzo Pasotti
Marc Lavielle



July 1, 2015

Contents

1 Overview 3
1.1 Major changes/extensions in version 0.7 . . . 3

2 *ProbOnto* – Ontology/Knowledge Base of Probability Distributions 5
2.1 Ontology versus Knowledge Base . . . 6
2.2 Limitations of the application of UncertML in PharmML . . . 6
2.3 ProbOnto Features . . . 7
2.3.1 Features under construction . . . 7
2.4 Working with ProbOnto . . . 8
2.4.1 New elements supporting ProbOnto . . . 8
2.5 Annotation of models with ProbOnto ontology . . . 8
2.5.1 Implementation in PharmML . . . 8
2.5.2 Annotation of PharmML . . . 9
2.5.3 Background Information is Contained in ProbOnto . . . 9
2.6 Alternative parameterisations – examples . . . 10
2.6.1 Negative binomial distribution . . . 10
2.6.2 Normal distribution . . . 10
2.6.3 Log-normal distribution . . . 11
2.7 ProbOnto examples . . . 16
2.7.1 Categorical data models . . . 16
2.7.2 Count data models . . . 18
2.7.3 Truncated distributions . . . 20
2.8 Independency of ProbOnto . . . 21

3 Parameter, Covariate and Observation Models 22
3.1 Distribution type – new model type . . . 22
3.2 Individual parameter representations – extended . . . 22
3.3 Population parameter – new . . . 25
3.4 Observation model – extended . . . 26
3.4.1 Discrete data models . . . 26
3.4.2 Continuous data models . . . 26

4 Hierarchical models and Bayesian Inference 28
4.1 winBUGS *Rats* example . . . 29
4.2 PK example . . . 30
4.2.1 Model definition . . . 31
4.2.2 Model implementation . . . 31
4.3 Hierarchical model example . . . 35
4.4 Additional examples implemented . . . 38

5 Design – redesigned 39
5.1 Design in PharmML ≤ 0.6 . . . 39
5.2 Modifications and extensions . . . 39
5.3 Selected features . . . 42
5.3.1 Actions – washout/reset options . . . 42
5.3.2 `<Interval>` element . . . 42
5.3.3 Using parameters in optimal design tasks . . . 43
5.3.4 Defining conditional covariate distribution . . . 44

5.3.5 Optimising covariates distribution . . . 44
5.4 Single subject design without arms . . . 46
5.5 Design examples implemented . . . 47
5.6 Differences compared to MDL . . . 47

6 Other changes 48
6.1 Box-Cox transformation . . . 48
6.1.1 Box-Cox in parameter model . . . 48
6.1.2 Box-Cox in observation model . . . 48
6.1.3 Box-Cox in covariate model . . . 49
6.2 Encoding of missing data . . . 49
6.2.1 Inline stored datasets . . . 49
6.2.2 External datasets . . . 50
6.3 Dataset headers . . . 50
6.4 Regressor support . . . 51
6.4.1 Support in $\leq$ 0.6 versions . . . 51
6.4.2 Support in 0.7 version . . . 52
6.5 Conditional distributions . . . 52
6.6 Minor changes/bug fixing . . . 53
6.7 Debugging with libPharmML . . . 54

A Selected distributions in ProbOnto knowledge base 55

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 ≤ 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 3	*Distribution*-type not available	NEW *Distribution*-type model can be used with either UncertML or ProbOnto
Individual parameter model, section 3.2	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 3.3	`<SimpleParameter>` used with no distribution support	NEW `<PopulationParameter>` element replaces `<SimpleParameter>` element
Observation model section 3.4.2	Gaussian and *Equation*-type	NEW *Distribution*-type

Covariate model, section 6.5	Transformation, interpolation and distribution of covariates supported	NEW Support for conditional distributions wrt covariates or design elements
section 6.6	not supported	`<CovariateModel>` has an additional option for creating new covariates out of exiting ones
Matrix/Vector operators, chapter 4	not supported basic types	NEW `<MatrixUniOp>` element with values *inverse, trace and transpose*
Transformation element, section 3.2 and 6.1	supported	new structure with attribute `type` to be assigned values such as *log*, *logit* etc. NEW *BoxCox*
Count data models, section 6.6		NEW `<NumberCounts>` element for variable k
	BAYESIAN INFERENCE & HIERARCHICAL MODELS	
Population parameter, chapter 4	not supported	NEW `<PopulationParameter>` with *Distribution*-type or *Equation*-type
	TRIAL DESIGN	
Structure, chapter 5	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 5.3.2	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 6.1	not available	`<Transformation>` with new value *BoxCox* for the `type` attribute and `<Parameter>` child element for *lambda* parameter
Missing data, section 6.2	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 6.3	not supported	NEW elements in dataset – `<Definition>`: `<Header>` with attributes `name`, `headerType`, `rowNumber` – `<Table>`: `<HeaderRow>` with attribute `order`
Regressors, section 6.4	supported when using datasets and lookup tables	NEW attribute `regressor` with values *yes/no* added to `<Variable>` element
SO column types, section 6.6	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.

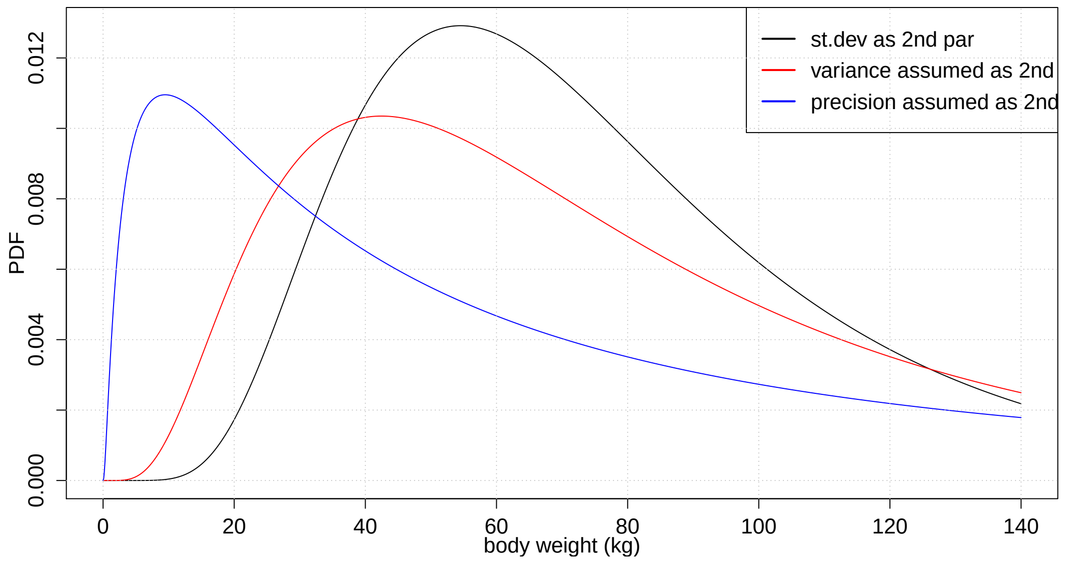


Figure 2.1: Illustration of possible model misspecification when using incorrect parameterisation. (1) The black curve corresponds to a log-normal distribution of body weight, $\mathcal{LN}(\mu = \log(70), \sigma = 0.5)$, the intended parameterisation. (2) Here it was mistakenly assumed that 0.5 corresponds 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, [23], 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, [14], 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 A 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 [7], Leemis et al. 2008 [12], Song & Chen 2011 [17], and Wolfram MathWorld [25].

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.

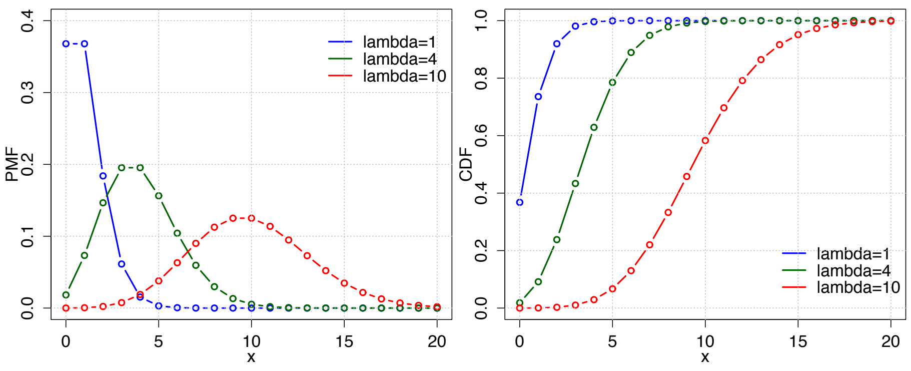


Figure 2.2: PMF and CDF of the Poisson distribution plotted using the R-code stored in ProbOnto.

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, [11]). 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, [15, 22]. 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.

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

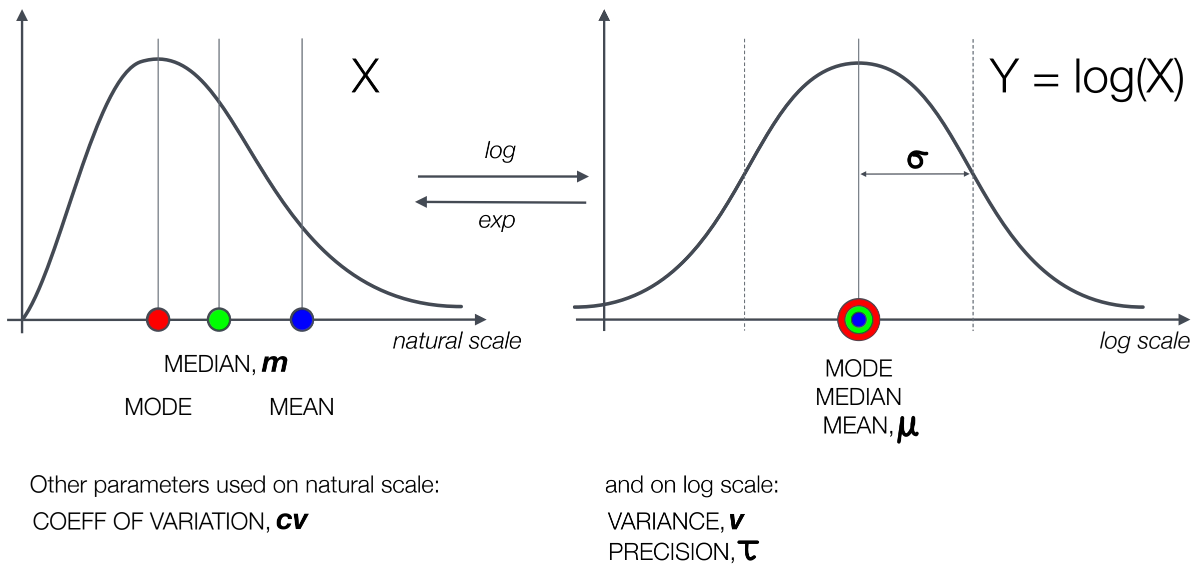


Figure 2.3: Schematic representation of the lognormaly distributed data on the natural (left) and logarithmic scale (right), see Figure 2.4 for real-life data example. **Bold** symbols stand for quantities commonly used to parameterise a log-normally distributed variable.

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.

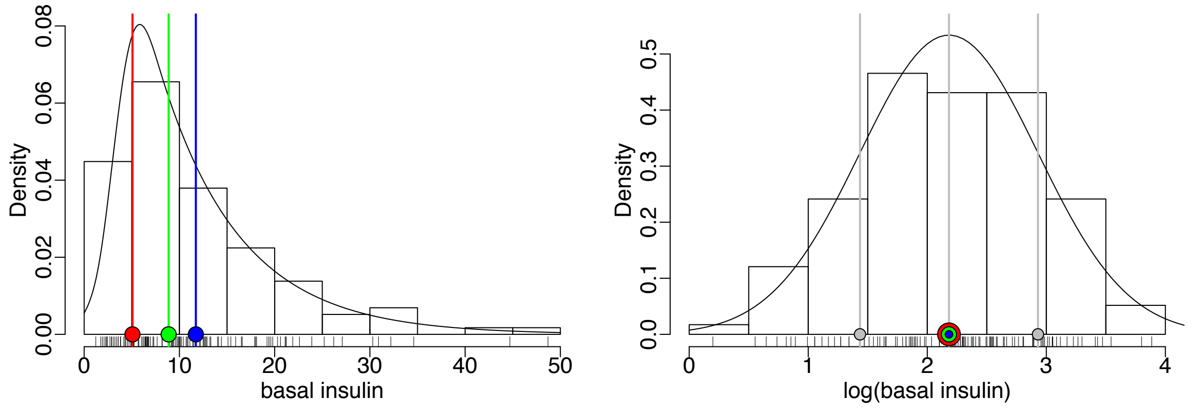


Figure 2.4: Representation of the lognormaly distributed basal insulin data in diabetic patients [16] on the natural scale (left) and on the logarithmic scale after *log*–transformation (right), colour code as in Figure 2.3. The density estimation for the data on the natural scale and its plotting was performed using the R package *logspline* [8].

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

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	$\log(m)$
$m^2e^{\sigma^2}[e^{\sigma^2}-1]$	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	$\sqrt{1/\tau}/\mu$
$1/\left(e^{2\mu+\frac{1}{\tau}}[e^{\frac{1}{\tau}}-1]\right)$	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.
Small print: The re-parameterisation formulas, page 11, 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 A 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`	$p_1, \dots, p_k$	`categoryProb`	–	–
`CategoricalUnordered`	$p_1, \dots, p_k$	`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 A 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, [20]:

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

$$
\begin{aligned}
p2 &:= P(y = 1) = a1/(a1 + a2 + a3) \\
p1 &:= P(y = 2) = a2/(a1 + a2 + a3)
\end{aligned}
$$

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, $p_1, \dots, p_k$

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, [15, 22], 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, [21],

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

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, [7]. 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, [24], 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 2.5 illustrates few examples of truncated normal distribution on the interval [-2.5,1]

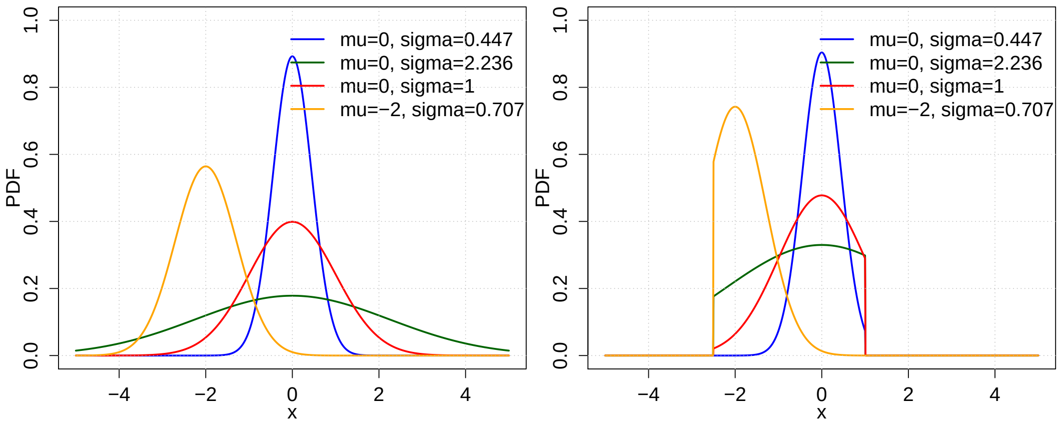


Figure 2.5: Truncated normal distribution, N1. The truncated density plots on the right hand side were performed using the *truncnorm* function of the *dtruncnorm* R-package.

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.

Chapter 3

Parameter, Covariate and Observation Models

3.1 Distribution type – new model type

The new model type follows the textbook notation for definition of the distribution of a random variable and can be applied for covariates, parameters, observations etc. It is required when defining e.g. a parameter model without the detour of using a random effect (sometimes called the *eta-free* notation).

- Distribution type

  $$h(X) \sim DistributionName(parameter1, parameter2, \dots)$$

  e.g.

  $$\log(X) \sim \mathcal{N}\big(\log(X_{typical}), \omega_X\big)$$

  with X standing for a covariate, parameter, observation etc. Examples will be given below.

This type can be encoded using both ProbOnto or UncertML although the latter option has limitations compared to ProbOnto, described in detail in section 2.2, in that it doesn't allow to encode arbitrary expressions as parameters, as required in the above example.

3.2 Individual parameter representations – extended

Following changes have been introduced to cover a broader class of parameter models

- `<GaussianModel>` element has been renamed to `<StructuredModel>` to account for its general purpose, beyond the normal distribution, an example follows below.

- `<PopulationParameter>` element has been renamed to `<PopulationValue>` for consistency with its meaning and use.

- `<Transformation>` element has been redesigned in order to account for Box-Cox transformation (see Section 6.1 for a description). Instead of

```
<StructuredModel>
    <Transformation>log</Transformation>
```

  now it reads

```
<StructuredModel>
    <Transformation type="log"/>
    ...
```

  with `type` to be assigned one of the values: {idenitity, log, logit, probit, BoxCox[1]}

- `<PopulationValue>` can be used without the parent element `<LinearCovariate>` if no covariate is taken into account. E.g. for the simplest case of a log-normally distributed parameter $\log(V) = \log(V_{pop}) + \eta_V$ the following completely describes the model and is shown in two equivalent versions, table 3.1.

[1]introduced in Section 6.1

NEW without `<LinearCovariate>` element	with `<LinearCovariate>` element
	`<IndividualParameter symbId="V">`
`<IndividualParameter symbId="V">`	`  <StructuredModel>`
`  <StructuredModel>`	`    <Transformation type="log"/>`
`    <Transformation type="log"/>`	`    <LinearCovariate>`
`    <PopulationValue>`	`      <PopulationValue>`
`      <ct:Assign>`	`        <ct:Assign>`
`        <ct:SymbRef symbIdRef="Vpop"/>`	`          <ct:SymbRef symbIdRef="Vpop"/>`
`      </ct:Assign>`	`        </ct:Assign>`
`    </PopulationValue>`	`      </PopulationValue>`
`    <RandomEffects>`	`    </LinearCovariate>`
`      <ct:SymbRef symbIdRef="eta_V"/>`	`    <RandomEffects>`
`    </RandomEffects>`	`      <ct:SymbRef symbIdRef="eta_V"/>`
`  </StructuredModel>`	`    </RandomEffects>`
`</IndividualParameter>`	`  </StructuredModel>`
	`</IndividualParameter>`

Table 3.1: Comparison of equivalent implementation of the basic parameter model, $\log(V) = \log(V_{pop}) + \eta_V$, without a covariate. The element `<PopulationValue>` doesn't have to be nested within `<LinearCovariate>` as was the case in ≤ 0.6.

The following representations types are available

Type I1 Structured (e.g. Gaussian) model

- A. Linear covariate model

$$h(\psi_i) = h(\psi_{pop}) + \beta c_i + \eta_i \quad \text{[Gaussian if } \eta_i \sim N(.,.)]$$

- B. General covariate model

$$h(\psi_i) = H(\beta, c_i) + \eta_i \quad \text{[Gaussian if } \eta_i \sim N(.,.)]$$

Type I2 Equation type

$$\psi_i = H(\beta, c_i, \eta_i)$$

Type I3 (NEW) Distribution type (i.e. eta-free notation)

$$h(\psi_i) \sim DistributionName(parameter1, parameter2, \dots)$$

Example 1. Individual model as used in the formulation of a hierarchical model example proposed in [10]

$$\log(\psi_i) \sim \mathcal{LN}\big(V_{pred}, \omega_V\big)$$

```xml
<IndividualParameter symbId="V">
    <LHSTransformation type="log"/>
    <ct:VariabilityReference>
        <ct:SymbRef blkIdRef="vm1" symbIdRef="indiv"/>
    </ct:VariabilityReference>
    <Distribution>
        <ProbOnto name="LogNormal1">
            <Parameter name="meanLog">
                <ct:Assign>
                    <ct:SymbRef symbIdRef="V_pred"/>
                </ct:Assign>
            </Parameter>
            <Parameter name="stdevLog">
                <ct:Assign>
                    <ct:SymbRef symbIdRef="omega_V"/>
                </ct:Assign>
            </Parameter>
        </ProbOnto>
    </Distribution>
</IndividualParameter>
```

Note 1. `<StructuredModel>` extends `<GaussianModel>` to be more general without loosing the ability to express the later – this updated element allows e.g. Student-T distributed parameters to be formulated in the additive form

$$\log(V_i) = \log(V_{pop}) + \beta c_i + \eta_i, \quad \text{with} \quad \eta_i \sim StudentT(0, \omega_V)$$

Note 2. New Distribution-type (Type 2B) allows using UncertML/ProbOnto to define a model without the random effect detour, e.g.

$$\log(V_i) \sim \mathcal{N}\big(\log(V_{pop}), \omega_V\big)$$

which is equivalent to

$$\log(V_i) = \log(V_{pop}) + \eta_i \quad \text{with} \quad \eta_i \sim \mathcal{N}(0, \omega_V)$$

Note 3. ProbOnto, compared to UncertML, provides additional support for expressions in parameters, e.g. $mean = \log(V_{pop})$ in the above equation.

Note 4. Type 2B comes with support for multiple levels of variability, Figure 3.1,

$$\log(V_{ik}) \sim \mathcal{N}\big(\log(V_{pop}), \underbrace{\omega_V}_{\substack{\text{IIV}\\ \text{level 0}}} + \underbrace{\kappa_V}_{\substack{\text{IOV}\\ \text{level 1}}}\big)$$

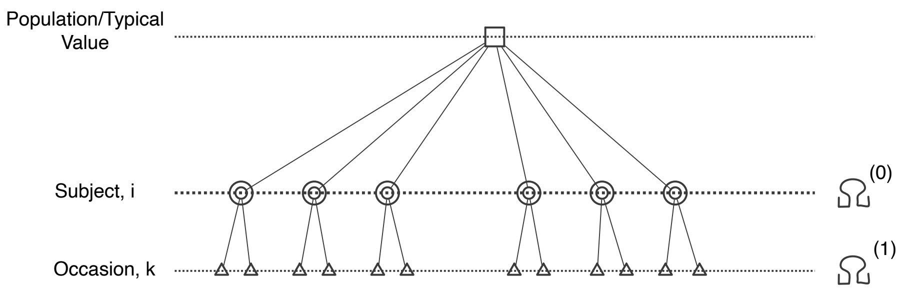


Figure 3.1: Model with IOV variability level.

as the following PharmML snippet shows

```
<IndividualParameter symbId="V">
    <ct:VariabilityReference>
        <ct:SymbRef symbIdRef="iov"/>
        <ct:RandomEffectMapping>
            <ct:SymbRef symbIdRef="kappa_V"/>
        </ct:RandomEffectMapping>
    </ct:VariabilityReference>
    <ct:VariabilityReference>
        <ct:SymbRef symbIdRef="subject"/>
        <ct:RandomEffectMapping>
            <ct:SymbRef symbIdRef="omega_V"/>
        </ct:RandomEffectMapping>
    </ct:VariabilityReference>
    <Distribution>
        <ProbOnto name="Normal1">
            <Parameter name="mean">
                <ct:Assign>
                    <math:Equation>
                        <math:Uniop op="log">
                            <ct:SymbRef symbIdRef="Vpop"/>
                        </math:Uniop>
                    </math:Equation>
                </ct:Assign>
            </Parameter>
            <Parameter name="stdev">
```

```
                        <ct:Assign>
                            <math:Equation>
                                <math:Binop op="plus">
                                    <ct:SymbRef symbIdRef="omega_V"/>
                                    <ct:SymbRef symbIdRef="kappa_V"/>
                                </math:Binop>
                            </math:Equation>
                        </ct:Assign>
                    </Parameter>
                </ProbOnto>
            </Distribution>
        </IndividualParameter>
```

The code above illustrates how

- expressions can be encoded using ProbOnto, here the $\log(Vpop)$ as the *mean* parameter of the normal distribution.
- higher variability levels – here the IOV – by mapping of the according variances to a particular level using the new `<RandomEffectMapping>` element. ω_V is mapped to IIV, κ_V is mapped to IOV. Note, that the same information could have been implemented using the `<StructuredModel>`, but additionally two random effects would have to be declared.

