Food Safety Knowledge Markup Language (FSK-ML)

Software Developer Guide

Version 0.9

Matthias Filter (Chair)	Federal Institute for Risk Assessment, Germany
Sascha Bulik	Federal Institute for Risk Assessment, Germany
Carolina Plaza-Rodriguez	Federal Institute for Risk Assessment, Germany
Miguel de Alba Aparicio	Federal Institute for Risk Assessment, Germany

Alumni contributors:

Guido Correia Carreia	Federal Institute for Risk Assessment, Germany
Alexander Falenski	Federal Institute for Risk Assessment, Germany

Contact:

Matthias Filter (matthias.filter@bfr.bund.de)

List of contents

List of tables

List of figures

1. Introduction

Food safety risk assessments, control of food production processes as well as the development of new food products are nowadays supported by application of mathematical modeling and data analysis techniques. This creates an increasing demand for resources facilitating the efficient, transparent and quality proven exchange of relevant information, e.g. analytical data, mathematical models, simulation setting as well as simulated data. For example, new parameterized microbial models are frequently made publicly available only in written mode via scientific publications. However, in order to apply these models to a given practical decision support question (e.g. on the growth/no-growth of a microorganism in a specific food matrix under given processing conditions) the interested end-user would have to re-implement the model based on information provided in a publication. Here it would be more efficient if those who create parameterized models could provide their model additionally as a file complying with a standardized file format that is also capable of transferring all relevant meta-information. Such a file could e.g. be provided as a supplement to the publication and could be read-in by the end user's software tools (thus overcoming an error-prone re-implementation process).

The required standardized file format has been proposed in the "Predictive Modelling in Food Markup Language (PMF-ML) Software Developer Guide". This document describes in detail how experimental data and mathematical models from the domain of predictive microbial modeling (and beyond) can be saved and encoded in a **software independent manner**. Here we further extend the PMF-ML format in order to enable in addition the exchange of knowledge / information that is **embedded in specific script-based programming languages** (e.g. "R", Matlab, Python). I.e. the FSK-ML guidance document primarily aims at harmonizing the exchange of food safety knowledge (e.g. predictive models) including the associated meta-information where this knowledge is only available in a **software-dependent** format.

The FSK-ML format therefore relaxes and adapts certain specifications of the PMF-ML format while at the same time maintaining the highest possible synergies between both formats. This will also help to make sure that food safety models encoded in a software-independent manner (using PMF-ML) can easily be interpreted by FSK-ML import and export software functions in the future.

1.1. Objectives

This guidance document is primarily designed for software developers and project managers that want to enhance their software tools with import and export functions for food safety models, simulations or food safety data or for those who want to develop new tools.

The document describes the structure and requirements of FSK-ML including the structure and requirements of a newly defined file format which is based on the Open Modeling EXchange guidelines (OMEX). The document also describes information needed to make encoded script-based models executable. Future versions of the document will also describe how to encode simulation settings and settings for data visualization.

1.2. Document conventions

This document uses the conventions defined in the SED-ML specification document (Bergmann, Cooper, Le Novère, Nickerson, & Waltermath, 2015).

UML 1.0 (Unified Modelling Language), (OMG, 2009), notation is used in this document to define the constructs provided by this package. The following colours are used to provide more meaningful diagrams.

- **Black**. Items coloured black in the UML diagrams are components taken unchanged from their definition in the SED-ML Level 1 Version 2 specification document.
- **Green**. Items coloured green are components that exist in SED-ML Level 1 Version 2, but are extended by this package.
- **Blue**. Items coloured blue are new components introduced in this package specification. They have no equivalent in the SED-ML Level 1 Version 2 specification.

The following typographical conventions distinguish the names of objects and data types from other entities.

AbstractClass: Abstract classes are never instantiated directly, but rather serve as parents of other classes. Their names begin with a capital letter and they are printed in a slanted, bold, sans-serif typeface. In electronic document formats, the class names defined within this document are also hyperlinked to their definitions; clicking on these items will, given appropriate software, switch the view to the section in this document containing the definition of that class. (However, for classes that are unchanged from their definitions in `SED-ML` Level 1 Version 2, the class names are not hyperlinked because they are not defined within this document.)