3.3 Population parameter – new

Population parameter comes with a flexible structure and can have an associated distribution, see Chapter 4 on Bayesian inference and hierarchical models. Two representations are available

Type P1 Equation type

$$\psi_{pop} = H(\beta_i, \eta_i, ...)$$

Type P2 Distribution type, i.e. eta-free notation

$$h(\psi_{pop}) \sim Distribution(parameter1, parameter2, ...)$$

Note 1. `<PopulationParameter>` replaces the `<SimpleParameter>` which became redundant as the latter was used so far to encode a population parameter.

Example 1. Using ProbOnto and Type P2 - from fourModels_hierarchical.xml, [10],

$$V_{pop} \sim \mathcal{LN}\big(log(Vs), gV\big)$$

is encoded in PharmML as

```
        <PopulationParameter symbId="V_pop">
            <ct:VariabilityReference>
                <ct:SymbRef blkIdRef="vm1" symbIdRef="pop"/>
            </ct:VariabilityReference>
            <Distribution>
                <ProbOnto name="LogNormal1">
                    <Parameter name="meanLog">
                        <ct:Assign>
                            <math:Equation>
                                <math:Uniop op="log">
                                    <ct:SymbRef symbIdRef="Vs"/>
                                </math:Uniop>
                            </math:Equation>
                        </ct:Assign>
                    </Parameter>
                    <Parameter name="stdevLog">
                        <ct:Assign>
                            <ct:SymbRef symbIdRef="gV"/>
                        </ct:Assign>
                    </Parameter>
                </ProbOnto>
            </Distribution>
        </PopulationParameter>
```

3.4 Observation model – extended

3.4.1 Discrete data models

The new features are

- Seriously simplified encoding of discrete data models. In PharmML versions ≤ 0.6 the implementation of explicit PMF was required for many models due to the lack of their coverage in UncertML.
- All common/relevant parametric distributions and/or their alternative parameterisations are supported by ProbOnto, as described in Chapter 2.
- Addition of new distributions is straightforward and will be done upon request.
- As an example of ProbOnto use, we discuss here a complete observation model for count data.
  - Type of observed variable – discrete/count
  - Model name: NegativeBinomial2
  - Count variable: y
  - Number of counts: k
  - Probability mass function

$$P(y_{ij} = k; \lambda, \tau) \quad = \quad \left[\frac{\Gamma\left(k + \frac{1}{\tau}\right)}{k! \times \Gamma\left(\frac{1}{\tau}\right)}\right] \times \left(\frac{1}{1 + \tau \times \lambda}\right)^{\frac{1}{\tau}} \times \left(\frac{\lambda}{\frac{1}{\tau} + \lambda}\right)^{k}$$

  - Link function: log
  - Dispersion parameter, τ
  - Constant rate parameter λ, the Poisson 'intensity': $\lambda(t_{ij}, \psi_{ij}) = \lambda_i$

and reads in PharmML as

```
<ObservationModel blkId="om2">
    <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>
                <ProbOnto name="NegativeBinomial2">
                    <Parameter name="rate">
                        <ct:Assign>
                            <ct:SymbRef blkIdRef="pm1" symbIdRef="lambda"/>
                        </ct:Assign>
                    </Parameter>
                    <Parameter name="overdispersion">
                        <ct:Assign>
                            <ct:SymbRef blkIdRef="pm1" symbIdRef="tau"/>
                        </ct:Assign>
                    </Parameter>
                </ProbOnto>
            </PMF>
        </CountData>
    </Discrete>
</ObservationModel>
```

with *lambda* and *tau* defined in the parameter model, `pm1`

3.4.2 Continuous data models

Available observation model representation has been extended and the available types are

Type O1 Structured model

$$u(y) = u(f) + g \times \epsilon \quad \text{[Gaussian if } \epsilon \sim \mathcal{N}(.,.)\text{]}$$

is still used with the `<Standard>` tag (but comes with extended interpretation and allows non-Gaussian residual errors (see the related discussion in section 3.2). Also the Box-Cox transformation can be applied in this case, see discussion in section 6.1.

Type O2 General model (equation type), unchanged compared to 0.6

$$h(y) = H(f, \xi, \epsilon)$$

Type O3 (NEW) Distribution type (ϵ-free notation)

$$u(y) \sim DistributionName(parameter1, parameter2, ...)$$

Type O1 representation	**Type O3** representation
$Y = C + SD\_ADD \times \epsilon \quad \text{with} \quad \epsilon \sim \mathcal{N}(0,1)$	$Y \sim \mathcal{N}(C, SD\_ADD)$
see code below (Type O1)	see code below (Type O3)

```
<Standard symbId="Y">
   <Output>
     <!-- blkIdRef="sm1" -->
     <ct:SymbRef symbIdRef="C"/>
   </Output>
   <ErrorModel>
     <!-- blkIdRef="pm1" -->
     <ct:Assign>
        <ct:SymbRef symbIdRef="SD_ADD"/>
     </ct:Assign>
   </ErrorModel>
   <ResidualError>
      <ct:SymbRef symbIdRef="eps"/>
   </ResidualError>
</Standard>
<!-- declaration of 'eps' omitted -->
```

```
<General symbId="Y">
   <ct:VariabilityReference>
      <ct:SymbRef symbIdRef="residual"/>
   </ct:VariabilityReference>
   <Distribution>
      <ProbOnto name="Normal1">
         <Parameter name="mean">
            <ct:Assign>
               <ct:SymbRef symbIdRef="C"/>
            </ct:Assign>
         </Parameter>
        <Parameter name="stdev">
            <ct:Assign>
               <ct:SymbRef symbIdRef="SD_ADD"/>
            </ct:Assign>
         </Parameter>
      </ProbOnto>
   </Distribution>
</General>
```

Table 3.2: Comparison of *Type O1* and *Type O3* (ϵ-free) representations. For better code readability the `blkIdRef` attributes have been removed.

Chapter 4

Hierarchical models and Bayesian Inference

While the detailed Bayesian support is described in the according MDL specification proposal, [3], we describe here a few PharmML elements, either entirely new, redefined or as used in the context of hierarchical models and Bayesian inference.

- The `<PopulationParameter>` element provides, as indicated in section 3.3, the support required for such models, i.e. distribution notation.
- ProbOnto provides missing distributions, such as *Inverse-Wishart* and other features required.
- Prior related variability level of any parameter extends its variability structure used so far, see Figure 4.1, and consequently the extended parameter related variability model for this case reads

```
<VariabilityModel blkId="model" type="parameterVariability">
    <Level symbId="pop"/>
    <Level symbId="indiv">
        <ParentLevel>
            <ct:SymbRef symbIdRef="pop"/>
        </ParentLevel>
    </Level>
</VariabilityModel>
```

- Few basic matrix operators, available as attributes of the new `<MatrixUniOp>` element – such as *inverse*, *transpose*, *trace* has been introduced to allow for encoding of related model features, see examples in the next section.

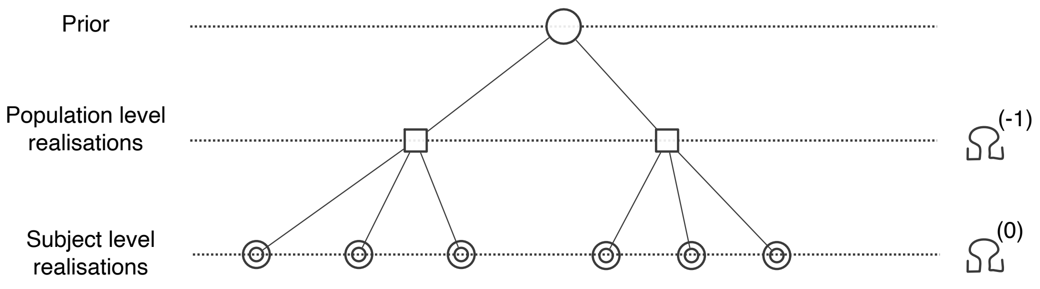


Figure 4.1: Basic variability structure with prior, population and subject levels.

In section 3.3 we have introduced the notation for population parameters with associated prior distribution. In the following we discuss three complete examples, two estimation examples as formulated typically in winBUGS and a generic hierarchical model, which can be used for simulation or estimation tasks.

4.1 winBUGS *Rats* example

As an independent example, i.e. not directly related with pharmacometrics, we tested the schema with a use case from the winBUGS example collection, [13].
The model

$$
\begin{aligned}
Y_{ij} &\sim \mathcal{N}(\alpha_i + \beta_i(x_j - x_{bar}), \tau_c)\\
\alpha_i &\sim \mathcal{N}(\alpha_c, \tau_\alpha)\\
\beta_i &\sim \mathcal{N}(\beta_c, \tau_\beta)
\end{aligned}
$$

reads in winBUGS code

```
for( i in 1 : N ) {
        for( j in 1 : T ) {
                Y[i , j] ~ dnorm(mu[i , j],tau.c)
                mu[i , j] <- alpha[i] + beta[i] * (x[j] - xbar)
        }
        alpha[i] ~ dnorm(alpha.c,alpha.tau)
        beta[i] ~ dnorm(beta.c,beta.tau)
}
tau.c~dgamma(0.001,0.001)
alpha.c ~ dnorm(0.0,1.0E-6)
alpha.tau ~ dgamma(0.001,0.001)
beta.c ~ dnorm(0.0,1.0E-6)
beta.tau ~ dgamma(0.001,0.001)
```

Observation model is encoded using the previously introduced distribution type, *O3*, section 3.4.2

```
<ObservationModel blkId="om1">
    <ContinuousData>
        <General symbId="Y">
            <ct:VariabilityReference>
                <ct:SymbRef blkIdRef="vm2" symbIdRef="residual"/>
            </ct:VariabilityReference>
            <Distribution>
                <ProbOnto name="Normal3">
                    <Parameter name="mean">
                        <ct:Assign>
                            <ct:SymbRef blkIdRef="pm1" symbIdRef="mu"/>
                        </ct:Assign>
                    </Parameter>
                    <Parameter name="precision">
                        <ct:Assign>
                            <ct:SymbRef blkIdRef="pm1" symbIdRef="tau.c"/>
                        </ct:Assign>
                    </Parameter>
                </ProbOnto>
            </Distribution>
        </General>
    </ContinuousData>
</ObservationModel>
```

with the mean, μ, and another two individual parameters, α and β encoded next.

Parameter model We show only implementation for μ and α (β is encoded similarily). μ is an expression but it could have been implemented directly in the observation model declaration making use of ProbOnto's capability to assign any expressions to parameters.

```
<IndividualParameter symbId="mu">
    <ct:Assign>
        <Equation xmlns="http://www.pharmml.org/pharmml/0.6/Maths">
            <Binop op="plus">
                <ct:SymbRef symbIdRef="alpha"/>
                <Binop op="times">
                    <ct:SymbRef symbIdRef="beta"/>
                    <Binop op="minus">
                        <ct:SymbRef symbIdRef="x"/>
                        <ct:SymbRef symbIdRef="xbar"/>
                    </Binop>
                </Binop>
            </Binop>
            </Equation>
        </ct:Assign>
    </IndividualParameter>
```

.

```
    <IndividualParameter symbId="alpha">
        <ct:VariabilityReference>
            <ct:SymbRef blkIdRef="vm1" symbIdRef="subject"/>
        </ct:VariabilityReference>
        <Distribution>
            <ProbOnto name="Normal3">
                <Parameter name="mean">
                    <ct:Assign>
                        <ct:SymbRef symbIdRef="alpha.c"/>
                    </ct:Assign>
                </Parameter>
                <Parameter name="precision">
                    <ct:Assign>
                        <ct:SymbRef symbIdRef="alpha.tau"/>
                    </ct:Assign>
                </Parameter>
            </ProbOnto>
        </Distribution>
    </IndividualParameter>
```

Priors All remaining population parameters τ_c, α_c, α_τ, β_c, β_τ are given *non-informative* priors (using Normal or Gamma distribution) and for simplicity we show only the first two.

```
    <PopulationParameter symbId="tau.c">
        <ct:VariabilityReference>
            <ct:SymbRef blkIdRef="vm1" symbIdRef="pop"/>
        </ct:VariabilityReference>
        <Distribution>
            <ProbOnto name="Gamma">
                <Parameter name="shape">
                    <ct:Assign>
                        <ct:Real>0.001</ct:Real>
                    </ct:Assign>
                </Parameter>
                <Parameter name="scale">
                    <ct:Assign>
                        <ct:Real>0.001</ct:Real>
                    </ct:Assign>
                </Parameter>
            </ProbOnto>
        </Distribution>
    </PopulationParameter>

    <PopulationParameter symbId="alpha.c">
        <ct:VariabilityReference>
            <ct:SymbRef blkIdRef="vm1" symbIdRef="pop"/>
        </ct:VariabilityReference>
        <Distribution>
            <ProbOnto name="Normal3">
                <Parameter name="mean">
                    <ct:Assign>
                        <ct:Real>0.0</ct:Real>
                    </ct:Assign>
                </Parameter>
                <Parameter name="precision">
                    <ct:Assign>
                        <ct:Real>1.0E-6</ct:Real>
                    </ct:Assign>
                </Parameter>
            </ProbOnto>
        </Distribution>
    </PopulationParameter>
```

4.2 PK example

This example, based on the section 3.3.2 in [3], comes with correlated parameters V, k and Vpop, kpop. Here the original model description is used: *In this case, the model parameters are not independent random variables* *at both the individual and population level.* It will be useful to present few new features such as

- multivariate distributions with parameters, e.g. $\mathcal{MVN}$
  - mean as vector
  - covariance matrix
- matrix operators, e.g. inverse of a matrix
- new distribution type, the gamma distribution $\Gamma(,)$

4.2.1 Model definition

Parameter model

$$\begin{pmatrix} \log(V_j) \\ \log(k_j) \end{pmatrix} \sim \mathcal{MVN}\left(\begin{pmatrix} \log(V_{pop}) \\ \log(k_{pop}) \end{pmatrix}, \Omega_P\right)$$
$$\log(\tau_{e_j}) \sim \mathcal{N}\big(\log(\tau_{pop}), \omega_\tau^2\big)$$

Prior distributions

$$\begin{pmatrix} \log(V_{pop}) \\ \log(k_{pop}) \end{pmatrix} \sim \mathcal{MVN}\left(\begin{pmatrix} \log(\mu_{V_{pop}}) \\ \log(\mu_{k_{pop}}) \end{pmatrix}, \Sigma_{P_{pop}}\right)$$
$$\Omega_P^{-1} \sim \mathcal{W}(R^{-1}, \rho)$$
$$\omega_\tau^{-2} \sim \Gamma(a_{\omega_\tau^2}, b_{\omega_\tau^2})$$
$$\tau_{pop} \sim \Gamma(a_{\tau_{pop}}, b_{\tau_{pop}})$$

Structural and observation models

$$z_{ij} \sim \mathcal{N}(c_{ij}, \sigma_{e_j}^2)$$
$$c_{ij} = D/V_j e^{-k_j t_i}$$

4.2.2 Model implementation

We will skip the structural and observation models as they don't contribute any new insights in the discussion relevant to this chapter. Otherwise, we follow the winBUGS code as provided, with few following changes

- removed `<RandomVariable>` declaration for `eps` because of the usage of distribution type observation model as defined in the model. Accordingly `<Standard>` has been replaced by `<General>` with `<Distribution>` tags (see *Type O1*, *Type O3* discussion in table 3.2).

Parameter model The individual parameters V and k have to be jointly extracted from a multivariate distribution. In particular, log(V) and log(k) are the elements of a vector, which is distributed as a multivariate normal with specific population mean and covariance matrix describing the inter-individual variability.

In the WinBUGS code, the multivariate normal requires the vector mean and the inverse of covariance matrix as arguments, while in PharmML it has been defined with vector mean and covariance matrix. On the other hand, the individual parameter tau_e is defined via a univariate normal distribution.

```
        #correlated distribution of V and k of the j-subject
        lP[j,1:2]~dmnorm(lPpop[], TP[ , ])
```

in PharmML reads

```
            <PopulationParameter symbId="lP">
                <ct:VariabilityReference>
                    <ct:SymbRef symbIdRef="indiv" blkIdRef="model"/>
                </ct:VariabilityReference>
                <Distribution>
                    <ProbOnto name="MultivariateNormal1">
                        <Parameter name="mean">
                            <ct:Assign>
                                <ct:SymbRef symbIdRef="lPOP_P"/>
                            </ct:Assign>
                        </Parameter>
                        <Parameter name="covarianceMatrix">
```

```
                            <ct:Assign>
                                <ct:SymbRef symbIdRef="OMEGA_P"/>
                            </ct:Assign>
                        </Parameter>
                    </ProbOnto>
                </Distribution>
            </PopulationParameter>
```

and

```
        #distribution of taue of the j-subject
        ltaue[j]~dnorm(lTpop, Ttau)
        taue[j] <- exp(ltaue[j])
```

in PharmML reads

```
        <IndividualParameter symbId="TAU">
            <StructuredModel>
                <Transformation type="log"/>
                <LinearCovariate>
                    <PopulationValue>
                        <ct:Assign>
                            <ct:SymbRef symbIdRef="POP_T"/>
                        </ct:Assign>
                    </PopulationValue>
                </LinearCovariate>
                <RandomEffects>
                    <ct:SymbRef symbIdRef="eta_T"/>
                </RandomEffects>
            </StructuredModel>
        </IndividualParameter>
```

with implementation of *eta_T* not shown here as it uses the well known `<RandomVariable>` element for its encoding.
The V and k individual parameters have to be retrieved. The elements of the lP vector are log(V) and log(k), so the exponential of the lP elements must be computed to obtain V and k.

```
        lV[j] <- lP[j,1]
        V[j] <- exp(lV[j])
        lk[j] <- lP[j,2]
        k[j] <- exp(lk[j])
```

in PharmML reads

```
        <IndividualParameter symbId="V">
            <ct:Assign>
                <Equation xmlns="http://www.pharmml.org/pharmml/0.6/Maths">
                    <Uniop op="exp">
                        <ct:VectorSelector>
                            <ct:SymbRef symbIdRef="lP"/>
                            <ct:Cell>
                                <ct:Int>2</ct:Int>
                            </ct:Cell>
                        </ct:VectorSelector>
                    </Uniop>
                </Equation>
            </ct:Assign>
        </IndividualParameter>

        <!-- k omitted here -->
```

Priors specification
The prior model on the fixed effect 'tau_Ppop' reads

```
        # prior on "THETA"
        tau_Ppop[1:2, 1:2] <- inverse(sigma_Ppop[ , ])
```

and requires the definition of the matrix and it reads in PharmML as

```
            <PopulationParameter symbId="SIGMA_POP_P">
                <ct:Assign>
                    <ct:Matrix matrixType="Any">
                        <ct:MatrixRow>
                            <ct:RowIndex><ct:Int>1</ct:Int></ct:RowIndex>
                            <ct:Real>1</ct:Real>
                            <ct:Real>0.1</ct:Real>
                            </ct:MatrixRow>
                        <ct:MatrixRow>
                            <ct:RowIndex><ct:Int>2</ct:Int></ct:RowIndex>
                            <ct:Real>0.1</ct:Real>
                            <ct:Real>1</ct:Real>
                        </ct:MatrixRow>
                    </ct:Matrix>
                </ct:Assign>
            </PopulationParameter>
```

The vector of the population values

```
lmu_Ppop[1] <- log(mu_Vpop)
lmu_Ppop[2] <- log(mu_kpop)
```

reads in PharmML

```
<PopulationParameter symbId="lMU_POP_P">
    <ct:Assign>
        <ct:Vector>
            <ct:VectorElements>
                <ct:SymbRef symbIdRef="lMU_POP_K"/>
                <ct:SymbRef symbIdRef="lMU_POP_V"/>
            </ct:VectorElements>
        </ct:Vector>
    </ct:Assign>
</PopulationParameter>

<PopulationParameter symbId="lMU_POP_V">
    <ct:Assign>
        <Equation xmlns="http://www.pharmml.org/pharmml/0.6/Maths">
            <Uniop op="log">
                <ct:SymbRef symbIdRef="MU_POP_V"/>
            </Uniop>
        </Equation>
    </ct:Assign>
</PopulationParameter>
<!-- with -->
<PopulationParameter symbId="MU_POP_V"/>
<!-- k omitted here -->
```

The vector of log of the population parameters Vpop and kpop is a-priori distributed as a multivariate Normal with user-defined vector mean and covariance matrix, reported above.

```
lPpop[1:2]~dmnorm(lmu_Ppop[], tau_Ppop[ , ])
```

reads in PharmML

```
<PopulationParameter symbId="lPOP_P">
    <ct:VariabilityReference>
        <ct:SymbRef symbIdRef="pop" blkIdRef="model"/>
    </ct:VariabilityReference>
    <Distribution>
        <ProbOnto name="MultivariateNormal1">
            <Parameter name="mean">
                <ct:Assign>
                    <ct:SymbRef symbIdRef="lMU_POP_P"/>
                </ct:Assign>
            </Parameter>
            <Parameter name="covarianceMatrix">
                <ct:Assign>
                    <ct:SymbRef symbIdRef="SIGMA_POP_P"/>
                </ct:Assign>
            </Parameter>
        </ProbOnto>
    </Distribution>
</PopulationParameter>
```

with the according covariance matrix *SIGMA_POP_P* defined above.

The inverse of the covariance matrix *OMEGA_P* is distributed as a Wishart with given parameters R and rho (not reported). A different parametrization has been used in WinBUGS (in which the inverse of R is required) and PharmML (in which the Wishart1 distribution enables the use of the R matrix)

```
# prior on inverse of "OMEGA"
Rinv[1:2,1:2] <- inverse(R[,])
TP[1:2,1:2]~dwish(Rinv[ , ], rho)
```

in PharmML reads

```
<PopulationParameter symbId="OMEGA_P">
    <ct:Assign>
        <Equation xmlns="http://www.pharmml.org/pharmml/0.6/Maths">
            <MatrixUniop op="inverse">
                <ct:SymbRef symbIdRef="invOMEGA_P"/>
            </MatrixUniop>
        </Equation>
    </ct:Assign>
</PopulationParameter>

<PopulationParameter symbId="invOMEGA_P">
    <ct:VariabilityReference>
        <ct:SymbRef symbIdRef="pop" blkIdRef="model"/>
    </ct:VariabilityReference>
    <Distribution>
        <ProbOnto name="Wishart1">
            <Parameter name="scaleMatrix">
                <ct:Assign>
                    <ct:SymbRef symbIdRef="rho"/>
                </ct:Assign>
            </Parameter>
            <Parameter name="degreesOfFreedom">
                <ct:Assign>
                    <ct:SymbRef symbIdRef="R"/>
                </ct:Assign>
            </Parameter>
        </ProbOnto>
    </Distribution>
</PopulationParameter>
```

The inverse of the variance of *eta_T* (*OMEGA_T*) is a-priori distributed as a Gamma with user-defined parameters.

```
Ttau~dgamma(a_omega_tau, b_omega_tau)
```

in PharmML reads

```
<PopulationParameter symbId="TAU_T">
    <ct:VariabilityReference>
        <ct:SymbRef symbIdRef="pop" blkIdRef="model"/>
    </ct:VariabilityReference>
    <Distribution>
        <ProbOnto name="Gamma">
            <Parameter name="shape">
                <ct:Assign>
                    <ct:SymbRef symbIdRef="a_OMEGA_T"/>
                </ct:Assign>
            </Parameter>
            <Parameter name="scale">
                <ct:Assign>
                    <ct:SymbRef symbIdRef="b_OMEGA_T"/>
                </ct:Assign>
            </Parameter>
        </ProbOnto>
    </Distribution>
</PopulationParameter>

<PopulationParameter symbId="OMEGA_T">
    <ct:Assign>
        <Equation xmlns="http://www.pharmml.org/pharmml/0.6/Maths">
            <Binop op="divide">
                <ct:Real>1</ct:Real>
                <ct:SymbRef symbIdRef="TAU_T"/>
            </Binop>
        </Equation>
    </ct:Assign>
</PopulationParameter>
```

POP_T is a-priori distributed as a Gamma with user-defined parameters.

```
# prior on "SIGMA"
Tpop~dgamma(a_taupop, b_taupop)
lTpop <- log(Tpop)
```

in PharmML reads

```
<PopulationParameter symbId="POP_T">
    <ct:VariabilityReference>
        <ct:SymbRef symbIdRef="pop" blkIdRef="model"/>
    </ct:VariabilityReference>
    <Distribution>
        <ProbOnto name="Gamma">
            <Parameter name="shape">
                <ct:Assign>
                    <ct:SymbRef symbIdRef="a_POP_T"/>
                </ct:Assign>
            </Parameter>
            <Parameter name="scale">
                <ct:Assign>
                    <ct:SymbRef symbIdRef="b_POP_T"/>
                </ct:Assign>
            </Parameter>
        </ProbOnto>
    </Distribution>
</PopulationParameter>
```

The structural model is a standard algebraic equations and its implementation will not be shown here. The Observation model on the other hand is implemented using the distribution model and read in PharmML:

```
<General symbId="C">
    <ct:VariabilityReference>
        <ct:SymbRef blkIdRef="resErrorModel" symbIdRef="residual"/>
    </ct:VariabilityReference>
    <Distribution>
        <ProbOnto name="Normal2">
            <Parameter name="mean">
                <ct:Assign>
                    <ct:SymbRef blkIdRef="sm1" symbIdRef="C"/>
                </ct:Assign>
            </Parameter>
            <Parameter name="var">
                <ct:Assign>
                    <ct:SymbRef symbIdRef="sigmaSquare"/>
                </ct:Assign>
            </Parameter>
        </ProbOnto>
    </Distribution>
</General>
```

with

```
<IndividualParameter symbId="sigmaSquare">
    <ct:Assign>
        <Equation xmlns="http://www.pharmml.org/pharmml/0.6/Maths">
            <Uniop op="sqrt">
                <Binop op="divide">
                    <ct:Real>1</ct:Real>
                    <ct:SymbRef symbIdRef="TAU"/>
                </Binop>
            </Uniop>
        </Equation>
    </ct:Assign>
</IndividualParameter>
```

declared in parameter model, `pm1`.

4.3 Hierarchical model example

The following model has been proposed by Marc Lavielle, [10], to test the capabilities of the DDMoRe platform with respect to the encoding of hierarchical models, encoded here in MLXTRAN

```
[LONGITUDINAL]
input = {V, k, b}
EQUATION:
D=100
f = D/V*exp(-k*t)
DEFINITION:
y = {distribution=normal, prediction=f, sd=b*f}

[INDIVIDUAL]
```

```
input = {V_pop, omega_V, w, w_pop}
EQUATION:
V_pred = V_pop*(w/w_pop)
DEFINITION:
V = {distribution=logNormal, prediction=V_pred, sd=omega_V}

[COVARIATE]
input = {w_pop, omega_w}
DEFINITION:
w = {distribution=normal, mean=w_pop, sd=omega_w}

[POPULATION]
input = {ws, gw, Vs, gV}
DEFINITION:
w_pop = {distribution=normal, mean=ws, sd=gw}
V_pop = {distribution=logNormal, mean=log(Vs), sd=gV}
```

In the following we will show only the relevant for this chapter model elements.

Observation model

$$y \sim \mathcal{N}(f, bf)$$

reads in PharmML

```
<General symbId="y">
    <ct:VariabilityReference>
        <ct:SymbRef blkIdRef="vm2" symbIdRef="resErr"/>
    </ct:VariabilityReference>
    <Distribution>
        <ProbOnto name="Normal1">
            <Parameter name="mean">
                <ct:Assign>
                    <ct:SymbRef blkIdRef="sm1" symbIdRef="f"/>
                </ct:Assign>
            </Parameter>
            <Parameter name="stdev">
                <ct:Assign>
                    <math:Equation>
                        <math:Binop op="times">
                            <ct:SymbRef blkIdRef="pm1" symbIdRef="b"/>
                            <ct:SymbRef blkIdRef="sm1" symbIdRef="f"/>
                        </math:Binop>
                    </math:Equation>
                </ct:Assign>
            </Parameter>
        </ProbOnto>
    </Distribution>
</General>
```

Note the encoding of expressions in the second parameter of the normal distribution easily done when using ProbOnto, was not possible with UncertML.

Individual parameter model

$$V \sim \mathcal{LN}(V_{pred}, \omega_V) \quad \text{with} \quad V_{pred} = V_{pop}(w/w_{pop})$$

reads in PharmML:

```
<IndividualParameter symbId="V">
    <ct:VariabilityReference>
        <ct:SymbRef blkIdRef="vm1" symbIdRef="indiv"/>
    </ct:VariabilityReference>
    <Distribution>
        <ProbOnto name="LogNormal1">
            <Parameter name="meanLog">
                <ct:Assign>
                    <ct:SymbRef symbIdRef="V_pred"/>
                </ct:Assign>
            </Parameter>
            <Parameter name="stdevLog">
                <ct:Assign>
                    <ct:SymbRef symbIdRef="omega_V"/>
                </ct:Assign>
            </Parameter>
        </ProbOnto>
    </Distribution>
</IndividualParameter>

<!-- V_pred = V_pop*(w/w_pop) -->
<PopulationParameter symbId="V_pred">
    <ct:Assign>
        <math:Equation>
            <math:Binop op="times">
                <ct:SymbRef symbIdRef="V_pop"/>
                <math:Binop op="divide">
                    <ct:SymbRef blkIdRef="cm1" symbIdRef="w"/>
                    <ct:SymbRef symbIdRef="w_pop"/>
                </math:Binop>
            </math:Binop>
        </math:Equation>
    </ct:Assign>
</PopulationParameter>
```

Covariate model describes the distribution of body weight

$$w \sim \mathcal{N}(w_{pop}, \omega_w)$$

can be encoded as

```
    <CovariateModel blkId="cm1">
        <Covariate symbId="w">
            <Continuous>
                <Distribution>
                    <ProbOnto name="Normal1">
                        <Parameter name="mean">
                            <ct:Assign>
                                <ct:SymbRef blkIdRef="pm1" symbIdRef="w_pop"/>
                            </ct:Assign>
                        </Parameter>
                        <Parameter name="stdev">
                            <ct:Assign>
                                <ct:SymbRef blkIdRef="pm1" symbIdRef="omega_w"/>
                            </ct:Assign>
                        </Parameter>
                    </ProbOnto>
                </Distribution>
            </Continuous>
        </Covariate>
    </CovariateModel>
```

Note, that the population parameters used in the covariate model are defined in the parameter model, `pm1`, as the next code snippet will show.

Population parameters of the model covariate model

$$w_{pop} \sim \mathcal{N}(ws, gw)$$

and

$$V_{pop} \sim \mathcal{LN}\big(\log(Vs), gV\big)$$

are encode as

```
    <!-- w_pop = {distribution=normal, mean=ws, sd=gw} -->
    <PopulationParameter symbId="w_pop">
        <ct:VariabilityReference>
            <ct:SymbRef blkIdRef="vm1" symbIdRef="pop"/>
        </ct:VariabilityReference>
        <Distribution>
            <ProbOnto name="Normal1">
                <Parameter name="mean">
                    <ct:Assign>
                        <ct:SymbRef symbIdRef="ws"/>
                    </ct:Assign>
                </Parameter>
                <Parameter name="stdev">
                    <ct:Assign>
                        <ct:SymbRef symbIdRef="gw"/>
                    </ct:Assign>
                </Parameter>
            </ProbOnto>
        </Distribution>
    </PopulationParameter>

    <!-- V_pop = {distribution=logNormal, mean=log(Vs), sd=gV} -->
    <PopulationParameter symbId="V_pop">
        <ct:VariabilityReference>
            <ct:SymbRef blkIdRef="vm1" symbIdRef="pop"/>
        </ct:VariabilityReference>
        <Distribution>
            <ProbOnto name="LogNormal1">
                <Parameter name="meanLog">
                    <ct:Assign>
                        <math:Equation>
                            <math:Uniop op="log">
                                <ct:SymbRef symbIdRef="Vs"/>
                            </math:Uniop>
                        </math:Equation>
                    </ct:Assign>
                </Parameter>
                <Parameter name="stdevLog">
                    <ct:Assign>
                        <ct:SymbRef symbIdRef="gV"/>
                    </ct:Assign>
                </Parameter>
            </ProbOnto>
        </Distribution>
    </PopulationParameter>
```

4.4 Additional examples implemented

In addition eight examples as described in [3] have been successfully implemented in PharmML 0.7 and are provided with the release: *example331.xml*, *example332.xml*, *example333.xml*, *example334.xml*, *example335.xml*, *example3311.xml*, *example3312.xml*, *example3321.xml*.

Chapter 5

Design – redesigned

5.1 Design in PharmML ≤ 0.6

Since version 0.2 PharmML was using SDM-XML, a CDISC standard [2], based trial design. With the beginning of the activities of a group working on trial design for MDL it became clear that this structure is insufficient and had to be reorganised and redesigned, see figure 5.1. The main reasons are

- CDISC based design is missing many elements, the most important ones being the observations and design spaces. Although the former were supported in PharmML, their implementation not only had to be done in `<ModellingSteps>` section but it also lacked the connectivity to study arms.
- A number of typical design features, such as arm size, number of arms, total size, number of samples, number of times, cost function, total cost, was not accounted for.
- The rigid structure with epochs/cells/segment was unfamiliar to modellers who need a more flexible arm based structure.
- Required specifically for optimal design, the re-definition of covariate model, required for covariates to be optimised, was not supported in the CDISC based structure.
- Design elements were spread over two sections, `<TrialDesign>` and `<ModellingSteps>` which created an inconsistent structure.
- The specification of external datasets, was located in `<ModellingSteps>`, but as the source for design it belongs to trial design section.
- `<ModellingSteps>` section should contain only task description.
- The lack of compatibility with the new MDL design proposal – translation would be very hard to achieve given such two different structures.

The new design/trial design structure addresses the above issues and because it was developed with both MDL and PharmML in mind, it promises a very high degree of compatibility between these two languages – to be verified during the coming months.

5.2 Modifications and extensions

The new trial design structure consists of

- Core structure based on the design elements developed for MDL, [5, 4].
- Additional features, some of which were available in PharmML since version 0.2.1 such as individual observations, dosing, covariates.

While the detailed design is described in the according MDL specification proposal, [5], the comparison on the following pages visualises the changes and differences. In the left column on page 41, the PharmML ≤ 0.6 design elements are shown, distributed over two sections, `<TrialDesign>` and `<ModellingSteps>`. On the right, a detailed list of current elements available for the trial design is given.

All examples provided in the 'Modelling Description Language. Design elements - Examples', [4], and all explicit design based examples from the PharmML specification, [21], have been successfully implemented.

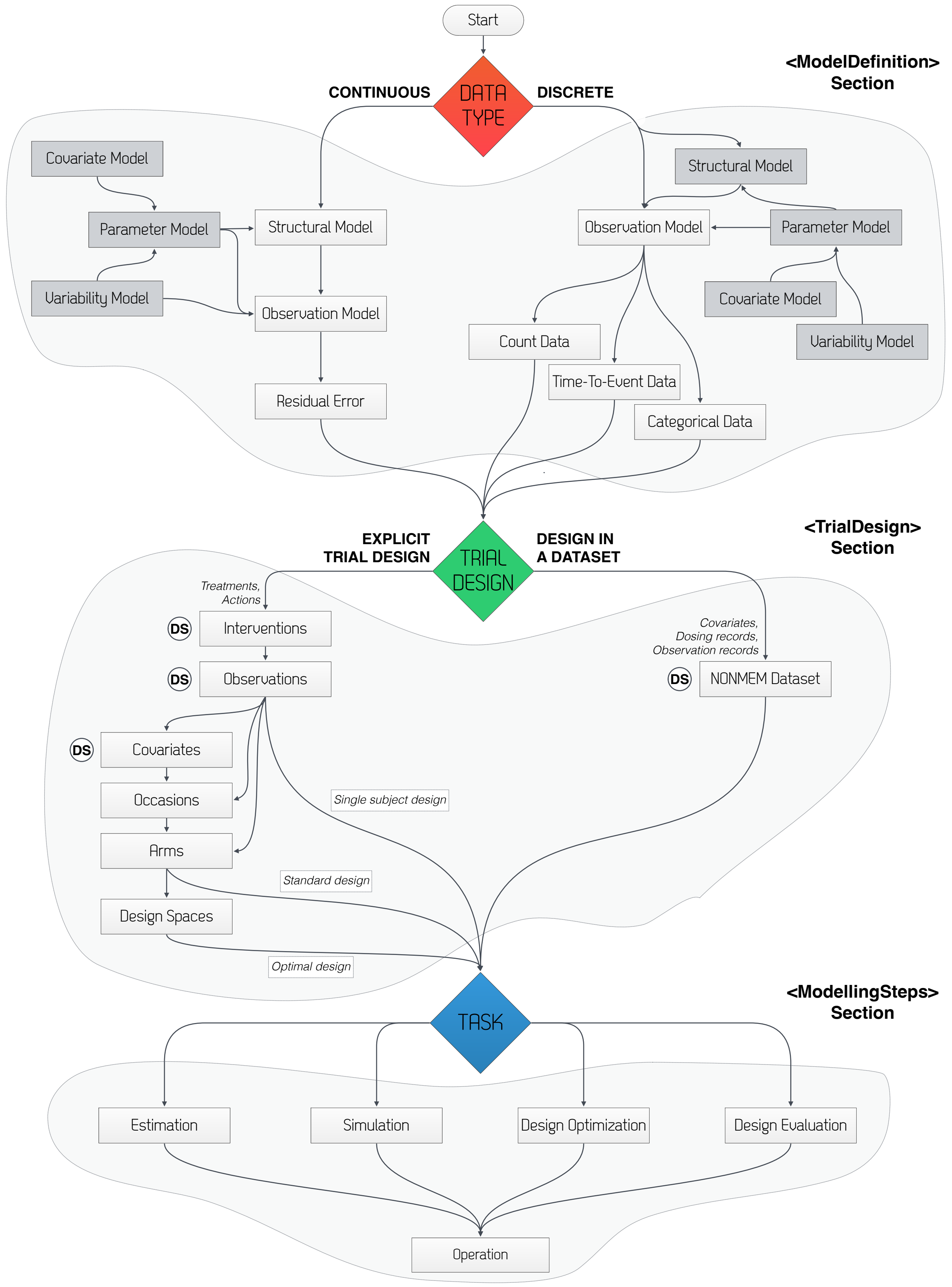


Figure 5.1: Working with PharmML – a schema showing three essential decision points: (A) the type of data (continuous and discrete), (B) the source of the study design and data (Monolix/NONMEM dataset or `<TrialDesign>`) and finally (C) the task type (for now only simulation and estimation tasks are fully supported, the design related tasks are under construction). **Note** that in comparison to 0.6 version, [21], all data and design elements are located in the `<TrialDesign>` section while the last section, `<ModellingSteps>`, carries only tasks related information.

PharmML ≤ 0.6
<TrialDesign>

- Structure
  - Arm
  - Cell
  - Epoch
  - Segment
  - Activity
    - Bolus
    - Infusion
    - Washout
    - Lookup table
    - Epoch Ref/Period
- Population
  - Arm memberships & covariates
  - Demographics
- Individual Dosing

<ModellingSteps>

- **ExternalDataSet**
- Simulation Step
  - **Observations**
  - Operation
- Estimation Step
  - **ObjectiveData**
  - Parameter Estimation
  - Operation
- Step dependencies

PharmML 0.7
<TrialDesign>

- ExternalDataSet
- Interventions
  - Administration
    - InterventionRef
    - Bolus
    - Infusion
  - IndividualAdministration
  - Action – Washout
    - full reset
    - variable-wise
  - InterventionsCombination
- Observations
  - LookupTable
  - IndividualObservations
  - Observation
  - ObservationsCombination
- Covariates
  - CovariateModel
    - Categorical/Category: Probability, OccasionRef, InterventionRef, InterventionSequence
  - IndividualCovariates
- Occasions/OccasionList
  - VariabilityReference
  - Occasion
- Arms
  - Simple elements: ArmSize, CostFunction, NumberArms, NumberSamples, NumberTimes, SameTimes, TotalCost, TotalSize
  - Arm
    - Simple elements: ArmSize, NumberSamples, NumberTimes, SameTimes
    - InterventionSequence/List
    - ObservationSequence/List
    - OccasionSequence/List
- DesignSpaces
  - References: InterventionRef, ObservationRef, ArmRef, SymbRef, CovariateModelRef & CovariateRef
  - Simple elements: ArmSize, DoseAmount, DosingTimes, Duration, NumberArms, NumberSamples, NumberTimes, SampleTimes

<ModellingSteps>

- Simulation Step / Operation
- Estimation Step / Operation
- Design Optimization Step – UNDER CONSTRUCTION
- Design Evaluation Step – UNDER CONSTRUCTION
- Step dependencies

5.3 Selected features

The trial design structure, although intuitive and clearly structured, is quite complex. The reader is referred to the original specification and example documents, [4, 5]. Here we describe only few examples of selected new features supported in this release and indicate differences between PharmML and MDL.

5.3.1 Actions – washout/reset options

Washout can now be customised while in previous versions only full reset was possible. One can define it for specific/selected variables in the model, e.g. for A1 – which is set to 10 at t=0.5, using `<VariableToReset>` with optional `<ResetValue>` and `<ResetTime>`

```
            <Action oid="W1">
                <Washout>
                    <VariableToReset>
                        <ct:SymbRef symbIdRef="A1"/>
                        <ResetValue>10</ResetValue>
                        <ResetTime>0.5</ResetTime>
                    </VariableToReset>
                </Washout>
            </Action>
```

or alternatively for the entire model using the `<FullReset>`

```
            <Action oid="w2">
                <Washout>
                    <VariableToReset>
                        <FullReset/>
                    </VariableToReset>
                </Washout>
            </Action>
```

5.3.2 `<Interval>` element

The `<Interval>` with children elements `<LeftEndpoint>` and `<RightEndpoint>` has been designed for the use e.g. in defining the design spaces as the following example for optimising the observation times shows (but use in other situations is possible as well).

```
    <DesignSpaces>
        <DesignSpace>
            <ObservationRef oidRef="window1"/>
            <ObservationTimes>
                <ct:Assign>
                    <ct:Interval>
                        <ct:LeftEndpoint>
                            <ct:Assign>
                                <ct:Real>0</ct:Real>
                            </ct:Assign>
                        </ct:LeftEndpoint>
                        <ct:RightEndpoint>
                            <ct:Assign>
                                <ct:Real>72</ct:Real>
                            </ct:Assign>
                        </ct:RightEndpoint>
                    </ct:Interval>
                </ct:Assign>
            </ObservationTimes>
        </DesignSpace>
```

First the observation in question is specified with `<ObservationRef>`, here `window1`, and subsequently the allowed interval, $[0, 72]$, is defined.

By default it is assumed that an endpoint is closed but the attribute `type` is available with two values {closed, open} to specified it accordingly to the model requirements. E.g. for $[0, 72)$[1] design space the following snippet shows the correct implementation

```
        <ct:LeftEndpoint>
            <!-- omitted details -->
        </ct:LeftEndpoint>
        <ct:RightEndpoint type="open">
            <!-- omitted details -->
        </ct:RightEndpoint>
```

[1]Note that in French notation this interval would be denoted as $[0, 72[$.

5.3.3 Using parameters in optimal design tasks

Consider a case of design optimisation, e.g. example 4, task 3 in [4]. Original description of a task where a parameter, *tdoseB*, is required, reads:
We assume now a sequential trial, where treatment B is given at time tdoseB after treatment A without washout, and we want to optimise the time between the two treatments.

step 1 First, the design parameter is declared and initialised:

```
<Interventions>
      <mdef:DesignParameter symbId="tdoseB">
          <ct:Assign>
              <ct:Real>0</ct:Real>
          </ct:Assign>
      </mdef:DesignParameter>
```

step 2 and used to define an administration, `trtB`, with dose amount equal 100 and dosing time defined as a sequence, starting at *tdoseB*, ending with *tdoseB*+72 with steps every 24, which is implemented as

```
<Administration oid="trtB">
    <Bolus>
        <DoseAmount inputTarget="admType">
            <TargetMapping blkIdRef="sm1">
                <ds:Map admNumber="2"/>
            </TargetMapping>
            <ct:Assign>
                <ct:Real>100</ct:Real>
            </ct:Assign>
        </DoseAmount>
        <DosingTimes>
            <ct:Assign>
                <ct:Sequence>
                    <ct:Begin>
                        <ct:SymbRef symbIdRef="tdoseB"/>
                    </ct:Begin>
                    <ct:StepSize>
                        <ct:Real>24</ct:Real>
                    </ct:StepSize>
                    <ct:End>
                        <Equation>
                            <Binop op="plus">
                                <ct:SymbRef symbIdRef="tdoseB"/>
                                <ct:Real>72</ct:Real>
                            </Binop>
                        </Equation>
                    </ct:End>
                </ct:Sequence>
            </ct:Assign>
        </DosingTimes>
    </Bolus>
</Administration>
```

step 3 Finally, the design space for *tdoseB* is defined as an interval [0,72]:

```
    <DesignSpace>
        <ct:SymbRef symbIdRef="tdoseB"/>
        <ct:Assign>
            <ct:Interval>
                <ct:LeftEndpoint type="closed">
                    <ct:Assign>
                        <ct:Real>0</ct:Real>
                    </ct:Assign>
                </ct:LeftEndpoint>
                <ct:RightEndpoint type="closed">
                    <ct:Assign>
                        <ct:Real>72</ct:Real>
                    </ct:Assign>
                </ct:RightEndpoint>
            </ct:Interval>
        </ct:Assign>
    </DesignSpace>
</DesignSpaces>
```

Note that design spaces are supposed to deal with any element of the design. The example document, [4] and their PharmML implementation, feature multiple application cases.