Class: Names of ordinary (concrete) classes begin with a capital letter and are printed in an upright, bold, sans-serif typeface. In electronic document formats, the class names are also hyperlinked to their definitions in this specification document. (However, as in the previous case, class names are not hyperlinked if they are for classes that are unchanged from their definitions in the `SED-ML` Level 1 Version 2 specification.)

`SomeThing`, `otherThing`: Attributes of classes, data type names, literal XML, and tokens other than `SED-ML` class names, are printed in an upright typewriter typeface.

2. Related standards

2.1. SBML

See “Predictive Modelling in Food Markup Language (PMF-ML) Software Developer Guide”.

2.2. SED-ML

See “Predictive Modelling in Food Markup Language (PMF-ML) Software Developer Guide”.

2.3. PMF-ML

See “Predictive Modelling in Food Markup Language (PMF-ML) Software Developer Guide”.

2.4. OMEX

Open Modelling EXchange format, OMEX (Bergmann, et al., 2014), aims to support the exchange of information necessary for modelling and simulating experiments in biology. OMEX defines a COMBINE Archive or OMEX files as a ZIP container containing at least a file with a listing of content in the archive (manifest file), optional metadata and the files describing e.g. food safety models. The use of COMBINE standard file formats for the encoding of models is encouraged, although several Internet media types are also supported.

3. FSK-ML Specifications

3.1. FSK Terminology

3.1.1. Model

A model is a mathematical description of a system consisting of state variables and their dependency on each other and external conditions (e.g. pH, temperature, time). The mathematical representation of these dependencies contain fixed parameter values specific for this model and variable values, also called “independent parameters” or “input parameters” here, that can be set to different values to attain model prediction for not measured combinations of these entities. It is to note that a model is normally only valid on a given range for these variable values.

3.1.2. Simulation

Model-based predictions generated from different parameterizations of the same model are called simulations. I.e. generating model-based predictions from a set of definite values for all model input parameters will be called a simulation. It should be noted, that a simulation might create different sets of definite values for input parameter as FSK-ML support script-based simulation settings.

3.2. The FSKX format

In order to support the exchange of sets of related files a FSKX container file is created. A FSKX container file is a ZIP file containing an arbitrary (at least one) number of files. We recommend the file extension “.fskx”. The FSKX container has to comply with the Open Modeling EXchange format (OMEX) specifications set out in (Bergmann, Adams et al. 2014). It allows to exchange the following files:

- a manifest file (“manifest.xml”) - mandatory; providing a listing of all files inside the FSKX container

- a metadata file (“metadata.rdf”) providing additional information on the archive and its content
- files related to the model(s):
  - Data used for model generation / validation (e.g. “experiment_1.numl”)
  - model metadata (e.g. “metaData_1.pmf”) providing all relevant metadata on referenced model script(s) / executable software (black-box) model(s)
  - model script(s) (e.g. “model_1.r”) / executable software (black-box) model(s) (e.g. “model_1.exe“/ pmfx files (e.g. “model_1.pmfx”)
  - scripts with default values for independent model parameters (e.g. “model_param_1.r”)
  - Third party (including binary) libraries required to run the model script(s) / executable software (black-box) model(s) (e.g. “triangle_0.10.zip”)

- file(s) related to simulation(s) / prediction and simulation / prediction result(s):
  - Simulation settings (e.g. “Simulation_1.sed”)
  - Simulation results (e.g. “Simulation_1_res.numl”)
  - Visualization script (e.g. “Simulation_1_plot.r”)
- Other file(s)
  - Model reference description (e.g. “Paper_model_1.pdf”)