5.3.4 Defining conditional covariate distribution

Covariates can vary from arm to arm and/or be dependent on other covariates, such as sex etc. Instead of defining them separately in according arms. Using a conditional distribution defined in the `<Covariates>` block outside the `<Arms>`, is a very effective way to express it. For example consider the following model

$$P(WT|ARM) = \begin{cases} \mathcal{N}\big(WT_{mean1}, WT_{variance1}\big) & \text{if} \quad ARM = arm1 \\ \mathcal{N}\big(WT_{mean2}, WT_{variance2}\big) & \text{if} \quad ARM = arm2 \end{cases}$$

The code is similar to that of conditional covariate distributions, shown in Section 6.5, and will not be repeated here with the exception of the `<Condition>` element using in this case the literal `<True>` of the Boolean data type

```
<Condition>
    <!-- "arm1" -->
    <LogicBinop op="eq">
        <ArmRef oidRef="arm1"/>
        <ct:True/>
    </LogicBinop>
</Condition>
```

5.3.5 Optimising covariates distribution

One of the objectives in optimal design is to optimise the distribution of relevant covariates, see *example3_taks2.xml* or the original MDL file, with design elements to optimise on

- proportion of each genotype &
- proportion of each gender

In such case the initial covariate model is encoded in `<ModelDefinition>` as the following code snippet shows

```
<CovariateModel blkId="cm1">
    <Covariate symbId="SEX">
        <Categorical>
            <Category catId="F"/>
            <Category catId="M"/>
        </Categorical>
    </Covariate>
    <Covariate symbId="Genetics">
        <Categorical>
            <Category catId="common_Hz"/>
            <Category catId="hz"/>
            <Category catId="rare_hz"/>
        </Categorical>
    </Covariate>
```

which can be overwritten in `<TrialDesign>` if required. The covariate model, if it applies to all arms, will be defined just after the `<Interventions>` and `<Observations>` (another option is to define arm-specific covariates within arms).

Then the covariate model in the trial design starts with a refernce to the base covariance model to be optimised.

```
<TrialDesign xmlns="http://www.pharmml.org/pharmml/0.6/TrialDesign">

    <mdef:DesignParameter symbId="Genp1">
        <!-- omitted details -->
    </mdef:DesignParameter>
    <!-- omitted Genp2 declaration -->
    <mdef:DesignParameter symbId="Genp3">
        <!-- omitted details -->
    </mdef:DesignParameter>

    <Interventions>
        <!-- omitted details -->
    </Interventions>
    <Observations>
        <!-- omitted details -->
    </Observations>

    <Covariates>
        <!-- COVARIATE MODEL - overwritting covariate model defined in ModelDefinition -->
        <CovariateModel oid="td_cm1">
            <CovariateModelRef blkIdRef="cm1"/>
            <Covariate symbId="SEX">
                <mdef:Categorical>
                    <mdef:Category catId="M">
                        <mdef:Probability>
                            <ct:Real>0.5</ct:Real>
                        </mdef:Probability>
                    </mdef:Category>
                    <mdef:Category catId="F">
                        <mdef:Probability>
                            <ct:Real>0.5</ct:Real>
                        </mdef:Probability>
                    </mdef:Category>
                </mdef:Categorical>
            </Covariate>
            <Covariate symbId="Genetics">
                <mdef:Categorical>
                    <mdef:Category catId="common_Hz">
                        <mdef:Probability>
                            <ct:SymbRef symbIdRef="Genp1"/>
                        </mdef:Probability>
                    </mdef:Category>
                    <mdef:Category catId="hz">
                        <mdef:Probability>
                            <ct:SymbRef symbIdRef="Genp2"/>
                        </mdef:Probability>
                    </mdef:Category>
                    <mdef:Category catId="rare_hz">
                        <mdef:Probability>
                            <ct:SymbRef symbIdRef="Genp3"/>
                        </mdef:Probability>
                    </mdef:Category>
                </mdef:Categorical>
            </Covariate>
        </CovariateModel>
    </Covariates>
    ...
```

The covariate model overwrites the base models in the `<ModelDefinition>` section, which is referenced with `<CovariateModelRef blkIdRef="cm1">`. Also in this case we use `<DesignParameter>` element introduced in Section 5.3.3, here *Genp1*, . . . , *Genp3* denoting the proportions of each genotype, to be used later in the design spaces. Note that `<DesignParameter>` can be defined anywhere in the `<TrialDesign>` and is valid in the entire section.

Finally the design spaces can be specified, here an interval [0.25, 1] for *Genp1*

```
        <DesignSpaces>
            <DesignSpace>
                <ct:SymbRef symbIdRef="Genp1"/>
                <ct:Assign>
                    <ct:Interval>
                        <ct:LeftEndpoint>
                            <ct:Assign>
                                <ct:Real>0.25</ct:Real>
                            </ct:Assign>
                        </ct:LeftEndpoint>
                        <ct:RightEndpoint>
                            <ct:Assign>
                                <ct:Real>1</ct:Real>
                            </ct:Assign>
                        </ct:RightEndpoint>
                    </ct:Interval>
                </ct:Assign>
            </DesignSpace>
            <!-- omitted design spaces for remaining design parameters-->
        </DesignSpaces>
```

Note that the covariate model in the trial design uses the object identifiers, `oid`, rather then block identifiers, `blkId`. The change was required for the consistency with the use of the former in the scope of the

`<TrialDesign>` section.

5.4 Single subject design without arms

This option is not currently supported in MDL proposal but was requested by the modellers. It can be used

- in single subject scenarios to shorten the implementation burden as such cases don't require a typical explicit design structure (with arms).
- to support existing simulation tools, such as Simulx.

It comes with the possibility to define dosing, observations, lookup tables etc. without the need to define the study arms. For consistency, the dosing, observations times etc. information will still be encoded in `<TrialDesign>` but without placing them within the `<Arms>` tags.

Example 1. As a simple illustration consider a Simulx example[2] with explicit defined administration

```
adm1 <- list(type=1, amount=100, time=seq(0, 84, by=6))
adm2 <- list(type=2, amount=50, time=seq(3, 87, by=12), tinf=1)
```

and observations[3]

```
Cc <- list(name='Cc', time=seq(-5, 100, by=.1))
```

for a model defined as a combination of PK macros and ODEs (here in MLXTRAN speak):

```
PK:
depot(type=1, target=Ad, p=F)
depot(type=2, target=Ac)

EQUATION:
k = Cl/V
ddt_Ad = -ka*Ad
ddt_Ac =  ka*Ad - k*Ac
Cc = Ac/V
```

The administration related part of the show case can be easily implemented using the `<Administration>` element within `<Interventions>` tag

```
<TrialDesign>
    <Interventions>
        <Administration oid="adm1">
            <Bolus>
                <DoseAmount inputTarget="derivativeVariable">
                    <ct:SymbRef symbIdRef="Ad"/>
                    <ct:Assign>
                        <ct:Real>100</ct:Real>
                    </ct:Assign>
                </DoseAmount>
                <DosingTimes>
                    <ct:Assign>
                        <ct:Sequence>
                            <ct:Begin><ct:Real>0</ct:Real></ct:Begin>
                            <ct:StepSize><ct:Real>6</ct:Real></ct:StepSize>
                            <ct:End><ct:Real>84</ct:Real></ct:End>
                        </ct:Sequence>
                    </ct:Assign>
                </DosingTimes>
            </Bolus>
        </Administration>
        <Administration oid="adm2">
            <Infusion>
                <DoseAmount inputTarget="derivativeVariable">
                    <ct:SymbRef symbIdRef="Ac"/>
                    <ct:Assign>
                        <ct:Real>50</ct:Real>
                    </ct:Assign>
                </DoseAmount>
                <DosingTimes>
                    <!-- skipped, as similar to adm1 -->
                </DosingTimes>
```

[2] http://webpopix.org:8080/dashboard/administration/
[3] The original code uses 'length' argument which is currently not supported in the `<Sequence>` element.

```
                    <Duration>
                        <ct:Assign>
                            <ct:Real>1</ct:Real>
                        </ct:Assign>
                    </Duration>
                </Infusion>
            </Administration>
        </Interventions>
```

Finally the observations are implemented as

```
        <Observations>
            <Observation oid="OBSoid_Cc">
                <ObservationTimes>
                    <ct:Assign>
                        <ct:Sequence>
                            <ct:Begin><ct:Real>-5</ct:Real></ct:Begin>
                            <ct:StepSize><ct:Real>.1</ct:Real></ct:StepSize>
                            <ct:End><ct:Real>100</ct:Real></ct:End>
                        </ct:Sequence>
                    </ct:Assign>
                </ObservationTimes>
                <Continuous>
                    <ct:SymbRef symbIdRef="Cc"/>
                </Continuous>
            </Observation>
        </Observations>
    </TrialDesign>
```

The `<SimulationStep>` will contain the references to the interventions and observations in question as shown in the code snippet

```
    <ModellingSteps xmlns="http://www.pharmml.org/pharmml/0.6/ModellingSteps">

        <SimulationStep oid="simTask1">
            <InterventionsReference>
                <ct:OidRef oidRef="adm1"/>
                <ct:OidRef oidRef="adm2"/>
            </InterventionsReference>

            <ObservationsReference>
                <ct:OidRef oidRef="OBSoid_Cc"/>
            </ObservationsReference>
        </SimulationStep>
    </ModellingSteps>
```

Note that the `<InterventionsReference>` element is new in this version.

5.5 Design examples implemented

The examples listed below coming with the MDL proposal, [4], have been successfully implemented in PharmML and are provided with the release (in total 16 examples):

- example1: Basic PK model with five tasks
- example2: PK/PD model with four tasks
- example3: PK model with two covariates with three tasks
- example4: PK with IOV and treatment covariate (different for each occasion)
- example5: Combination of two treatments - evaluation and optimisation

5.6 Differences compared to MDL

Here a short list of features related to the trial design not yet covered in MDL

- Conditional distributions, see section 5.3.4.
- Defining dosing/observations without arm structure, see section 5.4, is not yet available in MDL (or just not described in the MDL design spec, [5]) but is straight forward to achieve.
- Encoding of individual observations/administrations/covariates (available in PharmML since version 0.2.1).

Chapter 6

Other changes

6.1 Box-Cox transformation

The Box-Cox Transformation, [1], is used to transform data to an approximate normal distribution and reads

$$h(x) = \begin{cases} \frac{x^\lambda - 1}{\lambda} & \text{for} \quad \lambda \neq 0 \\ \log(x) & \text{for} \quad \lambda = 0 \end{cases}$$

It can be applied to parameters, observations and covariates. A random variable, X, is called power-normally distributed if its Box-Cox transformation is normally distributed, $h(x) \sim \mathcal{N}(\mu, \omega^2)$, and truncated so that $h(x) > 0$, [9].

6.1.1 Box-Cox in parameter model

A parameter to which the Box-Cox transformation is applied

$$V_i^{(\lambda)} = V_{pop}^{(\lambda)} + \eta_V$$

can be implemented in PharmML as

```
<IndividualParameter symbId="V">
    <StructuredModel>
        <Transformation type="BoxCox">
            <Parameter>
                <ct:Assign>
                    <ct:SymbRef blkIdRef="pm1" symbIdRef="lambda"/>
                </ct:Assign>
            </Parameter>
        </Transformation>
        <PopulationValue>
            <ct:Assign>
                <ct:SymbRef blkIdRef="pm1" symbIdRef="pop_V"/>
            </ct:Assign>
        </PopulationValue>
        <RandomEffects>
            <ct:SymbRef symbIdRef="eta_V"/>
        </RandomEffects>
    </StructuredModel>
</IndividualParameter>
```

Note the new `type` attribute value, *BoxCox*, in the `<Transformation>` tag and the associated additional element `<Parameter>` where the Box-Cox parameter is defined. The rest follows the pattern for the `<StructuredModel>`. Here the redesigned structure is used of `<StructuredModel>` in that the `<PopulationValue>` element can act alone without the `<LinearCovariate>` parent element, see section 3.2 for more details.

6.1.2 Box-Cox in observation model

The following observation model

$$Cc_{obs}^{(\lambda)} = Cc^{(\lambda)} + a\epsilon, \quad \text{with} \quad \epsilon \sim N(0,1)$$

is implemented analogously as

```
<Standard symbId="Cc_obs">
    <Transformation type="BoxCox">
        <Lambda>
            <ct:Assign>
                <ct:SymbRef blkIdRef="pm1" symbIdRef="lambda"/>
            </ct:Assign>
        </Lambda>
    </Transformation>
    <Output>
        <ct:SymbRef blkIdRef="sm1" symbIdRef="Cc"/>
    </Output>
    <ErrorModel>
        <ct:Assign>
            <ct:SymbRef blkIdRef="pm1" symbIdRef="a"/>
        </ct:Assign>
    </ErrorModel>
    <ResidualError>
        <ct:SymbRef symbIdRef="epsilon_Cc"/>
    </ResidualError>
</Standard>
```

6.1.3 Box-Cox in covariate model

The encoding of the Box-Cox transformation is already possible using the standard `<FunctionDefinition>` element and for now no new structure has been introduced.

6.2 Encoding of missing data

Following discussion at Pavia meeting November 2013, described in the meeting report [18], and later comments we have extended PharmML both for inline or external storage of data records. We reuse symbols for special numerical values used in R (see is.finite{base}) and SAS (see SAS/IML(R) 9.22)[1] to provide means to encode missing data.

6.2.1 Inline stored datasets

When data is stored inline within PharmML, few new predefined XML elements are required. Here the list of typical missing data types and corresponding elements introduced in this version:

- **NA** – not available/missing data, `<NA>`
- **NaN** – not a number – impossible values (e.g., dividing by zero), `<NaN>`
- **+Inf/-Inf** – positive/negative infinity, `<plusInf>` and `<minusInf>`
- **BLQ/ALQ** – below/above level of quantification[2], `<BLQ>` and `<ALQ>`

The following hypothetical dataset with several missing DV values

ID	TIME	DV
1	3.43	-99
1	5.2	48.03
1	42.13	L
1	52.63	+INF
1	57.53	72.3
...	...	...

Table 6.1: A dataset with examples of missing data.

can be encoded in PharmML as the following snippet shows

[1]Other possible codes are **DEE** or **-99** (SAS) – data entry error or **SRA** or **-999** (SAS) – subject refused answer, see `http://www.ats.ucla.edu/stat/sas/faq/how_to_code_missing_differently.htm`, but will not be introduced unless required

[2]**L/H** – symbols used in dataset in ADAPT5, [6]

```
    <Definition>
        <Column columnId="ID" columnType="id" valueType="id" columnNum="1"/>
        <Column columnId="TIME" columnType="idv" valueType="real" columnNum="2"/>
        <Column columnId="DV" columnType="dv" valueType="real" columnNum="3"/>
    </Definition>
    <Table>
        <!-- SUBJECT 1 -->
        <Row><ct:Id>i1</ct:Id><ct:Real>3.43</ct:Real><ct:NA/></Row>
        <Row><ct:Id>i1</ct:Id><ct:Real>5.3</ct:Real><ct:Real>48.03</ct:Real></Row>
        <Row><ct:Id>i1</ct:Id><ct:Real>42.13</ct:Real><ct:BLQ/></Row>
        <Row><ct:Id>i1</ct:Id><ct:Real>52.63</ct:Real><ct:plusInf/></Row>
        <Row><ct:Id>i1</ct:Id><ct:Real>57.53</ct:Real><ct:Real>72.3</ct:Real></Row>
```

using the above defined elements.

Note 1 Two of the elements described above, NA and ∞, were available before as child elements of `<Constant>` – now merged with other *missing values* for consistency. The current version is simpler as it does allow to specify the missing values directly, without a parent element.

Note 2 The missing data encoding capabilities might be also very useful in the SO, which inherits the dataset definition from PharmML. For this to work the SO has to upgraded to be compliant with PharmML 0.7 so that it can make advantage of these features.

6.2.2 External datasets

Similar strategy is followed if an external dataset contains *missing data* records and this information needs to be passed to the target tool. Following new elements are available

- `<MissingData>` element with two attributes
  - `dataCode` – any symbol used in the referenced dataset
  - `missingDataType` – one of {*NA*, *NaN*, *plusInf*, *minusInf*, *BLQ*, *ALQ* }, consistent with elements introduced in the previous section.

The dataset used previously, table 6.1, is assumed now to be stored externally as *myFile.csv*, see `<path>` element below. The missing data codes used in such the dataset are mapped to the allowed types as shown in the following code

```
        <ExternalDataSet toolName="NONMEM" oid="NMoid">
            <ds:DataSet>
                <ds:Definition>
                    <ds:Column columnId="ID" columnType="id" valueType="id" columnNum="1"/>
                    <ds:Column columnId="TIME" columnType="idv" valueType="real" columnNum="2"/>
                    <ds:Column columnId="DV" columnType="dv" valueType="real" columnNum="3"/>
                </ds:Definition>
                <ds:ExternalFile oid="extFile">
                    <ds:path>myFile.csv</ds:path>
                    <MissingData dataCode="-99" missingDataType="NA"/>
                    <MissingData dataCode="+INF" missingDataType="plusInf"/>
                    <MissingData dataCode="L" missingDataType="BLQ"/>
                </ds:ExternalFile>
            </ds:DataSet>
        </ExternalDataSet>
```

It is important to note that mappings within `<MissingData>` will vary between datasets coming from different tools and must be supplied each time by the user/tool writing the PharmML along with the particular dataset.

6.3 Dataset headers

Even though headers are currently not used by the main estimation target tools, Monolix or NONMEM, to carry any information which can directly be interpreted and used, others, such as Simcyp Simulator, makes frequent use of headers.

Therefore, new `<Header>` and `<HeaderRow>` elements, placed in the `<Definition>` and `<Table>` parts of the dataset, respectively, are introduced and their use is entirely optional. The header definition comes with following attributes

name which can be any string, see example below.

headerType with possible values {*mainHeader*, *subHeader*, *userDefined*}

rowNumber specifying the sequence of headers

The `<HeaderRow>` elements, containing the actual header information, are places at the top of the `<Table>`. Their number must be equal the number of `<Header>` elements as the defined in the `<Definition>`. Their sequence is indicated with the `order` attribute and must be in sync with the `rowNumber` values.

```
<IndividualCovariates>
    <ColumnMapping>
        <ds:ColumnRef columnIdRef="wt"/>
        <ct:SymbRef blkIdRef="cm1" symbIdRef="W"/>
    </ColumnMapping>
    <ds:DataSet>
        <ds:Definition>
            <ds:Header name="1stheader" headerType="mainHeader" rowNumber="1"/>
            <ds:Header name="header" headerType="subHeader" rowNumber="2"/>
            <ds:Header name="AddHeader" headerType="userDefined" rowNumber="3"/>
            <ds:Column columnId="ID" columnType="id" valueType="string" columnNum="1"/>
            <ds:Column columnId="ARM" columnType="arm" valueType="id" columnNum="2"/>
            <ds:Column columnId="BSA" columnType="covariate" valueType="real" columnNum="3"/>
        </ds:Definition>
        <ds:Table>
            <ds:HeaderRow order="1">
                <ct:String>ID_main</ct:String>
                <ct:String>ARM_main_header</ct:String>
                <ct:String>BSA_main_header</ct:String>
            </ds:HeaderRow>
            <ds:HeaderRow order="2">
                <ct:String>ID_sub_header</ct:String>
                <ct:String>ARM_sub_header</ct:String>
                <ct:String>BSA_sub_header</ct:String>
            </ds:HeaderRow>
            <ds:HeaderRow order="3">
                <ct:String>other_info</ct:String>
                <ct:String>other_info</ct:String>
                <ct:String>other_info</ct:String>
            </ds:HeaderRow>
            <ds:Row><ct:String>1</ct:String><ct:Id>arm1</ct:Id><ct:NA/></ds:Row>
            <ds:Row><ct:String>2</ct:String><ct:Id>arm1</ct:Id><ct:Real>60.0</ct:Real></ds:Row>
            <ds:Row><ct:String>3</ct:String><ct:Id>arm1</ct:Id><ct:Real>93.2</ct:Real></ds:Row>
            <ds:Row><ct:String>4</ct:String><ct:Id>arm1</ct:Id><ct:Real>85.7</ct:Real></ds:Row>
            <ds:Row><ct:String>5</ct:String><ct:Id>arm1</ct:Id><ct:Real>78.3</ct:Real></ds:Row>
            <!-- SNIP -->
            <ds:Row><ct:String>33</ct:String><ct:Id>arm1</ct:Id><ct:Real>94.1</ct:Real></ds:Row>
        </ds:Table>
    </ds:DataSet>
</IndividualCovariates>
```

6.4 Regressor support

6.4.1 Support in ≤ 0.6 versions

Already since version 0.3 (introduced in April 2014), PharmML accounts for regressors if they are coming from datasets, by means of `columnType` attribute which can be specified as `reg` for regressors (time-varying covariates) or `covariate` for time-constant covariates.

Another type of regressor support has been the `<LookupTable>` (also developed already for 0.3 version), which allows implementation of

- concentration data to be coupled with a PD model. The PK data is available as a lookup table, i.e. measurement records for which the underlying PK model is unknown or not essential.
- *minimal model*-type models used frequently in diabetes require the coupling of insulin input as discrete measurements to the glucose/insulin homeostasis model. Also here the data is coming in form of a lookup table, usually with two columns, one for time and the other for the dependent variable.

The support comes with an build-in list of interpolation algorithm types (not the algorithms of course) to choose from such as

- *nearest, constant, linear, spline, chip, cubic* plus any user-defined algorithms, see [19] for detailed description and examples. We introduce now one new type, the *last value* used by Monolix.

On the other hand, the common `<Variable>` type can already (again already since 0.3 version) play a role of a regressor, e.g. one could simply define variable 'C' with interpolation type with respect to a variable of modellers choice, e.g.

```
<ct:Variable symbolType="real" symbId="C">
    <ct:Assign>
        <ct:Interpolation>
            <ct:Algorithm>spline</ct:Algorithm>
            <ct:InterpIndepVar>
                <ct:SymbRef symbIdRef="t"/>
            </ct:InterpIndepVar>
        </ct:Interpolation>
    </ct:Assign>
</ct:Variable>
```

6.4.2 Support in 0.7 version

The issue with the usage of `<Variable>` as regressor it that it cannot be easily recognised as such. Therefore we have introduced an additional and optional attribute

- `regressor` – with possible values *yes*/*no*

which will give e.g. Monolix the required support. This will also allow models such as that described in `http://simulx.webpopix.org/userguide/function-time-regression` to be implemented in PharmML and run with Simulx . This means support for yet another tool which is a valuable extension from the perspective of the DDMoRe platform.

The PharmML looks then almost identical to the previous case if the source of data for C is a data vector in the `<LookupTable>`

```
<ct:Variable symbolType="real" regressor="yes" symbId="C">
    <ct:Assign>
        <!-- details skipped here -->
    </ct:Assign>
</ct:Variable>
```

or even as short as

```
<ct:Variable symbolType="real" regressor="yes" symbId="C"/>
```

if the model for C comes from an external source.

6.5 Conditional distributions

This structure can be used e.g. in covariate or other models. In this particular example the body weight, WT, is distributed differently for female and male subjects:

$$P(WT|SEX) = \begin{cases} \mathcal{N}\big(WT^F_{mean}, WT^F_{variance}\big) & \text{for} \quad SEX == F \\ \mathcal{N}\big(WT^M_{mean}, WT^M_{variance}\big) & \text{for} \quad SEX == M \end{cases}$$

To encode that first the SEX covariate needs to be defined

```
<CovariateModel blkId="cm1">
    <Covariate symbId="SEX">
        <Categorical>
            <Category catId="M"/>
            <Category catId="F"/>
        </Categorical>
    </Covariate>
```

and then the WT covariate and its distribution with the piece-wise structure

```
    <Covariate symbId="WT">
        <Continuous>
            <Distribution>
                <Piecewise>
                    <Piece xmlns="http://www.pharmml.org/pharmml/0.6/Maths">
                        <ProbOnto name="Normal2">
                            <mdef:Parameter name="mean">
                                <ct:Assign>
                                <ct:SymbRef symbIdRef="WT_F_mean"/>
                            </ct:Assign>
                        </mdef:Parameter>
                        <mdef:Parameter name="var">
                            <ct:Assign>
                                <ct:SymbRef symbIdRef="WT_F_variance"/>
                            </ct:Assign>
                        </mdef:Parameter>
                    </ProbOnto>
                    <Condition>
                        <!-- SEX=="F" -->
                        <LogicBinop op="eq">
                            <ct:SymbRef blkIdRef="cm1" symbIdRef="SEX"/>
                            <ct:CatRef catIdRef="F"/>
                        </LogicBinop>
                    </Condition>
                </Piece>
                <Piece xmlns="http://www.pharmml.org/pharmml/0.6/Maths">
                    <ProbOnto name="Normal2">
                        <mdef:Parameter name="mean">
                            <ct:Assign>
                                <ct:SymbRef symbIdRef="WT_M_mean"/>
                            </ct:Assign>
                        </mdef:Parameter>
                        <mdef:Parameter name="var">
                            <ct:Assign>
                                <ct:SymbRef symbIdRef="WT_M_variance"/>
                            </ct:Assign>
                        </mdef:Parameter>
                    </ProbOnto>
                    <Condition>
                        <!-- SEX=="M" -->
                        <LogicBinop op="eq">
                            <ct:SymbRef blkIdRef="cm1" symbIdRef="SEX"/>
                            <ct:CatRef catIdRef="M"/>
                        </LogicBinop>
                    </Condition>
                </Piece>
            </Piecewise>
        </Distribution>
    </Continuous>
</Covariate>
```

Note that we use ProbOnto's normal distribution, `Normal2`, which is parameterised with mean and variance.

6.6 Minor changes/bug fixing

- Covariate model, `<CovariateModel>`, comes with an explicit assignment option – any expression can be encoded here providing a missing so far option to define new covariates out of existing ones. For example, the definition a new covariate based on exiting ones, such as $C = A + B$ reads in PharmML as follows

```
<CovariateModel blkId="cm1">
    <Covariate symbId="A">
        <!-- detailes skipped here -->
    </Covariate>
    <Covariate symbId="B">
        <!-- detailes skipped here -->
    </Covariate>
    <Covariate symbId="C">
        <Continuous>
            <ct:Assign>
                <math:Equation>
                    <math:Binop op="plus">
                        <ct:SymbRef symbIdRef="A"/>
                        <ct:SymbRef symbIdRef="B"/>
                    </math:Binop>
                </math:Equation>
            </ct:Assign>
        </Continuous>
    </Covariate>
</CovariateModel>
```

- `<InitialEstimate>`, `<LowerBound>` and `<UpperBound>` definition has been extended to allow for initial assignments required for parameters (vectors or matrices) of multivariate distributions, as the following code snippet shows

```
<ParameterEstimation>
    <ct:SymbRef symbIdRef="SIGMA_POP_P"/>
    <InitialEstimate>
        <ct:Matrix matrixType="Any">
            <ct:MatrixRow>
                <ct:RowIndex><ct:Int>1</ct:Int></ct:RowIndex>
                <ct:Real>1</ct:Real>
                <ct:Real>0.1</ct:Real>
            </ct:MatrixRow>
            <ct:MatrixRow>
                <ct:RowIndex><ct:Int>2</ct:Int></ct:RowIndex>
                <ct:Real>0.1</ct:Real>
                <ct:Real>1</ct:Real>
            </ct:MatrixRow>
        </ct:Matrix>
    </InitialEstimate>
</ParameterEstimation>
```

  in which the covariance matrix of a multivariate normal distribution is assigned initial values.

- `<NumberCounts>` element allows to identify the dependent variable in discrete count data models, usually denoted as k. This was implemented mistakingly with as a parameter in 0.6 examples. These examples have all been corrected.

- Added several `columnType` attribute values to be used in SO
  - *indivParameter* to identify the individual parameters, such as *CL*.
  - *popParameter* to identify the population parameters, such as *POP_CL*
  - *randEffect* to identify the random effects, such as *ETA_CL*.
  - *residual* to identify the residuals, such as *CWRES*.
  - *strataVariable* to identify the stratification variables, such as *STRATA_DOSE*.
  - *statPrecision* to identify the measures and quantities expressing statistical precision, such as *SE*, *RSE* or *ETA_SHRINKAGE_CL*.
  - *structParameter* to identify structural parameters.
  - *varParameter* to identify variability parameters.

- The majority of the PK macro examples, released with the 0.6 spec and schema contained wrong references to structural model within the `<TargetMapping>`, e.g.

```
<ColumnMapping>
    <ds:ColumnRef columnIdRef="ADM"/>
    <ds:TargetMapping blkIdRef="sm1">
        <ds:Map dataSymbol="1" admNumber="1"/>
    </ds:TargetMapping>
</ColumnMapping>
```

  Instead of `sm1`, it should be the value assigned to *blkIdRef* of the according structural model, e.g. `sm12` in the *PKmacros_advan12.xml* example. Thanks to Henrik for spotting that.

- A number of typos/bugs were found in remaining 0.6 examples, see next section for a detailed description.

6.7 Debugging with libPharmML

The examples used in PharmML 0.6 have been fixed thanks to the validation procedure of the version 0.4.1 of libPharmML. This last version includes validation of symbol, object and column references, additionally to dataset validation. Those examples contained some unresolved references, sometimes due to a missing `blkIdRef` attribute value, an undefined referred parameter or a typo. Some of the dataset rows also used a data type incompatible with the column definition.

The validation can be performed within the non-Java tools using the stand-alone validator that can be run from the command line. The jar is available on sourceforge: `https://sourceforge.net/projects/libpharmml.ddmore.p/files/Stand-alone%20validator/`. A new validator version for PharmML 0.7 will be provided soon.

Appendix A

Selected distributions in ProbOnto knowledge base

The full first version of ProbOnto will be released soon. Here we provide a brief overview of a part of its content for the most frequently used distributions and/or alternative parameterisations (38 out of about 55).

The according plots have been performed using the R code stored in ProbOnto and provided for each distribution for which it's available.

Symbols used Some of the symbols used in definitions of the functions and quantities listed in the subsequent sections are collected here with references

- Beta function, B
  `http://mathworld.wolfram.com/BetaFunction.html`
  `http://en.wikipedia.org/wiki/Beta_function`
- Regularized incomplete Beta function, I_p, I_{1-p}
  `http://mathworld.wolfram.com/RegularizedBetaFunction.html`
  `http://en.wikipedia.org/wiki/Beta_function#Incomplete_beta_function`
- Error function, erf
  `http://mathworld.wolfram.com/Erf.html`
  `https://en.wikipedia.org/wiki/Error_function`
- Floor function, $\lfloor k \rfloor$
  `http://mathworld.wolfram.com/FloorFunction.html`
  `https://en.wikipedia.org/wiki/Floor_and_ceiling_functions`
- Gamma function, Γ
  `http://mathworld.wolfram.com/GammaFunction.html`
  `http://en.wikipedia.org/wiki/Gamma_function`
- Lower incomplete gamma function, γ
  `http://mathworld.wolfram.com/IncompleteGammaFunction.html`
  `https://en.wikipedia.org/wiki/Incomplete_gamma_function`
- Multivariate Gamma function, Γ_p
  `https://en.wikipedia.org/wiki/Multivariate_gamma_function`
- Iverson bracket, $[x = i]$
  `http://mathworld.wolfram.com/IversonBracket.html`
  `http://en.wikipedia.org/wiki/Iverson_bracket`
- Linear span, $span$
  `http://mathworld.wolfram.com/VectorSpaceSpan.html`
  `https://en.wikipedia.org/wiki/Linear_span`
- (Generalized) Hypergeometric function, $({}_pF_q)$, ${}_2F_1$
  `http://mathworld.wolfram.com/HypergeometricFunction.html`
  `http://en.wikipedia.org/wiki/Hypergeometric_function`

Bernoulli

name	Bernoulli (ID: 0000000)
type	discrete
variate	k, scalar
support	$k \in \{0, 1\}$

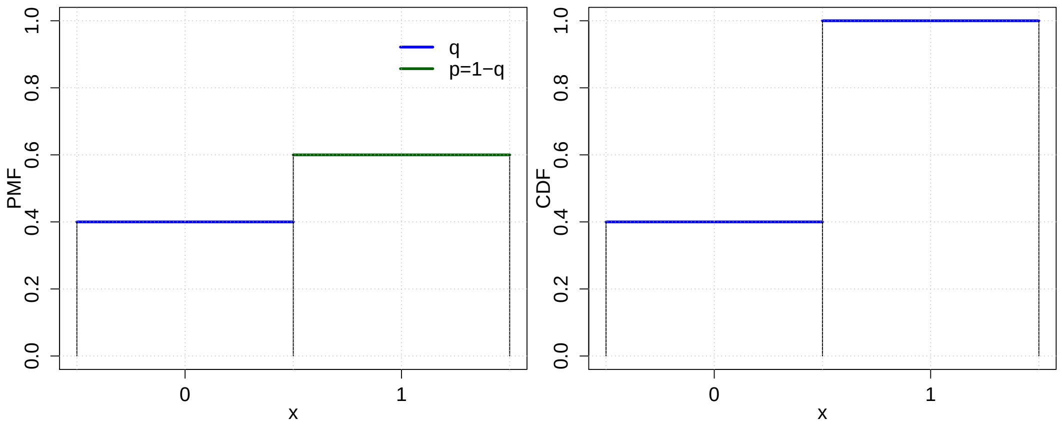


Figure A.1: PDF and CDF of the Bernoulli distribution for p=0.6.

Parameter: probability

name	probability
type	scalar
symbol	p
definition	$0 < p < 1, p \in R$

Functions

PMF

$$\begin{cases} q = (1-p) & \text{for } k = 0 \\ p & \text{for } k = 1 \end{cases}$$

CDF

$$\begin{cases} 0 & \text{for } k < 0 \\ q & \text{for } 0 \le k < 1 \\ 1 & \text{for } k \ge 1 \end{cases}$$

Beta

name	Beta (ID: 0000009)
type	continuous
variate	x, scalar
support	$x \in (0, 1)$

Parameter: alpha

name	shape
type	scalar
symbol	α
definition	$\alpha > 0$

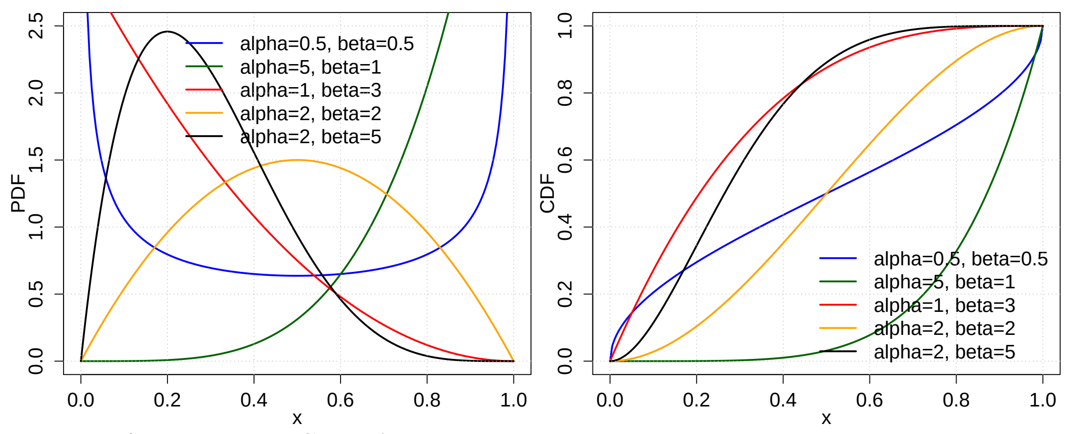


Figure A.2: PDF and CDF of the Beta distribution plotted using the provided R-code.

Parameter: beta

name	shape
type	scalar
symbol	β
definition	$\beta > 0$

Functions

PDF

$$\frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha,\beta)}$$

PDF in R

```
(x^(alpha-1)*(1-x)^(beta-1))/beta(alpha,beta)
```

CDF

$$I_x(\alpha,\beta)$$

CDF in R

```
Rbeta(x, alpha, beta)
```

Binomial

name	Binomial (ID: 0000020)
type	discrete
variate	k, scalar
support	$k \in \{0,\ldots,n\}$

Parameter: numberOfFailures

name	number of trials
type	scalar
symbol	n
definition	$n \in N, n \geq 0$

Parameter: probability

name	success probability in each trial
type	scalar
symbol	p
definition	$p \in [0,1]$

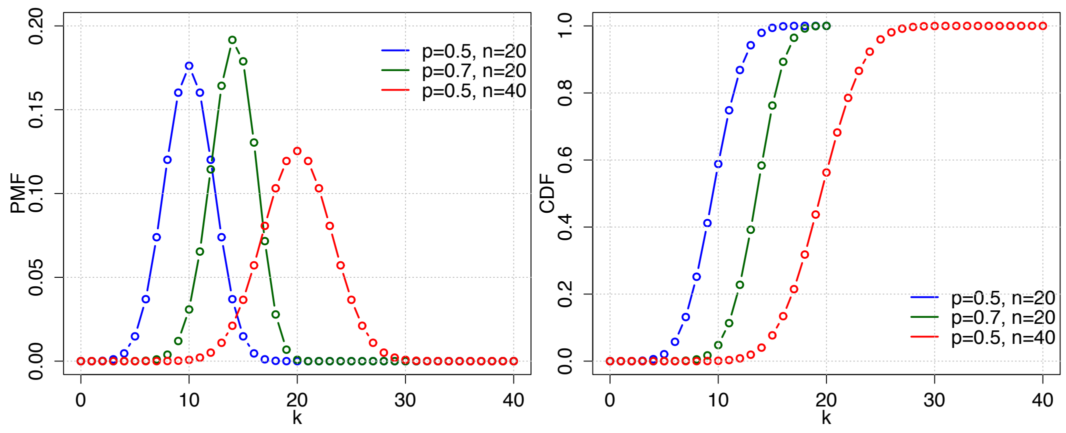


Figure A.3: PMF and CDF of the Binomial distribution plotted using the provided R-code.

Functions

PMF

$$\binom{n}{k} p^k (1-p)^{n-k}$$

PMF in R

```
choose(n,k) * p^k*(1-p)^(n-k)
```

CDF

$$I_{1-p}(n-k, 1+k)$$

CDF in R

```
Rbeta(1-p, n-k, 1+k)
```

CategoricalNonordered

name	Categorical Nonordered (ID: 0000040)
type	discrete
variate	x, scalar
support	$x \in \{1, \ldots, k\}$

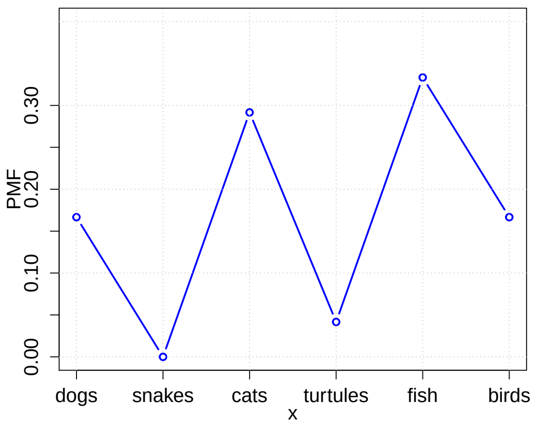


Figure A.4: An example for a PMF of the Categorical Unordered distribution for pets owned by children, with probabilities for each category { 0.17, 0, 0.29, 0.042, 0.33, 0.167}. In this case the CDF is undefined.

Parameter: categoryProb

name	category probabilities
type	vector
symbol	$p_1, \ldots, p_k$
definition	$0 \leq p_i \leq 1, \Sigma p_i = 1$

Functions

PMF

$$p(x = i) = p_i$$

CDF

$$undefined$$

CategoricalOrdered

name	Categorical Ordered (ID: 0000031)
type	discrete
variate	x, scalar
support	$x \in \{1, \ldots, k\}$

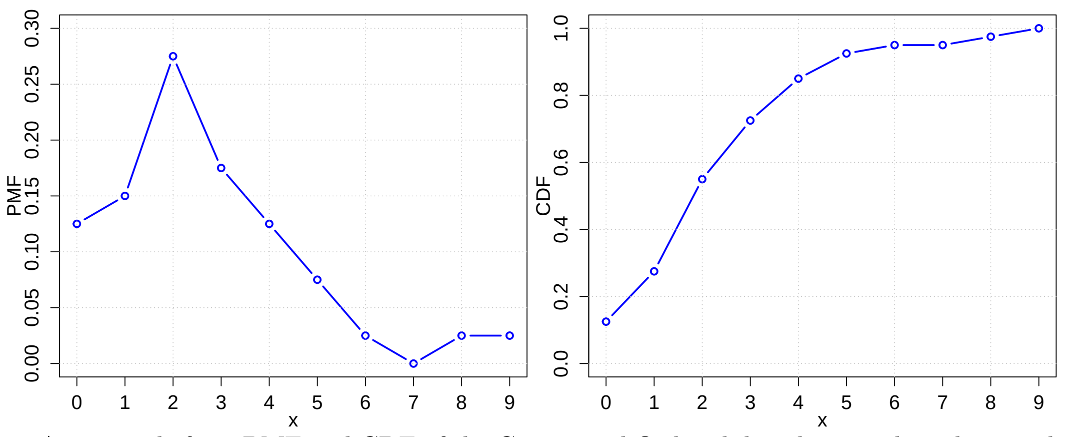


Figure A.5: An example for a PMF and CDF of the Categorical Ordered distribution plotted using the provided R-code. Using data from `https://onlinecourses.science.psu.edu/stat414/node/13`

Parameter: categoryProb

name	category probabilities
type	vector
symbol	$p_1, \ldots, p_k$
definition	$0 \leq p_i \leq 1, \Sigma p_i = 1$

Functions

PMF

$$p(x = i) = p_i$$

CDF

$$\begin{cases} 0 & \text{for } x < 1 \\ \sum_{j=1}^{i} p_j & \text{for } x \in [i, i+1) \\ 1 & \text{for } x \geq k \end{cases}$$

Cauchy

name	Cauchy (ID: 0000049)
type	continuous
variate	x, scalar
support	$x \in (-\infty, +\infty)$

Parameter: location

name	location
type	scalar
symbol	x_0
definition	$x_0 \in R$

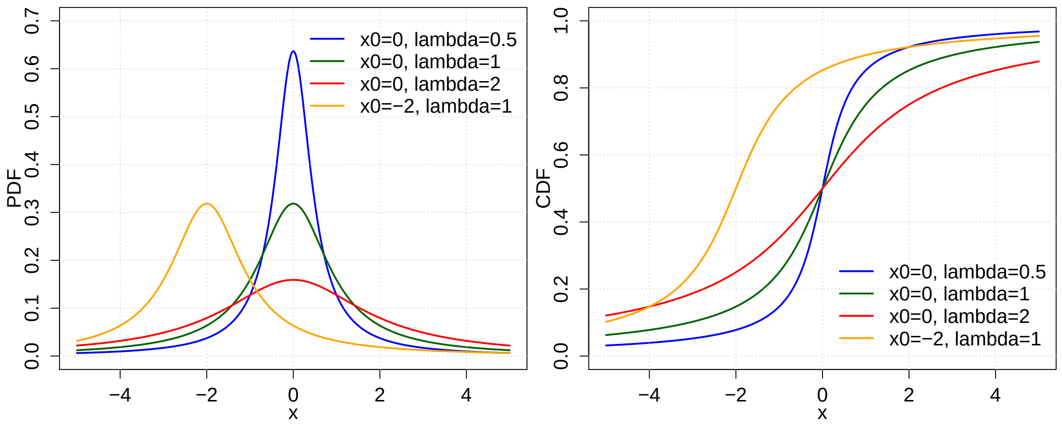


Figure A.6: PDF and CDF of the Cauchy distribution plotted using the provided R-code.

Parameter: scale

name	scale
type	scalar
symbol	γ
definition	$\gamma \in R$

Functions

PDF

$$\frac{1}{\pi\gamma\left[1+\left(\frac{x-x_0}{\gamma}\right)^2\right]}$$

PDF in R

```
1/(pi*gamma*(1 + ((x-x0)^2/gamma^2)))
```

CDF

$$\frac{1}{\pi}\arctan\left(\frac{x-x_0}{\gamma}\right)+\frac{1}{2}$$

CDF in R

```
1/pi * atan((x-x0)/gamma)+1/2
```

ChiSquared

name	Chi-squared (ID: 0000060)
type	continuous
variate	x, scalar
support	$x \in [0, +\infty)$

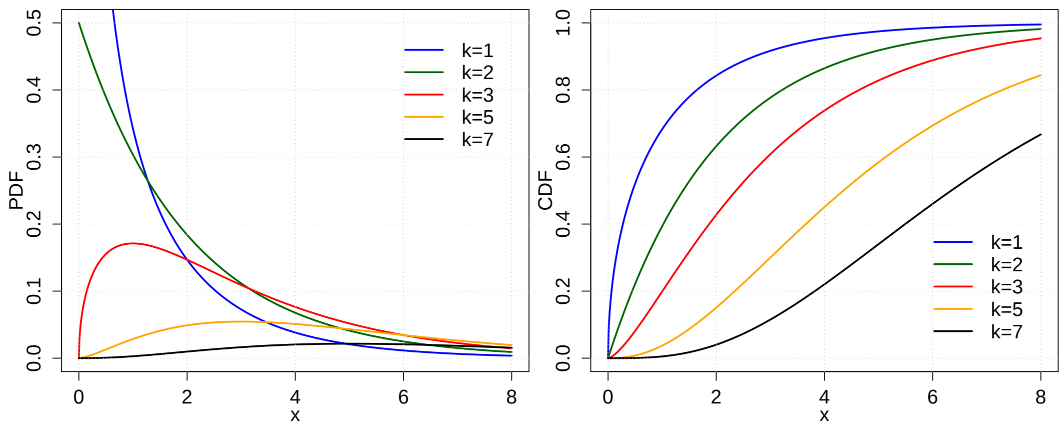


Figure A.7: PDF and CDF of the Chi-squared distribution plotted using the provided R-code.