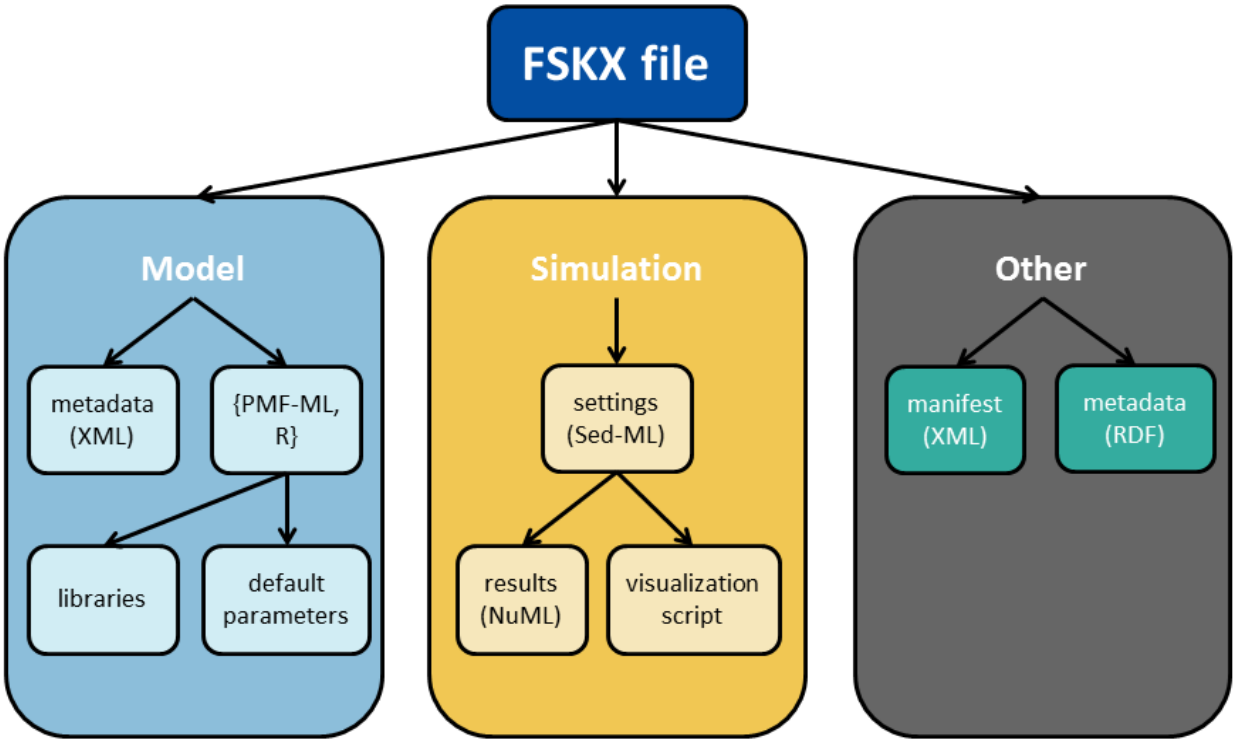

Figure 1 Structure of the FSKX file

3.3. Supported data types

Metadata values and result or input data of the specified models must adhere to the supported data types defined here:

- Numeric: Real numbers.
- Integer
- Character: String values
- Vector: one-dimensional array
- Matrix: two-dimensional array

Code vocabulary for the supported data types:

Numeric	`numeric`
Integer	`integer`
Character	`character`
Vector	`vector`
Matrix	`matrix`

Table 1 Supported data types

The supported data types are case-insensitive.

```
# Examples of scalar data types
pi = 3.14
year = 2016
name = "Miguel, Lord of Westeros"

# Vectors
cookies_per_meeting <- c(10, 5, 20, 15, 10)
cookies_per_meeting <- c(10, "fünf", 20, "fünfzehn", 10)
cookies_per_meeting <- c("zehn", "fünf", "zwanzig", "fünfzehn",
       "zehn")

# Matrices
menus = matrix(
       c("Hänchen", "Reis", "Fishfilet", "Geschnezeltes", "Spinach",
              "Boulette"),
   nrow = 2, ncol = 3, byrow = TRUE,
   dimnames = list(
       c("Menu 1", "Menu 2"), c("Montag", "Dienstag", "Mittwoch")))
```

Examples variables

3.4. Files related to models

3.4.1. Model script

The model script is a script stored within the FSKX archive that calculates the results of the model. Each model script has to be declared in the manifest.xml, annotated by a dedicated metadata file (.pmf) and can be referenced by any simulation settings file (.sed)

```
######################################################################
test_model_1 = function(a=1, b=2, c=3){
  res = 100*(rbeta(a, shape1 = b + 1, shape2 = c - b + 1))
  return(res)
}
test_model_1()
```

Example model script

3.4.2. (Binary) Model libraries

For an autonomous run the model needs to be provided such that required libraries are provided together with the model. In the case of an R model the corresponding libraries would be binary libraries for the system the model was created on.

While the placement of the libraries in the archive is free, they can be stored anywhere in the FSKX file; it is advisable to save them into a separated "lib" folder within the archive.