Parameter: degreesOfFreedom

name	degrees of freedom
type	scalar
symbol	k
definition	$k \in N$

Functions

PDF

$$\frac{1}{2^{\frac{k}{2}}\Gamma\left(\frac{k}{2}\right)}\, x^{\frac{k}{2}-1}e^{-\frac{x}{2}}$$

PDF in R

```
1/( 2^k/2 * gamma(k/2) ) *  x^(k/2-1) * exp(-x/2)
```

CDF

$$\frac{1}{\Gamma\left(\frac{k}{2}\right)}\,\gamma\left(\frac{k}{2},\frac{x}{2}\right)$$

CDF in R

```
1/gamma(k/2) * Igamma(k/2,x/2)
```

Dirichlet

name	Dirichlet (ID: 0000076)
type	continuous
variate	x, vector
support	$x_1, \cdots, x_K$ where $x_i \in [0,1]$ *and* $\sum_{i=1}^{K} x_i = 1$

Parameter: concentration

name	concentration
type	vector
symbol	$\alpha_1, \cdots, \alpha_K$
definition	$\alpha_1, \cdots, \alpha_K, \alpha_i > 0$

Functions

PDF

$$\frac{1}{B(\alpha)}\prod_{i=1}^{K} x_i^{\alpha_i-1}\text{where}\quad B(\alpha)=\frac{\prod_{i=1}^{K}\Gamma(\alpha_i)}{\Gamma(\sum_{i=1}^{K}\alpha_i)}\text{where}\quad \alpha=(\alpha_1,\ldots,\alpha_K)$$

CDF

$$-$$

Exponential

name	Exponential (ID: 0000085)
type	continuous
variate	x, scalar
support	$x \in [0, +\infty)$

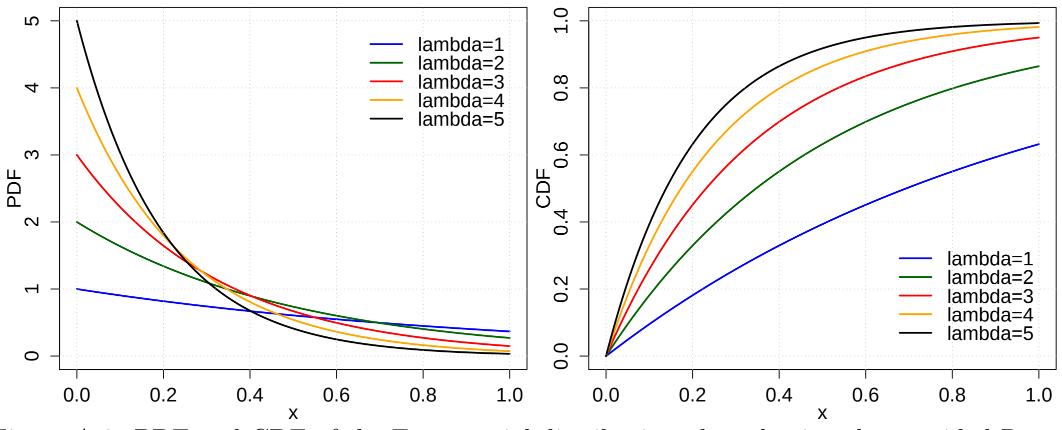


Figure A.8: PDF and CDF of the Exponential distribution plotted using the provided R-code.

5 **Parameter: rate**

name	rate or inverse scale
type	scalar
symbol	λ
definition	$\lambda > 0$

Functions

PDF

$$\lambda e^{-\lambda x}$$

PDF in R

```
lambda*exp(-lambda*x)
```

CDF

$$1 - e^{-\lambda x}$$

10 **CDF in R**

```
1-exp(-lambda*x)
```

F

name	F (ID: 0000096)
type	continuous
variate	x, scalar
support	$x \in [0, +\infty)$

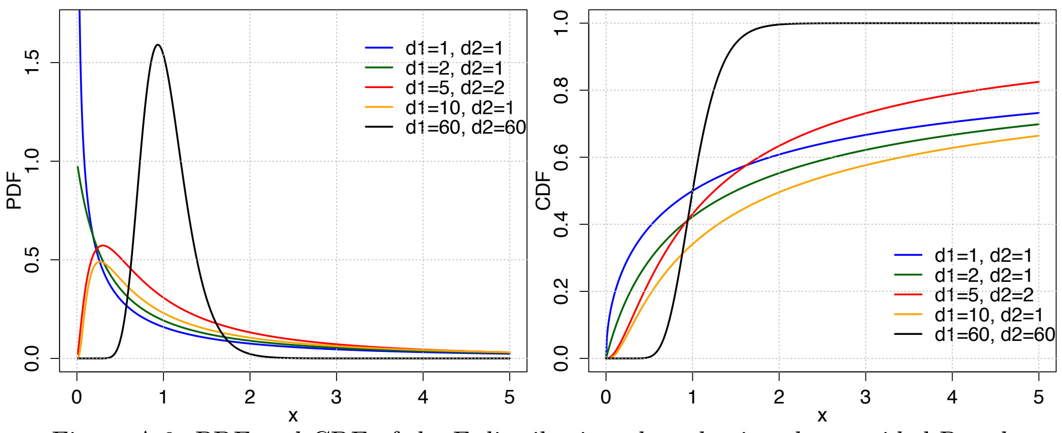


Figure A.9: PDF and CDF of the F distribution plotted using the provided R-code.

Parameter: numerator

name	degree of freedom
type	scalar
symbol	d_1
definition	$d_1 > 0$

Parameter: denominator

name	degree of freedom
type	scalar
symbol	d_2
definition	$d_2 > 0$

Functions

PDF

$$\frac{\sqrt{\frac{(d_1 x)^{d_1} d_2^{d_2}}{(d_1 x + d_2)^{d_1+d_2}}}}{x B\left(\frac{d_1}{2}, \frac{d_2}{2}\right)}$$

PDF in R

```
sqrt((d1*x)^d1*d2^(d2) / (d1*x+d2)^(d1+d2) ) / (x*beta(d1/2,d2/2))
```

CDF

$$I_{\frac{d_1 x}{d_1 x + d_2}}\left(\frac{d_1}{2}, \frac{d_2}{2}\right)$$

CDF in R

```
Rbeta(d1*x / (d1*x + d2), d1/2, d2/2)
```

Gamma

name	Gamma (ID: 0000106)
type	continuous
variate	x, scalar
support	$x \in (0, +\infty)$

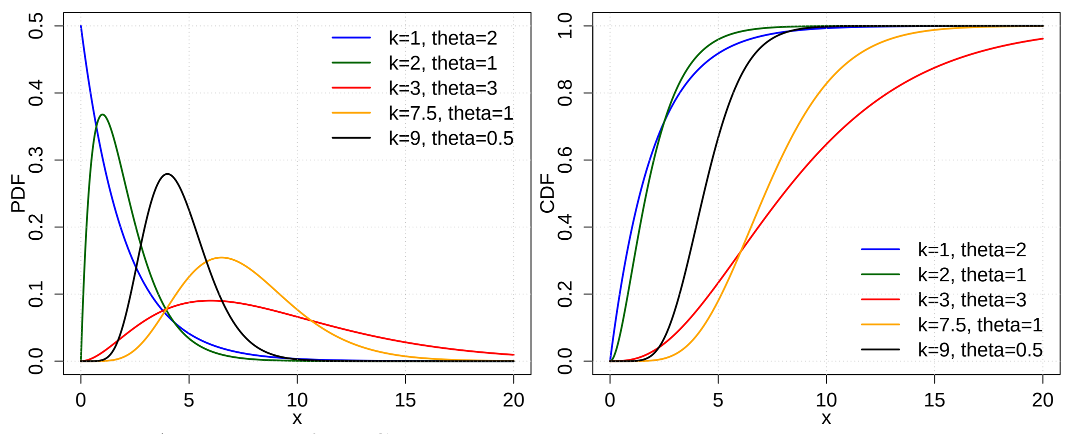


Figure A.10: PMF of the Gamma distribution plotted using the provided R-code.

Parameter: shape

name	shape
type	scalar
symbol	k
definition	$k > 0$

Parameter: scale

name	scale
type	scalar
symbol	θ
definition	$\theta > 0$

Functions

PDF

$$\frac{1}{\Gamma(k)\theta^k} x^{k-1} e^{-\frac{x}{\theta}}$$

PDF in R

```
1 / (gamma(k) * theta^k) * x^(k-1) * exp(-x/theta)
```

CDF

$$\frac{1}{\Gamma(k)} \gamma\left(k, \frac{x}{\theta}\right)$$

CDF in R

```
1/gamma(k) * Igamma(k,x/theta,lower=T)
```

GeneralizedGamma1

name	Generalized Gamma 1 (ID: 0000123)
type	continuous
variate	x, scalar
support	$x \in (0, +\infty)$

Parameter: scale

name	scale
type	scalar
symbol	a
definition	$a > 0$

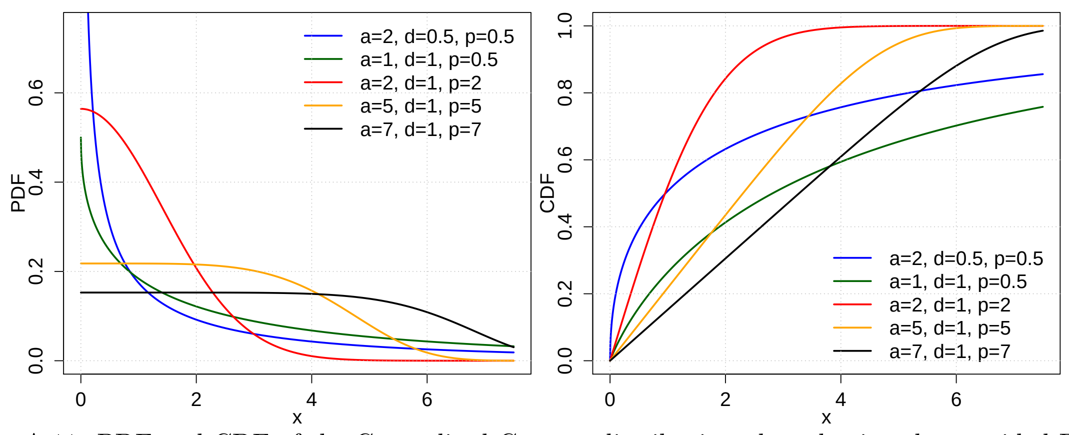


Figure A.11: PDF and CDF of the Generalised Gamma distribution plotted using the provided R-code.

Parameter: shape1

name	shape
type	scalar
symbol	d
definition	$d > 0$

Parameter: shape2

name	shape
type	-
symbol	p
definition	$p > 0$

Functions

PDF

$$\frac{p/a^d}{\Gamma(d/p)} x^{d-1} e^{-(x/a)^p}$$

PDF in R

```
p/a^d/gamma(d/p) * x^(d-1) * exp(-(x/a)^p)
```

CDF

$$\frac{\gamma(d/p, (x/a)^p)}{\Gamma(d/p)}$$

CDF in R

```
Igamma(d/p, (x/a)^p, lower=T) / gamma(d/p)
```

GeneralizedPoisson

name	Generalized Poisson (ID: 0000145)
type	discrete
variate	k, scalar
support	$k \in \{0, 1, 2, 3, \dots\}$

Parameter: rate

name	Poisson intensity
type	scalar
symbol	λ
definition	$\lambda \in R, \lambda > 0$

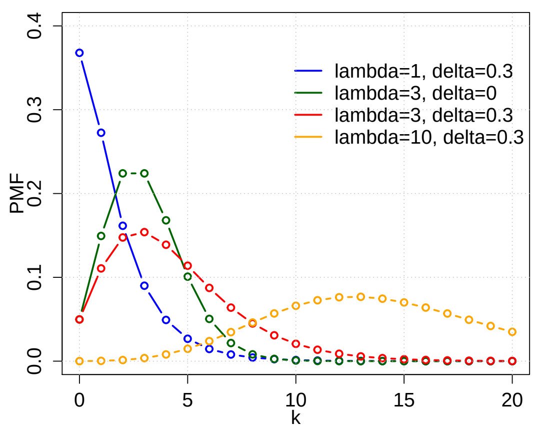


Figure A.12: PMF of the Generalised Poisson distribution plotted using the provided R-code.

Parameter: dispersion

name	dispersion
type	scalar
symbol	δ
definition	$\max(-1, -\lambda/4) < \delta < 1$

Functions

PMF

$$\frac{\lambda(\lambda + k\delta)^{k-1} \times e^{-\lambda-k\delta}}{k!}$$

PMF in R

```
(lambda*(lambda+k*delta)^(k-1) * exp(-lambda-k*delta)) / factorial(k)
```

CDF

–

Gompertz

name	Gompertz (ID: 0000155)
type	continuous
variate	x, scalar
support	$x \in (-\infty, +\infty)$

Parameter: shape

name	shape
type	scalar
symbol	η
definition	$\eta > 0$

Parameter: scale

name	scale
type	scalar
symbol	b
definition	$b > 0$

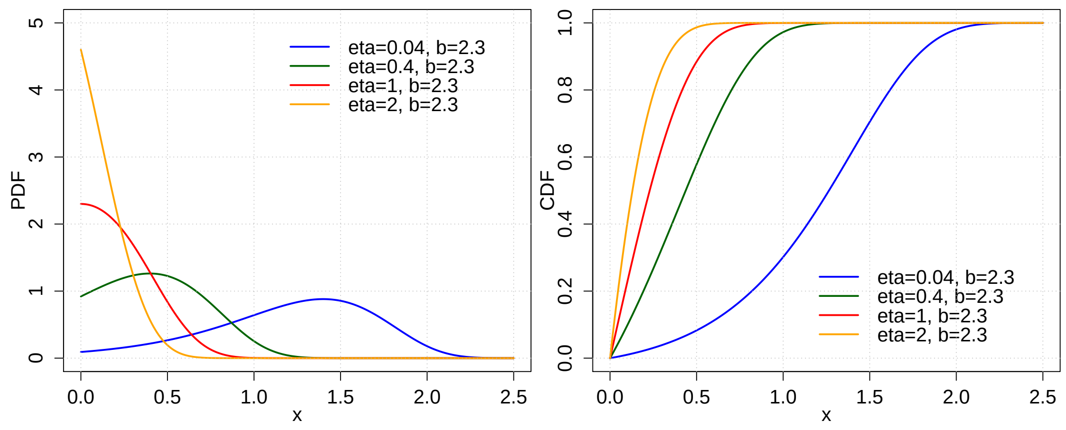


Figure A.13: PMF of the Gompertz distribution plotted using the provided R-code.

Functions

PDF

$$b\eta e^{bx} e^{\eta} \exp\left(-\eta e^{bx}\right)$$

PDF in R

```
b*eta*exp(b*x)*exp(eta)*exp(-eta*exp(b*x))
```

CDF

$$1 - \exp\left(-\eta\left(e^{bx} - 1\right)\right)$$

CDF in R

```
1-exp(-eta*(exp(b*x)-1))
```

InverseWishart

name	Inverse-Wishart (ID: 0000197)
type	continuous
variate	X, matrix
support	$X(p \times p)$ – positive-definite matrix

Parameter: scaleMatrix

name	scale matrix
type	matrix
symbol	Ψ
definition	$\Psi > 0$, positive-definite matrix

Parameter: degreesOfFreedom

name	degrees of freedom
type	scalar
symbol	ν
definition	$\nu > p - 1, \nu \in R$

Functions

PDF

$$\frac{|\Psi|^{\frac{\nu}{2}}}{2^{\frac{\nu p}{2}}\Gamma_p\left(\frac{\nu}{2}\right)} |X|^{-\frac{\nu+p+1}{2}} e^{-\frac{1}{2}\mathrm{tr}(\Psi X^{-1})}$$

CDF

–

Laplace1

name	Laplace 1 (ID: 0000207)
type	continuous
variate	x, scalar
support	$x \in (-\infty, +\infty)$

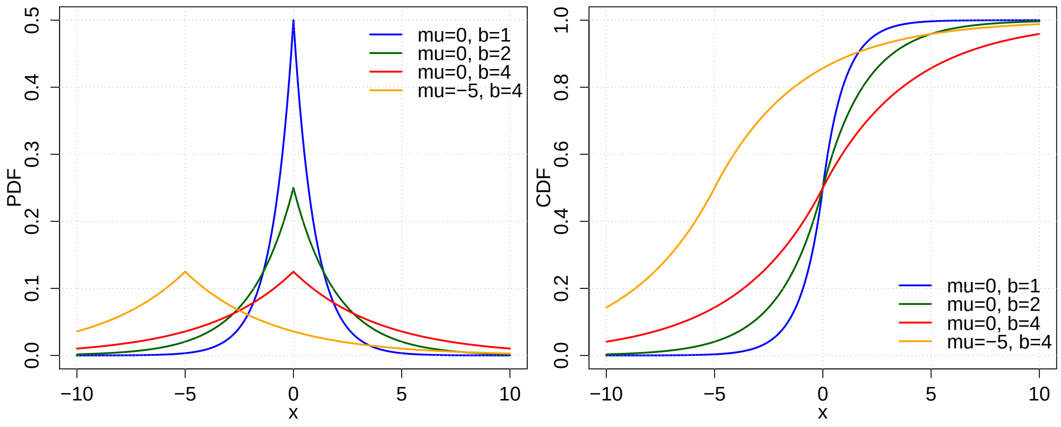


Figure A.14: PMF of the Laplace1 distribution plotted using the provided R-code.

Parameter: location

name	location
type	scalar
symbol	μ
definition	$\mu \in R$

Parameter: scale

name	scale
type	scalar
symbol	b
definition	$b > 0, b \in R$

Functions

PDF

$$\frac{1}{2\,b} \exp\left(-\frac{|x-\mu|}{b}\right)$$

PDF in R

```
1/(2*b) * exp(-abs(x-mu)/b)
```

CDF

$$\begin{cases} \frac{1}{2}\exp\left(\frac{x-\mu}{b}\right) & \text{if } x < \mu \\ 1 - \frac{1}{2}\exp\left(-\frac{x-\mu}{b}\right) & \text{if } x \geq \mu \end{cases}$$

CDF in R

```
1/2 * exp( (x-mu)/b ) for x < mu
1- 1/2 * exp( -(x-mu)/b ) for x >= mu
```

LogNormal1

name	Log-Normal 1 (ID: 0000227)
type	continuous
variate	x, scalar
support	$x \in (0, +\infty)$

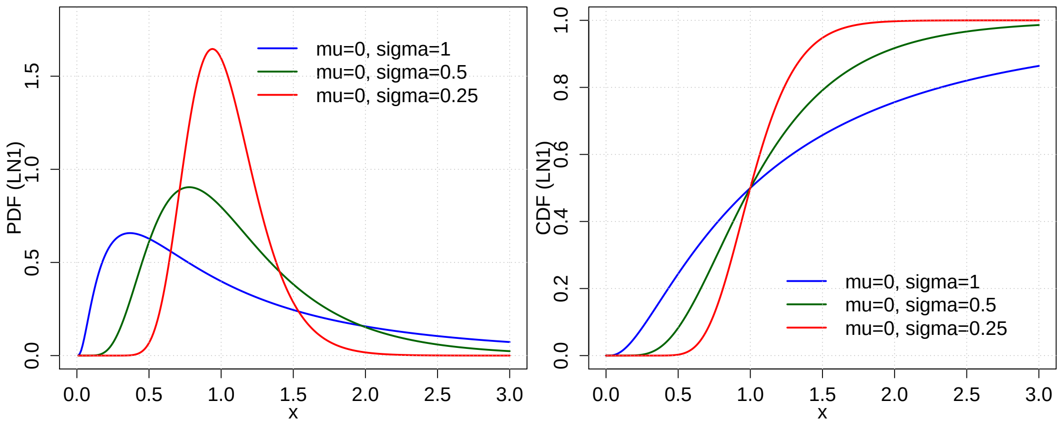


Figure A.15: PDF and CDF of the Log-Normal distribution, LN1, plotted using the provided R-code.

Parameter: meanLog

name	mean of log(x)
type	scalar
symbol	μ
definition	$\mu \in R$

Parameter: stdevLog

name	shape
type	scalar
symbol	σ
definition	$\sigma > 0$

Functions

PDF

$$\frac{1}{x\sigma\sqrt{2\pi}}\, e^{-\frac{(\log x-\mu)^2}{2\sigma^2}}$$

PDF in R

```
1/(x*sigma*sqrt(2*pi)) * exp((-(log(x)-mu)^2)/(2*sigma^2))
```

CDF

$$\frac{1}{2}+\frac{1}{2}\,\mathrm{erf}\left[\frac{\log x-\mu}{\sqrt{2}\sigma}\right]$$

CDF in R

```
erf <- function(x) 2 * pnorm(x * sqrt(2)) - 1
1/2 + 1/2 *erf( (log(x)-mu)/(sqrt(2)*sigma) )
```

LogNormal2

name	Log-Normal 2 (ID: 0000238)
type	continuous
variate	x, scalar
support	$x \in (0, +\infty)$

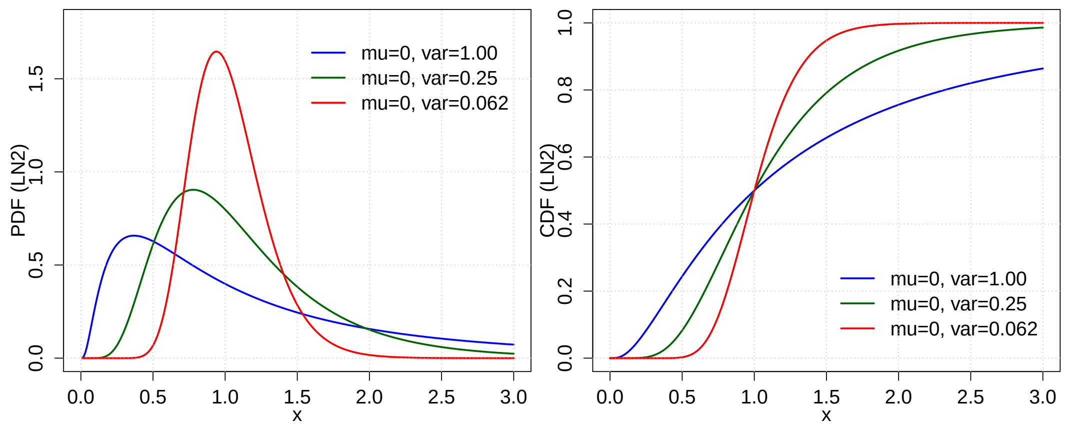


Figure A.16: PDF and CDF of the Log-Normal distribution, LN1, plotted using the provided R-code.

Parameter: meanLog

name	mean of log(x)
type	scalar
symbol	μ
definition	$\mu \in R$

Parameter: varLog

name	shape
type	scalar
symbol	v
definition	$v > 0$

Functions

PDF

$$\frac{1}{x\sqrt{v}\sqrt{2\pi}}\, e^{-\frac{(\log x-\mu)^2}{2v}}$$

PDF in R

```
1/(x*sqrt(v)*sqrt(2*pi)) * exp(-(ln(x)-mu)^2/(2*v))
```

CDF

$$\frac{1}{2}+\frac{1}{2}\operatorname{erf}\left[\frac{\log x-\mu}{\sqrt{2\,v}}\right]$$

CDF in R

```
erf <- function(x) 2 * pnorm(x * sqrt(2)) - 1
1/2 + 1/2*erf( (log(x)-mu) / (sqrt(2*v)))
```

LogNormal3

name	Log-Normal 3 (ID: 0000248)
type	continuous
variate	x, scalar
support	$x \in (0, +\infty)$

Parameter: median

name	median / geometric mean
type	scalar
symbol	m
definition	$m > 0$

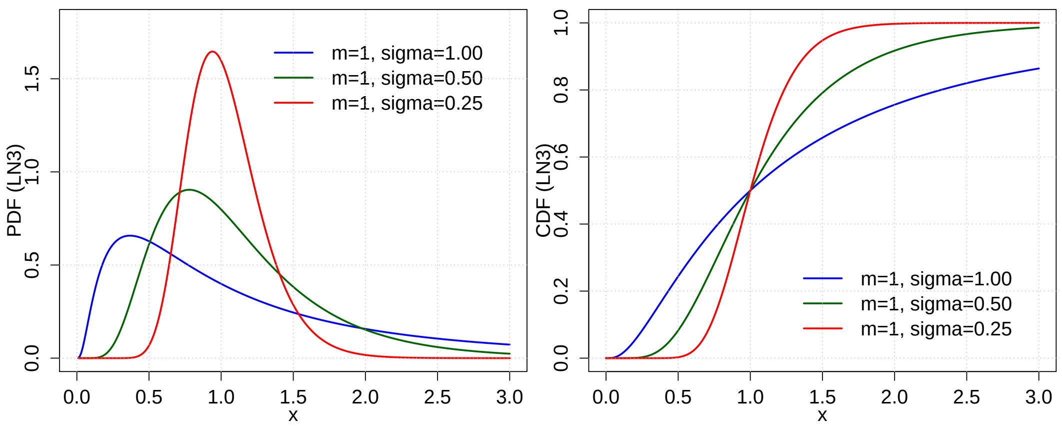


Figure A.17: PDF and CDF of the Log-Normal distribution, LN3, plotted using the provided R-code.

Parameter: stdevLog

name	shape
type	scalar
symbol	σ
definition	$\sigma > 0$

Functions

PDF

$$\frac{1}{x\sigma\sqrt{2\pi}}\, e^{-\frac{[\log(x/m)]^2}{2\sigma^2}}$$

PDF in R

```
1/(x*sigma*sqrt(2*pi)) * exp(-(log(x/m))^2 / (2*sigma^2))
```

CDF

$$\frac{1}{2} + \frac{1}{2}\,\mathrm{erf}\Big[\frac{\log x - \log m}{\sqrt{2}\sigma}\Big]$$

CDF in R

```
erf <- function(x) 2 * pnorm(x * sqrt(2)) - 1
1/2 + 1/2 * erf( (log(x)-log(m)) / (sqrt(2)*sigma) )
```

LogNormal4

name	Log-Normal 4 (ID: 0000253)
type	continuous
variate	x, scalar
support	$x \in (0, +\infty)$

Parameter: median

name	median / geometric mean
type	scalar
symbol	m
definition	$m > 0$

Parameter: coefVar

name	coefficient of variation
type	scalar
symbol	cv
definition	$cv > 0$

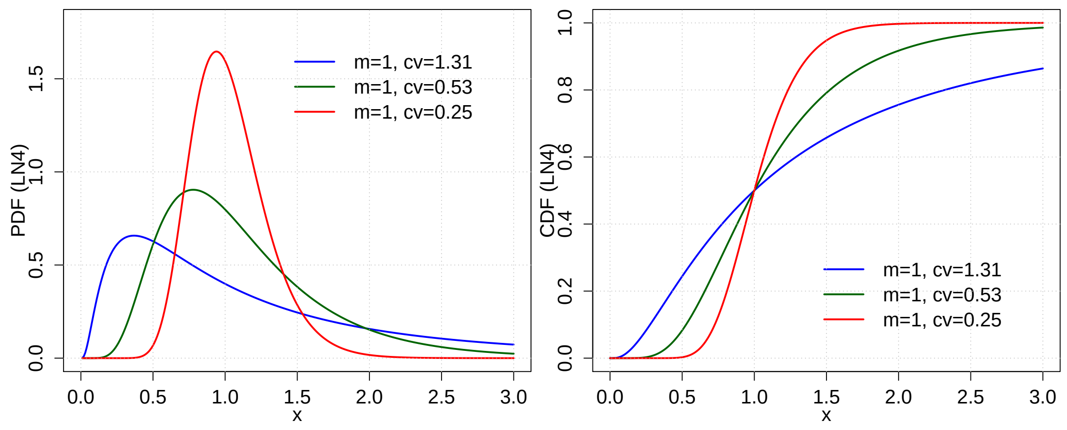


Figure A.18: PDF and CDF of the Log-Normal distribution, LN4, plotted using the provided R-code.

Functions

PDF

$$\frac{1}{x\sqrt{\log(cv^2+1)}\sqrt{2\pi}}\; e^{-\frac{[\log(x/m)]^2}{2\log(cv^2+1)}}$$

PDF in R

```
1/(x*sqrt(log(cv^2+1))*sqrt(2*pi)) * exp( -(log(x/m))^2 / (2*log(cv^2+1)) )
```

CDF

$$\frac{1}{2}+\frac{1}{2}\,\mathrm{erf}\Big[\frac{\log x-\log m}{\sqrt{2}\sqrt{\log(cv^2+1)}}\Big]$$

CDF in R

```
erf <- function(x) 2 * pnorm(x * sqrt(2)) - 1
1/2 + 1/2*erf( (log(x)-log(m)) / (sqrt(2*log(cv^2+1))) )
```

LogNormal5

name	Log-Normal 5 (ID: 0000258)
type	continuous
variate	x, scalar
support	$x \in (0, +\infty)$

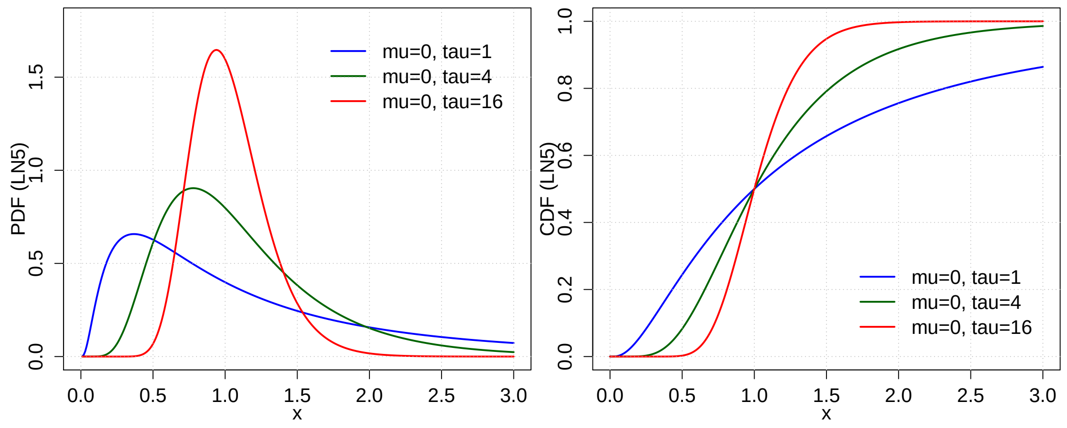


Figure A.19: PDF and CDF of the Log-Normal distribution, LN5, plotted using the provided R-code.

Parameter: meanLog

name	mean of log(x)
type	scalar
symbol	μ
definition	$\mu \in R$

Parameter: precision

name	precision
type	scalar
symbol	τ
definition	$\tau > 0$

Functions

PDF

$$\sqrt{\frac{\tau}{2\pi}}\frac{1}{x}e^{-\frac{\tau}{2}(\log x-\mu)^2}$$

PDF in R

```
sqrt(tau/(2*pi)) * (1/x) * exp(-(tau/2)*(log(x)-mu)^2)
```

CDF

$$\frac{1}{2}+\frac{1}{2}\,\text{erf}\Big[\frac{\ln x-\mu}{\sqrt{2/\tau}}\Big]$$

CDF in R

```
erf <- function(x) 2 * pnorm(x * sqrt(2)) - 1
1/2 + 1/2*erf( (log(x)-mu) / sqrt(2/tau) )
```

MixtureDistribution

name	Mixture Distribution (ID: 0000268)
type	continuous
variate	–, -
support	–

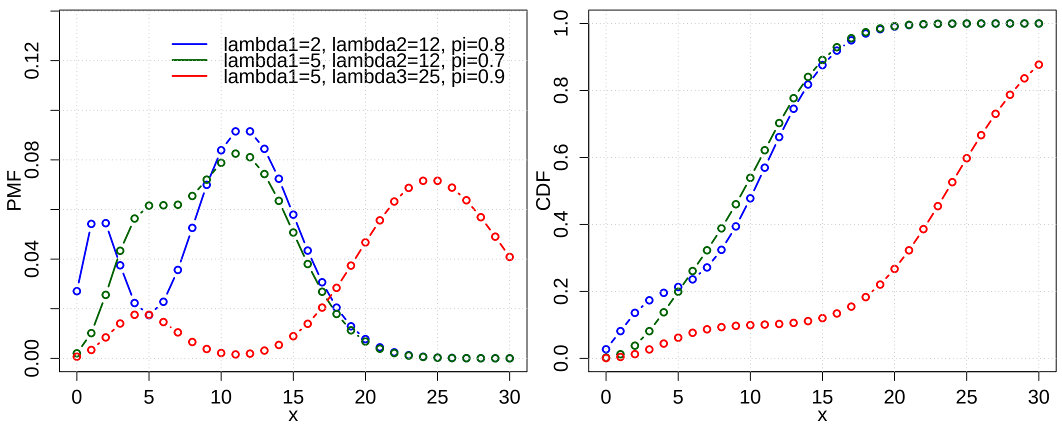


Figure A.20: PMF and CDF of the Mixture Poisson distribution plotted using the formula for for various values as shown in the legend of the left plot. The PMF reads: $(1-\pi)\,\lambda_1^k/k!\,\exp(-\lambda_1)+\pi\,\lambda_2^k/k!\,\exp(-\lambda_2)$. The CDF reads: $(1-\pi)\,\Gamma(\lfloor k+1,\lambda_1\rfloor)/\lfloor k\rfloor!+\pi\,\Gamma(\lfloor k+1,\lambda_2\rfloor)/\lfloor k\rfloor!$.

Parameter: weight

name	mixing coefficients
type	vector
symbol	$\pi_1, \ldots, \pi_k$
definition	$\Sigma_{i=1}^{K} \pi_i = 1; 0 \leq \pi_i \leq 1$

Functions

PDF

$$f(x; \pi, \theta) = \sum_{i=1}^{K} \pi_i \; p_i(x; \theta_i) \text{ where } p_i(x; \theta_i) \text{ the PDF of the } i^{th} \text{ component with parameters } \theta_i$$

PMF

$$f(x; \pi, \theta) = \sum_{i=1}^{K} \pi_i \; p_i(x; \theta_i) \text{ where } p_i(x; \theta_i) \text{ the PMF of the } i^{th} \text{ component with parameters } \theta_i$$

CDF

$$-$$

Multinomial

name	Multinomial (ID: 0000273)
type	discrete
variate	X, vector
support	$X_i \in \{0, \ldots, n\}, \Sigma X_i = n$

Parameter: numberOfTrials

name	number of trials
type	scalar
symbol	n
definition	$n > 0, n \in N$

Parameter: probabilityOfSuccess

name	event probabilities
type	vector
symbol	$p_1, \ldots, p_k$
definition	$p_1, \ldots, p_k, \Sigma p_i = 1$

Functions

PMF

$$\frac{n!}{x_1! \cdots x_k!} p_1^{x_1} \cdots p_k^{x_k}$$

CDF

$$-$$

MultivariateNormal1

name	Multivariate Normal 1 (ID: 0000278)
type	continuous
variate	x, vector
support	$x \in \mu + \text{span}(\Sigma) \subseteq R^k$

Parameter: mean

name	location
type	vector
symbol	μ
definition	$\mu \in R^k$

Parameter: covarianceMatrix

name	covariance matrix
type	matrix
symbol	Σ
definition	$\Sigma \in R^{k \times k}$

Functions

PDF

$$(2\pi)^{-\frac{k}{2}}|\Sigma|^{-\frac{1}{2}}\,e^{-\frac{1}{2}(x-\mu)'\Sigma^{-1}(x-\mu)}$$

CDF

no analytic expression

NegativeBinomial1

name	Negative Binomial 1 (ID: 0000035)
type	discrete
variate	k, scalar
support	$k \in \{0, 1, 2, 3, \dots\}$

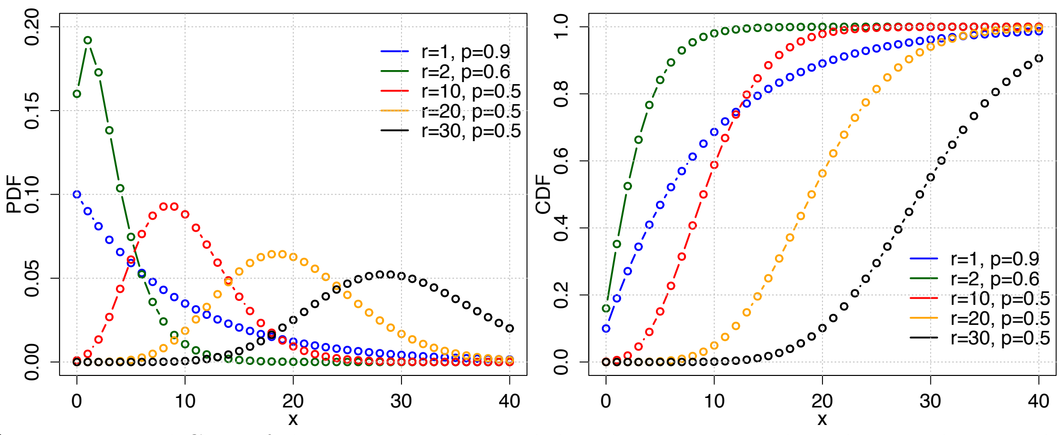


Figure A.21: PMF and CDF of the Negative Binomial distribution, NB1, plotted using the provided R-code.

Parameter: numberOfFailures

name	number of failures
type	scalar
symbol	r
definition	$r > 0, r \in N$

Parameter: probability

name	success probability
type	scalar
symbol	p
definition	$p \in [0, 1]$

Functions

PMF

$$\binom{k+r-1}{k}(1-p)^r p^k$$

PMF in R

```
choose(k+r-1,k)*(1-p)^r*p^k
```

CDF

$$1 - I_p(k+1, r)$$

CDF in R

```
1 - Rbeta(p, k+1, r)
```

NegativeBinomial2

name	Negative Binomial 2 (ID: 0000044)
type	discrete
variate	k, scalar
support	$k \in \{0, 1, 2, 3, \dots\}$

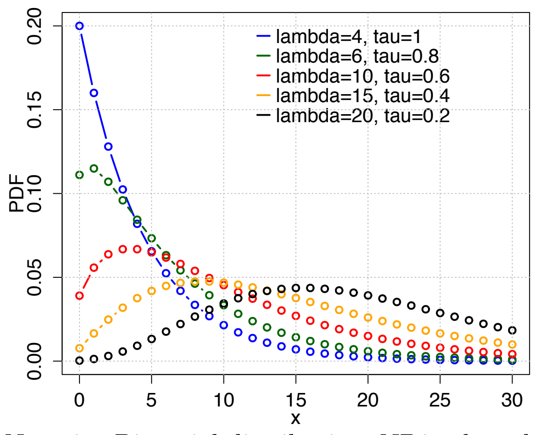


Figure A.22: PMF of the Negative Binomial distribution, NB2, plotted using the provided R-code.

Parameter: rate

name	Poisson intensity
type	scalar
symbol	λ
definition	$\lambda \in R, \lambda > 0$

Parameter: overdispersion

name	overdispersion
type	scalar
symbol	τ
definition	$\tau \in R$

Functions

PMF

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

PMF in R

```
gamma(k+1/tau)/(factorial(k)*gamma(1/tau))*1/(1+tau*lambda)^(1/tau)*(lambda/(1/tau + lambda))^k
```

CDF

$$-$$

Normal1

name	Normal 1 (ID: 0000064)
type	continuous
variate	x, scalar
support	$x \in R$

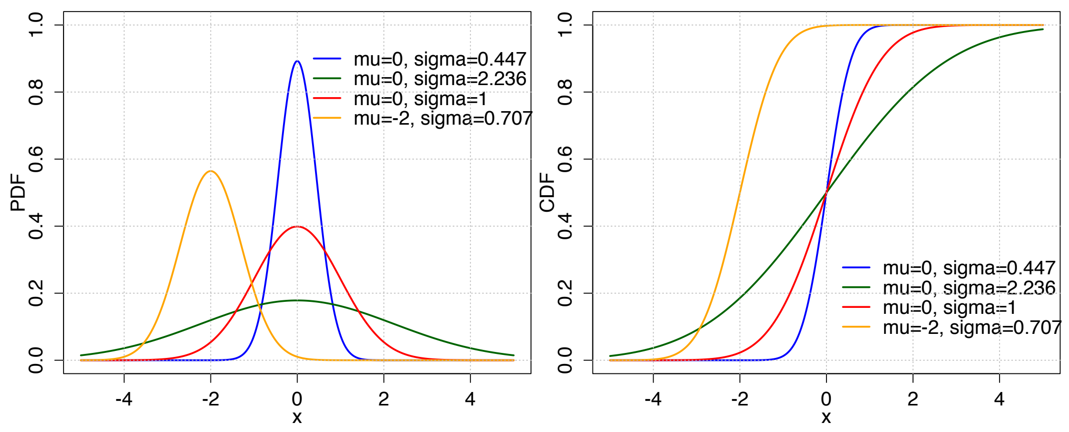


Figure A.23: PDF and CDF of the Normal distribution, N1, plotted using the provided R-code.

Parameter: mean

name	mean
type	scalar
symbol	μ
definition	$\mu \in R$

Parameter: stdev

name	standard deviation
type	scalar
symbol	σ
definition	$\sigma > 0$

Functions

PDF

$$\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

PDF in R

```
1/(sigma*sqrt(2*pi))*exp(-(x-mu)^2/(2*sigma^2))
```

CDF

$$\frac{1}{2}\left[1+\operatorname{erf}\left(\frac{x-\mu}{\sigma\sqrt{2}}\right)\right]$$

CDF in R

```
erf <- function(x) 2 * pnorm(x * sqrt(2)) - 1
1/2 * (1 + erf((x-mu)/(sigma*sqrt(2))))
```

Normal2

name	Normal 2 (ID: 0000071)
type	continuous
variate	x, scalar
support	$x \in R$

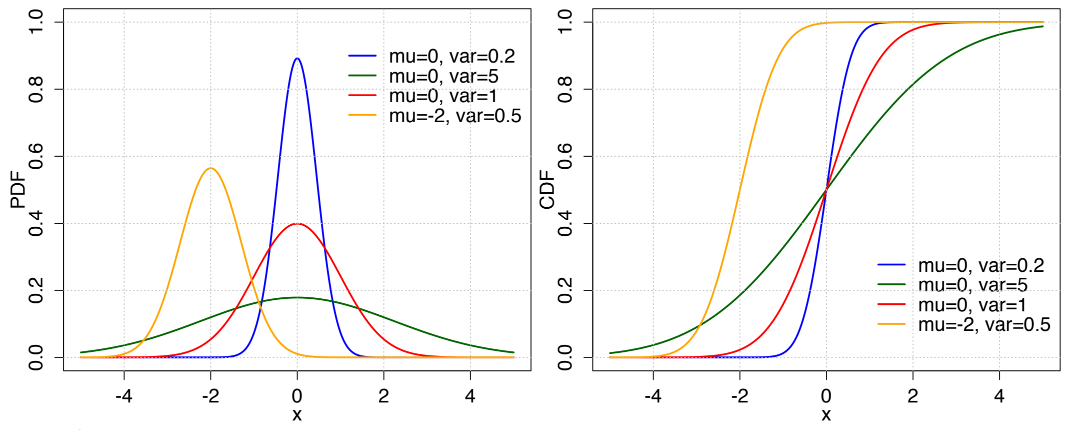


Figure A.24: PDF and CDF of the Normal distribution, N2, plotted using the provided R-code.

Parameter: mean

name	mean
type	scalar
symbol	μ
definition	$\mu \in R$

Parameter: var

name	variance
type	scalar
symbol	v
definition	$v > 0$

Functions

PDF

$$\frac{1}{\sqrt{v}\sqrt{2\pi}}e^{-\frac{(x-\mu)^2}{2*v}}$$

PDF in R

```
1/(sqrt(var)*sqrt(2*pi))*exp(-(x-mu)^2/(2*var))
```

CDF

$$\frac{1}{2}\left[1+\operatorname{erf}\left(\frac{x-\mu}{\sqrt{v}\sqrt{2}}\right)\right]$$

CDF in R

```
erf <- function(x) 2 * pnorm(x * sqrt(2)) - 1
1/2 * (1 + erf((x-mu)/(sqrt(var)*sqrt(2))))
```

Normal3

name	Normal 3 (ID: 0000080)
type	continuous
variate	x, scalar
support	$x \in R$

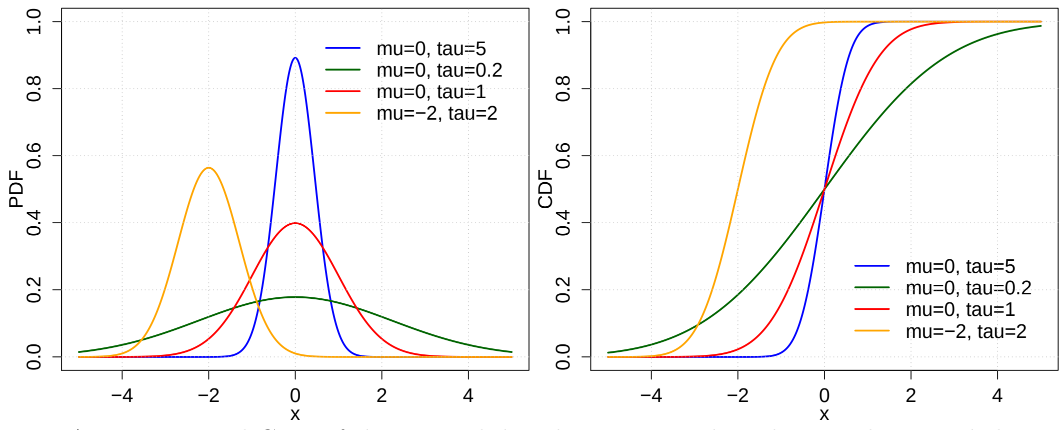


Figure A.25: PDF and CDF of the Normal distribution, N3, plotted using the provided R-code.

Parameter: mean

name	mean
type	scalar
symbol	μ
definition	$\mu \in R$

Parameter: precision

name	precision
type	scalar
symbol	τ
definition	$\tau > 0$

Functions

PDF

$$\sqrt{\frac{\tau}{2\pi}} e^{-\frac{\tau}{2}(x-\mu)^2}$$

PDF in R

```
sqrt(tau/(2*pi))*exp(-tau/2*(x-mu)^2)
```

CDF

$$\frac{1}{2}\left[1 + \operatorname{erf}\left(\frac{x-\mu}{\sqrt{1/\tau}\sqrt{2}}\right)\right]$$

CDF in R

```
erf <- function(x) 2 * pnorm(x * sqrt(2)) - 1
1/2*(1+erf((x-mu)/(sqrt(1/tau)*sqrt(2))))
```

Poisson

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

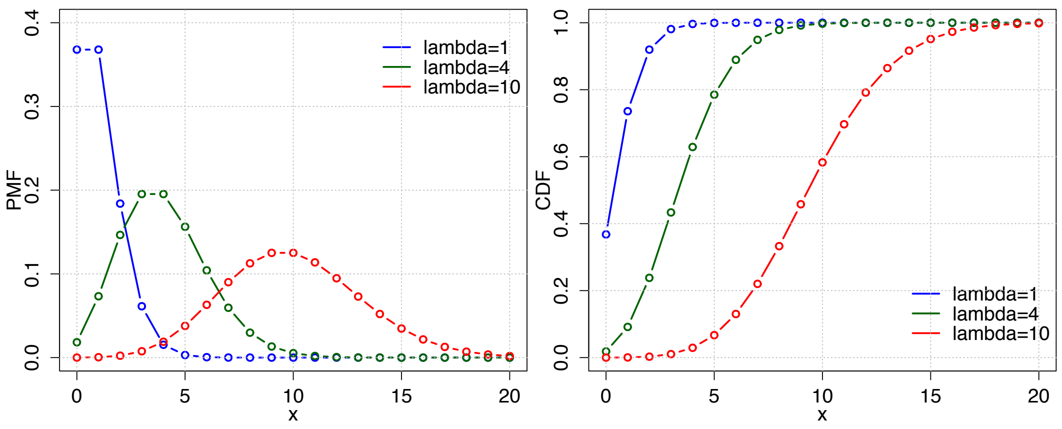


Figure A.26: PMF and CDF of the Poisson distribution plotted using the provided R-code.