3.4.3. Model metadata

The model metadata are stored in a JSON file encoding model metadata in a "generic module" (Figure 2). The "generic module" contains an exhaustive list of metadata to describe in detail each model (Table 2). The "generic module" in the JSON file describes the model with a list of name/value pairs matching the metadata described in Table 2. A specific dimension has been attributed to each metadata, in order to specify if the metadata is mandatory or not and how many inputs can be included for a specific metadata. Each of the FSK model classes (see section 3.4.4. FSK Model Classes) contain specific metadata that has been selected from the list of metadata from the "generic module", in order to facilitate and orientate the end-user when providing information on a specific type of model. In this case the dimension of each metadata has been adapted accordingly.

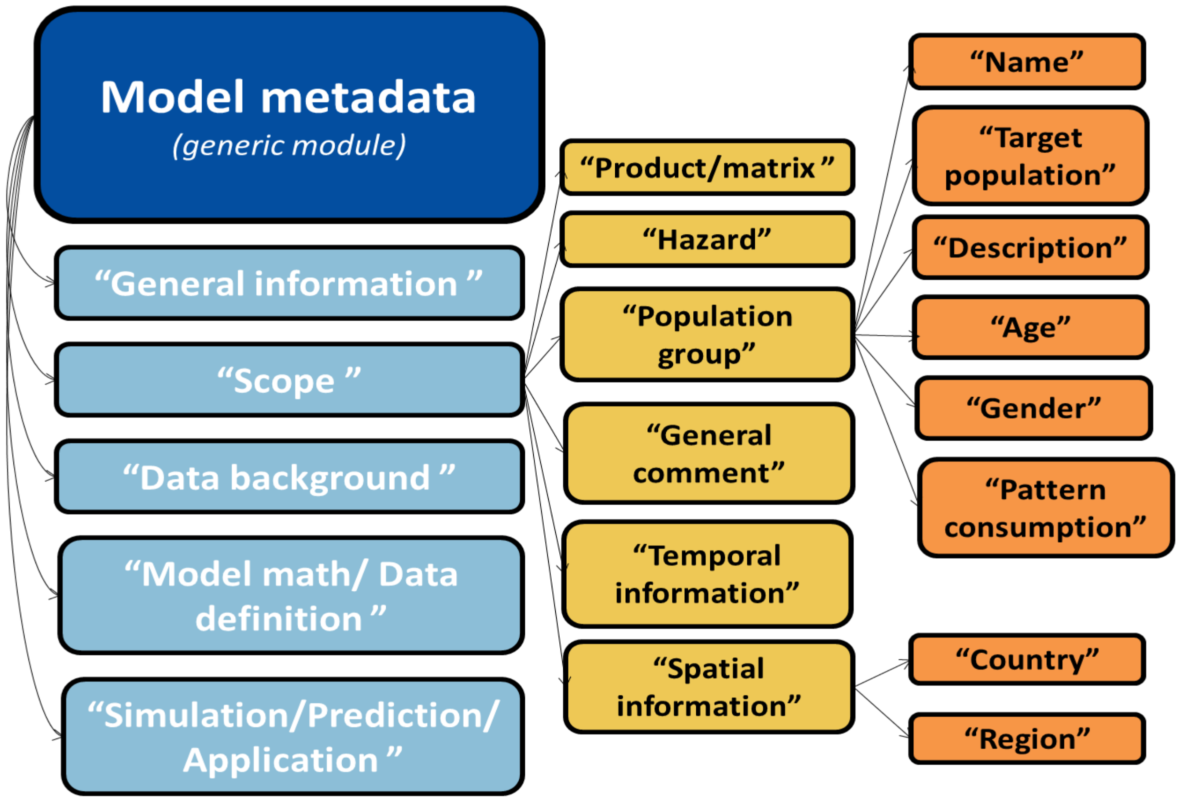

Figure 2 Model metadata encoded in a standardized way within the FSK-ML file

General Information	1	**Study / Data / Model Name**	1		
		Source	0:1		
		Identifier	1		
		Creator(s)	1:N	*vCard 4.0 standard*	*1*
		Date	1:N	*Creation date*	*1*
				Last modified date	*0:N*
		Rights	1	*Rights*	*1*
		Availability	1		
		URL	0:1		
		Format	0:1		
		References	1:N	*Is_reference_description?*	*1*
				Publication type	*0:1*
				Publication date	*0:1*
				PubMed ID	*0:1*
				Publication DOI	*1*
				Publication Author List	*0:1*
				Publication Title	*1*
				Publication Abstract	*0:1*
				Publication Journal / Vol / Issue, etc.	*0:1*
				Publication Status	*0:1*
				Publication website	*0:1*
				Comment	*0:1*
		Language	0:1		
		Software	0:1		
		Programing language	0:1		
		Model category	0:1	*Model Class*	*1*
				Model Sub-Class	*0:N*
				Model Class comment	*0:1*
				Basic process	*0:N*
		Status	0:1		
		Objective	0:1		
		Description	0:1		
Scope	1	**Product / matrix**	0:1	*Product/matrix name*	*1*