Parameter: rate

name	Poisson intensity
type	scalar
symbol	λ
definition	$\lambda \in R, \lambda > 0$

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`.

Note There are many different names for the Poisson parameter, such as

- *shape* in Distributome, `http://www.distributome.org/V3/Distributome.xml.html`
- *mean* or *variance* in Forbes et al. 2010, [7]
- *rate* in UncertML, `http://www.uncertml.org/distributions/poisson`
- *scale* in Leemis et al. 2008, `http://www.math.wm.edu/~leemis/chart/UDR/PDFs/Poisson.pdf`

As we have followed the UncertML naming convention, *rate* is the code name to be used with ProbOnto.

StandardNormal

name	Standard Normal (ID: 0000129)
type	continuous
variate	x, scalar
support	$x \in R$

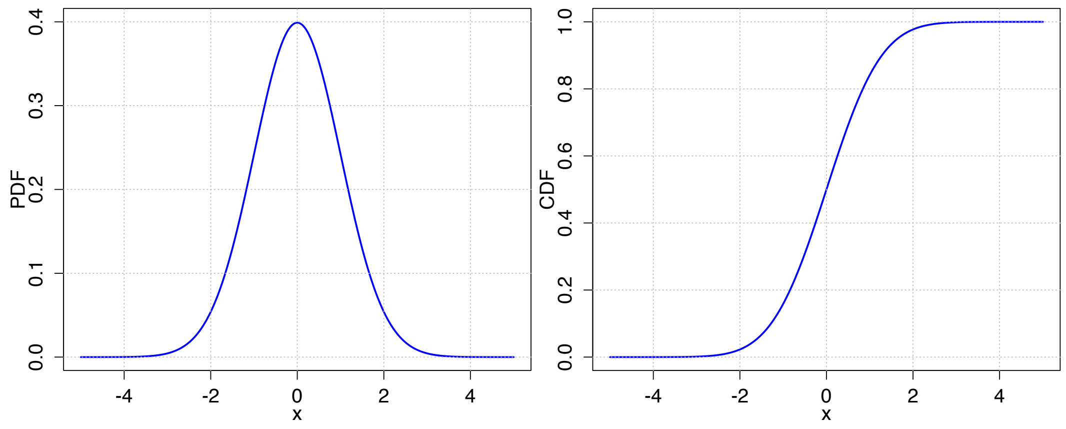


Figure A.27: PDF and CDF of the Standard Normal distribution plotted using the provided R-code.

Parameter: mean

name	mean
type	scalar
symbol	μ
definition	$\mu = 0$

Parameter: stdev

name	standard deviation
type	scalar
symbol	σ
definition	$\sigma = 1$

Functions

PDF

$$\frac{e^{-\frac{1}{2}x^2}}{\sqrt{2\pi}}$$

PDF in R

```
1/(sqrt(2*pi))*exp(-x^2/2)
```

CDF

$$\frac{1}{2}\left[1 + \text{erf}\left(\frac{x}{\sqrt{2}}\right)\right]$$

CDF in R

```
erf <- function(x) 2 * pnorm(x * sqrt(2)) - 1
1/2 * (1 + erf(x/(sqrt(2))))
```

StandardUniform

name	Standard Uniform (ID: 0000150)
type	continuous
variate	x, scalar
support	$x \in [0, 1]$

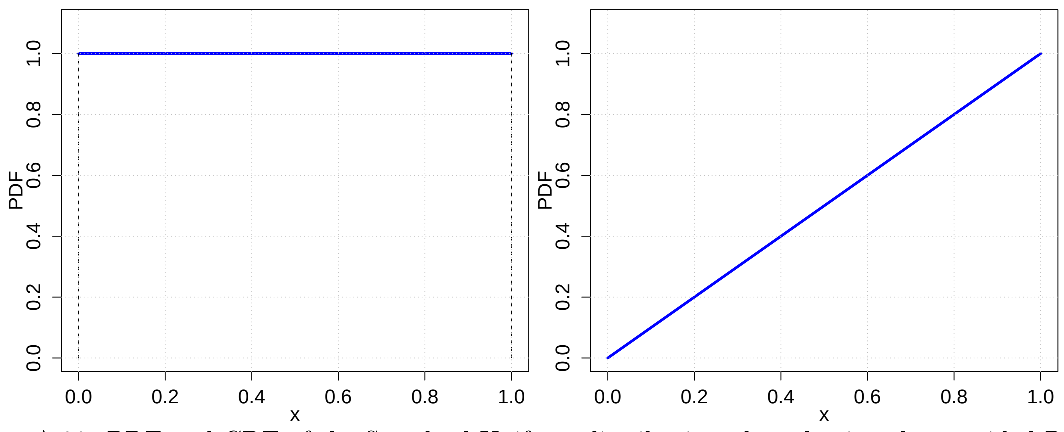


Figure A.28: PDF and CDF of the Standard Uniform distribution plotted using the provided R-code.

Parameter: minimum

name	minimum
type	scalar
symbol	a
definition	$a = 0$

Parameter: maximum

name	maximum
type	scalar
symbol	b
definition	$b = 1$

Functions

PDF

$$1$$

PDF in R

```
1
```

CDF

$$x$$

CDF in R

```
x
```

StudentT

name	Student’s t-distribution (ID: 0000141)
type	continuous
variate	x, scalar
support	$x \in (-\infty, +\infty)$

Parameter: degreesOfFreedom

name	degrees of freedom
type	scalar
symbol	ν
definition	$\nu > 0, \nu \in R$

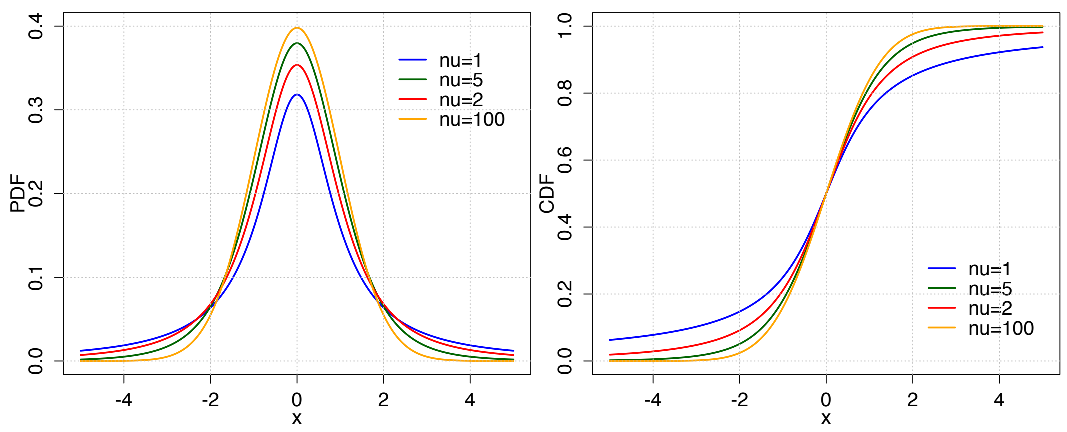


Figure A.29: PDF and CDF of the StudentT distribution plotted using the provided R-code.

Functions

PDF

$$\frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\sqrt{\nu\pi}\,\Gamma\left(\frac{\nu}{2}\right)}\left(1+\frac{x^2}{\nu}\right)^{-\frac{\nu+1}{2}}$$

PDF in R

```
gamma((nu+1)/2)/(sqrt(nu*pi)*gamma(nu/2))*(1+x^2/nu)^(-(nu+1)/2)
```

CDF

$$\frac{1}{2}+x\Gamma\left(\frac{\nu+1}{2}\right)\times\frac{{}_2F_1\left(\frac{1}{2},\frac{\nu+1}{2};\frac{3}{2};-\frac{x^2}{\nu}\right)}{\sqrt{\pi\nu}\,\Gamma\left(\frac{\nu}{2}\right)}$$

CDF in R

```
1/2+x*gamma((nu+1)/2)*hypergeo(1/2,(nu+1)/2,3/2,-x^2/nu)/( sqrt(pi*nu) *gamma(nu/2))
```

5 Uniform

name	Uniform (ID: 0000160)
type	continuous
variate	x, scalar
support	$x \in [a, b]$

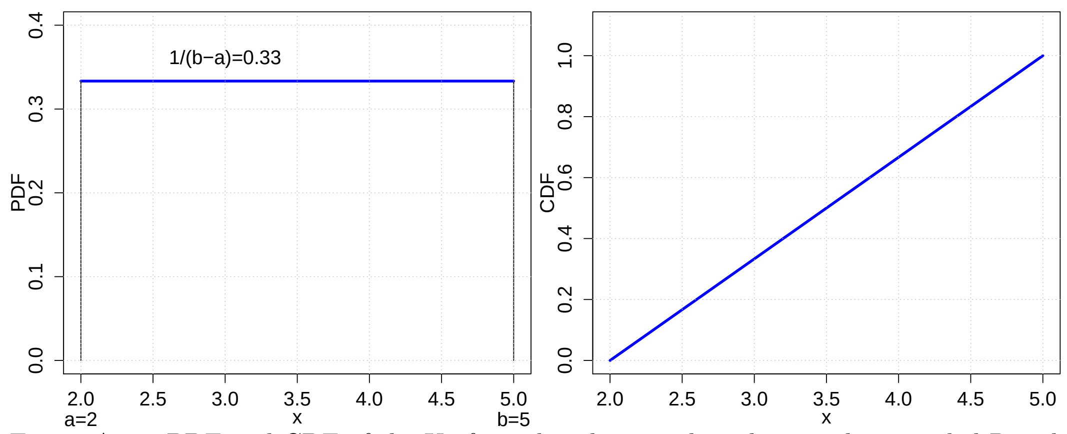


Figure A.30: PDF and CDF of the Uniform distribution plotted using the provided R-code.

Parameter: minimum

name	minimum
type	scalar
symbol	a
definition	$a \in R$

Parameter: maximum

name	maximum
type	scalar
symbol	b
definition	$b \in R, a < b$

Functions

PDF

$$\begin{cases} \frac{1}{b-a} & \text{for } x \in [a,b] \\ 0 & \text{otherwise} \end{cases}$$

PDF in R

```
1/(b-a)
```

CDF

$$\begin{cases} 0 & \text{for } x < a \\ \frac{x-a}{b-a} & \text{for } x \in [a,b) \\ 1 & \text{for } x \geq b \end{cases}$$

CDF in R

```
(x-a)/(b-a)
```

Weibull1

name	Weibull 1 (ID: 0000192)
type	continuous
variate	x, scalar
support	$x \in [0, +\infty)$

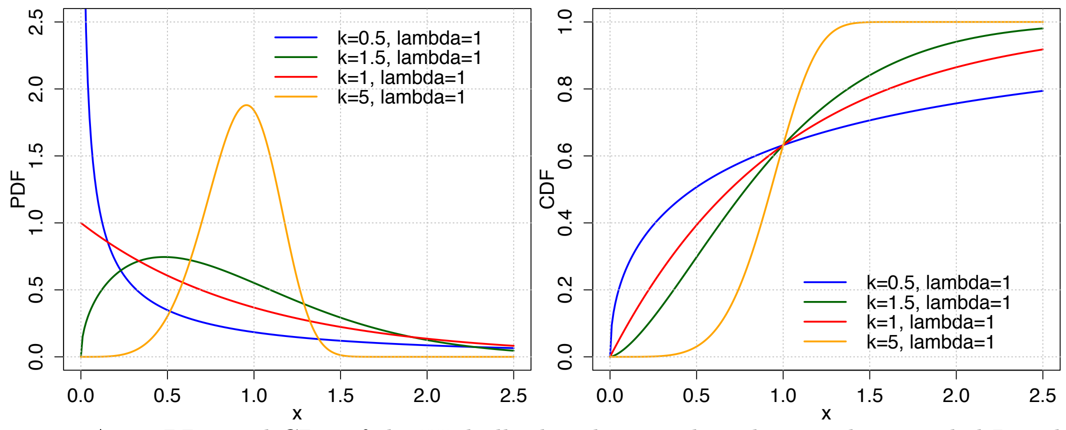


Figure A.31: PDF and CDF of the Weibull1 distribution plotted using the provided R-code.

Parameter: scale

name	scale
type	scalar
symbol	λ
definition	$\lambda \in (0, +\infty)$

Parameter: shape

name	shape
type	scalar
symbol	k
definition	$k \in (0, +\infty)$

Functions

PDF

$$\begin{cases} \frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{k-1} e^{-(x/\lambda)^k} & x \geq 0 \\ 0 & x < 0 \end{cases}$$

PDF in R

```
k/lambda * (x/lambda)^(k-1) * exp(-(x/lambda)^k)
```

CDF

$$\begin{cases} 1 - e^{-(x/\lambda)^k} & x \geq 0 \\ 0 & x < 0 \end{cases}$$

CDF in R

```
1- exp(-(x/lambda)^k)
```

Wishart1

name	Wishart 1 (ID: 0000212)
type	continuous
variate	X, matrix
support	$X(p \times p)$ – positive definite matrix

Parameter: scaleMatrix

name	scale matrix
type	matrix
symbol	V
definition	$V > 0, p \times p$ – positive definite matrix

Parameter: degreesOfFreedom

name	degrees of freedom
type	scalar
symbol	n
definition	$n > p - 1$

Functions

PDF

$$\frac{|X|^{\frac{n-p-1}{2}} e^{-\frac{\mathrm{tr}(V^{-1}X)}{2}}}{2^{\frac{np}{2}} |V|^{\frac{n}{2}} \Gamma_p(\frac{n}{2})}$$

CDF

$$-$$

Wishart2

name	Wishart 2 (ID: 0000222)
type	continuous
variate	X, matrix
support	$X(p \times p)$ – symmetric, positive definite matrix

Parameter: inverseScaleMatrix

name	inverse scale matrix
type	matrix
symbol	R
definition	$p \times p$ – symmetric, positive definite matrix

Parameter: degreesOfFreedom

name	degrees of freedom
type	scalar
symbol	k
definition	–

Functions

PDF

$$|R|^{k/2}|x|^{(k-p-1)/2}e^{-\frac{1}{2}\,tr(Rx)}$$

CDF

$$-$$

ZeroInflatedNegativeBinomial

name	Zero-Inflated Negative Binomial (ID: 0000232)
type	discrete
variate	y, scalar
support	$k \in \{0, 1, 2, 3, \ldots\}$

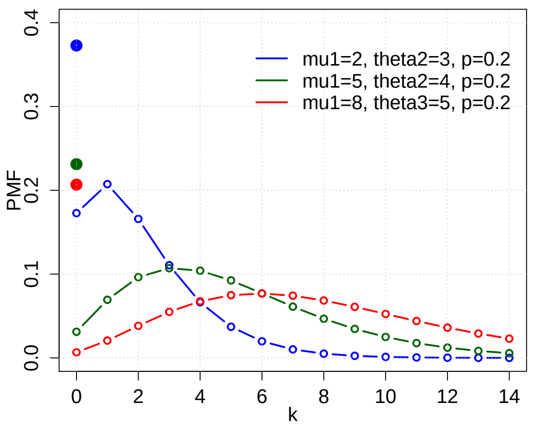


Figure A.32: PDF and CDF of the ZINB distribution plotted using the provided R-code.

Parameter: mean

name	mean
type	scalar
symbol	μ
definition	–

Parameter: sizeParameter

name	size parameter
type	scalar
symbol	θ
definition	$-$

Parameter: probabilityOfZero

name	probability of zero
type	scalar
symbol	p
definition	$0 < p < 1, p \in R$

Functions

PMF

$$\begin{cases} p + (1-p)\left(\frac{\theta}{\theta+\mu}\right)^{\theta}\left(\frac{\mu}{\theta+\mu}\right)^{y} & \text{for } y = 0 \\ (1-p)\frac{\Gamma(\theta+y)}{\Gamma(a)\Gamma(y+1)}\left(\frac{\theta}{\theta+\mu}\right)^{\theta}\left(\frac{\mu}{\theta+\mu}\right)^{y} & \text{for } y > 0 \end{cases}$$

PMF in R

```
p+(1-p)*(theta/(theta+mu))^theta if k=0
(1-p)*gamma(theta+k)/gamma(theta)/gamma(k+1)*(theta/(theta+mu))^theta*(mu/(theta+mu))^k   if k>0
```

CDF

$-$

ZeroInflatedPoisson

name	Zero-inflated Poisson (ID: 0000243)
type	discrete
variate	k, scalar
support	$k \in \{0, 1, 2, 3, \dots\}$

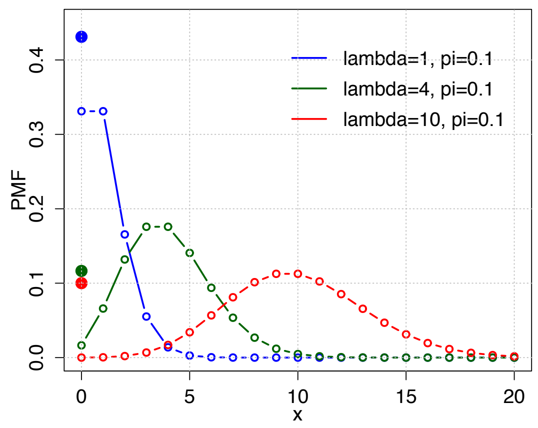


Figure A.33: PMF of the ZIP distribution plotted using the provided R-code.

Parameter: rate

name	Poisson intensity
type	scalar
symbol	λ
definition	$\lambda \in R, \lambda > 0$

Parameter: probabilityOfZero

name	probability of extra zeros
type	scalar
symbol	π
definition	$0 < \pi < 1, \pi \in R$

Functions

PMF

$$\begin{cases} \pi + (1-\pi)e^{-\lambda} & \text{for } k = 0 \\ (1-\pi)e^{-\lambda}\frac{\lambda^k}{k!} & \text{for } k > 0 \end{cases}$$

PMF in R

```
pi + (1-pi)*exp(-lambda)  if k=0
(1-pi)*exp(-lambda) * lambda^k/factorial(k)   if k>0
```

CDF

–

Bibliography

[1] GEP Box and DR Cox. An analysis of transformations. *Journal of the Royal Statistical Society. Series B (Methodological)*, 26(2):211–252, 1964.

[2] CDISC consortium. CDISC Study Design Model in XML (SDM-XML), Version 1.0. Technical report, Clinical Data Interchange Standards Consortium, Inc, 2011.

[3] Lorenzo Chiudinelli, Nicola Melillo, Paolo Magni, Enrica Mezzalana, and Lorenzo Pasotti. Bayesian model requirements for MDL and PharmML. Different situations to be represented and corresponding WinBugs implementation. Technical report, UNIPV, 2015.

[4] Emmanuelle Comets, Marylore Chenel, and Andrew Hooker. Modelling Description Language, Design elements - Examples, Draft 1 version 2. Technical report, INSERM, UPD; Servier; Uppsala University, 2015.

[5] Emmanuelle Comets, Marylore Chenel, and Andrew Hooker. Modelling Description Language, Design elements, Draft 1 version 2. Technical report, INSERM, UPD; Servier; Uppsala University, 2015.

[6] A. Schumitzky D'Argenio, D.Z. and X. Wang. Adapt 5 user's guide: Pharmacokinetic/pharmacodynamic systems analysis software. Technical report, Biomedical Simulations Resource, Los Angeles, 2009.

[7] Catherine Forbes, Merran Evans, Nicholas Hastings, and Brian Peacock. *Statistical Distributions*. Wiley, 4 edition, 2010.

[8] Charles Kooperberg. Logspline: Logspline density estimation routines, 2013. R package version 2.1.5.

[9] Marc Lavielle. *Mixed Effects Models for the Population Approach: Models, Tasks, Methods and Tools*. Chapman & Hall/CRC Biostatistics Series, 2014.

[10] Marc Lavielle. Four models. Technical report, INRIA Saclay, April 3, 2014.

[11] David S LeBauer, Michael C Dietze, and Benjamin M Bolker. Translating probability density functions: From r to bugs and back again. *R Journal*, 5(1), 2013.

[12] Leemis, M Lawrence, Mcqueston, and T Jacquelyn. Univariate distribution relationships. *The American Statistician*, 62(1):45–53, 2008.

[13] MRC Biostatistics Unit. winBUGS Examples - Volume 1. Available at `http://www.mrc-bsu.cam.ac.uk/wp-content/uploads/WinBUGS_Vol1.pdf`.

[14] Natalya F Noy, Nigam H Shah, Patricia L Whetzel, Benjamin Dai, Michael Dorf, Nicholas Griffith, Clement Jonquet, Daniel L Rubin, Margaret-Anne Storey, Christopher G Chute, et al. Bioportal: ontologies and integrated data resources at the click of a mouse. *Nucleic acids research*, page gkp440, 2009.

[15] Elodie L Plan, Alan Maloney, Iñaki F Trocóniz, and Mats O Karlsson. Performance in population models for count data, part i: maximum likelihood approximations. *J Pharmacokinet Pharmacodyn*, 36(4):353–66, Aug 2009.

[16] A S Rudenski, D R Matthews, J C Levy, and R C Turner. Understanding "insulin resistance": both glucose resistance and insulin resistance are required to model human diabetes. *Metabolism*, 40(9):908–17, Sep 1991.

[17] Wheyming Tina Song and Yi-Chun Chen. Eighty univariate distributions and their relationships displayed in a matrix format. *Automatic Control, IEEE Transactions on*, 56(8):1979–1984, 2011.

[18] Maciej J Swat. Pavia PharmML workshop, November 2013. Technical report, EMBL-EBI, 2014.

[19] Maciej J Swat, Sarala Wimalaratne, and Niels Rode Kristensen. Changes in PharmML 0.3 and 0.3.1. Technical report, EMBL-EBI, July 2014.

[20] Maciej J Swat, Sarala Wimalaratne, and Niels Rode Kristensen. Changes in PharmML 0.4 and 0.4.1. Technical report, EMBL-EBI, September 2014.

[21] Maciej J Swat, Sarala M. Wimalaratne, Niels R Kristensen, Florent Yvon, Stuart Moodie, and N. Le Novère. Pharmacometrics Markup Language (PharmML), Language Specification for Version 0.6. January 2015.

[22] Iñaki F Trocóniz, Elodie L Plan, Raymond Miller, and Mats O Karlsson. Modelling overdispersion and markovian features in count data. *J Pharmacokinet Pharmacodyn*, 36(5):461–77, Oct 2009.

[23] UncertML Team. Uncertainty Markup Language: UncertML Version 3.0. Avaiable at `http://www.uncertml.org`, 2014.

[24] P Wang, M L Puterman, I Cockburn, and N Le. Mixed poisson regression models with covariate dependent rates. *Biometrics*, 52(2):381–400, Jun 1996.

[25] Eric Weisstein. Wolfram mathworld, 2007.