				Product/matrix description	*0:1*
				Product/matrix unit	*1*
				Method of production	*0:N*
				Packaging	*0:N*
				Product treatment	*0:N*
				Country of origin	*0:1*
				Area of origin	*0:1*
				Fisheries area	*0:1*
				Date of production	*0:1*
				date of expiry	*0:1*
		Hazard	**0:1**	*Hazard type*	*1*
				Hazard name	*1*
				Hazard description	*0:1*
				Hazard unit	*1*
				Adverse effect	*0:1*
				Origin	*0:1*
				Benchmark Dose (BMD)	*0:1*
				Maximum Residue Limit (MRL)	*0:1*
				No Observed Adverse Effect Level (NOAEL)	*0:1*
				Lowest Observed Adverse Effect Level (LOAEL)	*0:1*
				Acceptable Operator Exposure Level (AOEL)	*0:1*
				Acute Reference Dose (ARfD)	*0:1*
				Acceptable Daily Intake (ADI)	*0:1*
				Hazard ind/sum	*0:1*
		Population Group	**0:1**	*Population name*	*1*
				Target population	*0:1*
				Population Span (years)	*0:N*
				Population description	*0:N*
				Population age	*0:N*
				Population gender	*0:1*
				BMI	*0:N*
				Special diet groups	*0:N*
				Pattern consumption	*0:N*
				Region	*0:N*
				Country	*0:N*
				Risk and population factors	*0:N*
				Season	*0:N*
		General comment	**0:1**		
		Temporal information	**0:1**	*Time*	*1:N*
		Spatial information	**0:1**	*Region*	*0:N*
				Country	*0:N*
Data background	**0:1**	**Study**	**1**	*Study Identifier*	*1*
				Study Title	*1*
				Study Description	*0:1*
				Study Design Type	*0:1*
				Study Assay Measurement Type	*0:1*
				Study Assay Technology Type	*0:1*
				Study Assay Technology Platform	*0:1*
				Accreditation procedure for the assay technology	*0:1*
				Study Protocol Name	*0:1*
				Study Protocol Type	*0:1*
				Study Protocol Description	*0:1*
				Study Protocol URI	*0:1*
				Study Protocol Version	*0:1*
				Study Protocol Parameters Name	*0:1*
				Study Protocol Components Name	*0:1*
				Study Protocol Components Type	*0:1*
		Study Sample	**0:1**	*Sample Name (ID)*	*1*
				Protocol of sample collection	*1*
				Sampling strategy	*0:1*
				Type of sampling program	*0:1*
				Sampling method	*0:1*
				Sampling plan	*1*
				Sampling weight	*1*
				Sampling size	*1*
				Lot size unit	*0:1*
				Sampling point	*0:1*
		Dietary assessment method	**0:1**	*Methodological tool to collect data*	*1*
				Number of non-consecutive one-day	*1*
				Dietary software tool	*0:1*
				Number of food items	*1:N*
				Type of records	*1:N*
				Food descriptors	*1:N*
		Laboratory	**0:1**	*Laboratory accreditation*	*1:N*

				Laboratory Name	*0:1*
				Laboratory country	*0:1*
		Assay	**0:1**	*Assay Name*	*1*
				Assay description	*0:1*
				Percentage of moisture	*0:1*
				Percentage of fat	*0:1*
				Limit of detection	*0:1*
				Limit of quantification	*0:1*
				Left-censored data	*0:1*
				Range of contamination	*0:1*
				Uncertainty value	*0:1*
Model math / Data definition	**1**	**Parameter / Factor / Input / Output / "Data column"**	**1:N**	*Parameter ID*	*1*
				Parameter classification	*1*
				Parameter name	*1*
				Parameter description	*0:1*
				Parameter type	*0:1*
				Parameter unit	*1*
				Parameter unit category	*1*
				Parameter data type	*1*
				Parameter source	*0:1*
				Parameter subject	*0:1*
				Parameter distribution	*0:1*
				Parameter value	*0:1*
				Parameter Reference	*0:1*
				Parameter variability subject	*0:1*
				Range of applicability of the model	*0:N*
				Parameter error	*0:1*
		Quality measures	**0:N**	*SSE / MSE / RMSE / Rsquared / AIC / BIC*	*0:N*
				Sensitivity analysis	*0:N*
		Model equation	**0:N**	*Model equation name*	*1*
				Model equation class/distribution	*0:1*
				Model equation reference	*0:N*
				Model equation / Script	*1*
				Hypothesis of the model	
		Fitting procedure	**0:1**		
		Exposure	**0:1**	*methodological treatment of left-censored data*	
				Level of contamination after left-censored data treatment	
				type of exposure	
				Scenario	
				Uncertainty estimation	
		Events	**0:N**		

Table 2 Model metadata

In addition, FSK-ML provides a list of controlled vocabularies for some specific metadata (Table 3) and https://docs.google.com/spreadsheets/d/1C6N4-YWX9OMmNStd2rYlSUaVys-aiJGLj00cD44aVc8/edit#gid=1479548673 , based on the terms used by other sources like ontologies, standards and tools (SSD-CODE, FOODON, MIME, PMM-Lab, OpenFSMR, Bibliographic Ontology Specification, etc.)

Source	*controlled vocabulary_Terms and concepts*
Rights	*controlled vocabulary*
Format	*controlled vocabulary_Internet Media Types [MIME]*
Publication Type	*controlled vocabulary_Bibliographic Ontology Specification*
Publication Status	*controlled vocabulary_Bibliographic Ontology Specification*
Software	*controlled vocabulary*
Language	*controlled vocabulary_SSD-code*
Language written in	*controlled vocabulary*
Model Class	*controlled vocabulary*
Model Sub-Class	*class-specific controlled vocabularies incl. OTHER (OpenFSMR)*
Basic process	*sub-sub-class-specific controlled vocabularies incl. OTHER*
Status	*controlled vocabulary*

Product-matrix name	*controlled vocabulary_**SSD-code***
Product-matrix unit	*controlled vocabulary_**SSD-code_PMM-Lab***
Method of production	*controlled vocabulary_**SSD-code***
Packaging	*controlled vocabulary_**SSD-code***
Product treatment	*controlled vocabulary_**SSD-code***
Country of origin	*controlled vocabulary_**SSD-code***
Area of origin	*controlled vocabulary_**SSD-code***
Fisheries area	*controlled vocabulary_**SSD-code***
Hazard type	*controlled vocabulary_**SSD-code***
Hazard name	*controlled vocabulary_**SSD-code***
Hazard unit	*controlled vocabulary_**SSD-code_PMM-Lab***
Hazard ind-sum	*controlled vocabulary_**SSD-code***
Population name	*controlled vocabulary_**FOODON***
Laboratory country	*controlled vocabulary_**SSD-code***
Region	*controlled vocabulary_**SSD-code***
Country	*controlled vocabulary_**SSD-code***
Study Assay Technology Type	*controlled vocabulary_**SSD-code***
Accreditation procedure for the assay technology	*controlled vocabulary_**SSD-code***
Sampling strategy	*controlled vocabulary_**SSD-code***
Type of sampling program	*controlled vocabulary_**SSD-code***
Sampling method	*controlled vocabulary_**SSD-code***
Lot size unit	*controlled vocabulary_**SSD-code_PMM-Lab***
Sampling point	*controlled vocabulary_**SSD-code***
Methodological tool to collect data	*controlled vocabulary*
Type of records	*controlled vocabulary*
Food descriptors	*controlled vocabulary_**SSD-code***
Laboratory accreditation	*controlled vocabulary_**SSD-code***
Parameter classification	*controlled vocabulary*
Parameter type	*controlled vocabulary*
Parameter unit	*controlled vocabulary_**SSD-code_PMM-Lab***
Parameter unit category	*controlled vocabulary_**PMM-Lab***
Parameter data type	*controlled vocabulary*
Parameter source	*controlled vocabulary*
Parameter subject	*controlled vocabulary*
Parameter distribution	*controlled vocabulary_**probONTO***
Model equation class - distribution	*controlled vocabulary_**iRISK***
Fitting procedure	*controlled vocabulary*
type of exposure	*controlled vocabulary*
Simulation algorithm	*controlled vocabulary*

Table 3 List of metadata that are associated with controlled vocabularies lists and its sources

3.4.4. FSK Model Classes

FSK model classes characterize the particular models that can currently be described by FSK-ML. This classification is partly based on the definition made by CODEX *Alimentarius* on the Risk Assessment (Table 4). Some of the FSK model classes are subdivided into Sub-classes, which are included as controlled vocabularies lists. Apart from this FSK Model Classes, FSK-ML allows to annotate models under the label “Generic module” which allows more flexibility in providing information on the model.

FSK Model Class	FSK Model Sub-Class
Data	
Predictive model	Growth; Growth boundary model; Inactivation; Maximum population density (MPD); Metabolite formation; Spoilage; Survival; Transfer; Primary; Secondary; Primary-secondary; Other.
Other empirical models	Pharmacokinetics models; Time-temperature modes; Temporary models; Epidemiological models; Fluid dynamic models; Other.
Process models	Animal farm; Crop farm; Fishery; Processing centre-slaughter; Storage; Transport-distribution; Retail; Consumer; Food service; Other.
Consumption models	
Exposure model	
Dose-response model	
Toxicological reference value model	
Health metrics model	
QRA model	

Table 4 FSK model classes

3.5. Simulation files

FSK-ML provides an opportunity to define, store and exchange simulation scenarios. All relevant settings (including the reference to the used model files) are stored in XML files using the FSK-SED-ML format (see section 4). Also visualizations generated on the basis of the simulation results can be referenced / defined in the FSK-SED-ML file.

The results of simulations can be stored either as software independent NuML XML file or as a scripting language specific binary file where the results from executing a defined simulation are stored. In case of the R scripting language a valid simulation result file would therefore be a “R workspace” generated after the simulation of one R model.

3.5.1. FSK-SED-ML files

The FSK-SED-ML files are XML files (.sed) that describe specific simulations to be carried out with model(s). For this it has to describe:

- The input parameter values for the simulation or the script defining the generation of input parameters for the simulation. In addition it has to be defined how the simulation is carried out itself (including the definition how the model script is executed with each simulation input parameter set)

- Metadata on the type of simulation: deterministic, statistic or probabilistic
- The output format (data types) in which the results of the simulation are stored if the results are provided inside a language-specific workspace. If simulation results are given as NuML files this data type definition is given inside the NuML file.

FSK-SED-ML is described in more detail in section 4 FSK-SED-ML.

3.5.2. Visualization scripts

Visualization scripts can be included and might contain a number of commands in a scripting language corresponding to the one used for the model. It can be used to create plots or charts using from the results generated in a simulation. Visualization scripts might also be referenced from within FSK-SED-ML files.

```
hist(test_model_1(200, 20, 100), breaks=50, main="Headline", xlab="Text", col="32")
```

Example visualization script

3.5.3. Results (NuML)

The results of the model are preferably encoded with NuML. Each of the supported data types are encoded in a different fashion.

General guidelines

The variables of the results are kept within a NuML **CompositeDescription** in the **Dimension Description** of a **ResultComponent**. This **CompositeDescription** keeps the names and types of the variables. The values of the variables are saved into a single **Dimension** corresponding to the **Composite Description**.

```
<numl>
...
  <resultComponent id=”rc1”>
    <dimensionDescription>
      <compositeDescription>
      Variables ...
    </dimensionDescription>
    <dimension>
      Values ...
    </dimension>
  </resultComponent>
</numl>
```

3.5.3.1. Integer variable

An integer variable is described with a single **AtomicDescription** of type `integer`. Its value is described with an **AtomicValue**.

```
<atomicDescription name=”integer_variable” valueType=”integer” />
```

NuML Integer variable description

```
<atomicValue>10</atomicValue>
```

NuML Integer variable value

3.5.3.2. Numeric variable

An integer variable is described with a single **AtomicDescription** of type `numeric`Its value is described with an **AtomicValue**.

```
<atomicDescription name=”numeric_variable” valueType=”numeric” />
```

NuML numeric variable description

```
<atomicValue>3.141592653589793</atomicValue>
```

NuML numeric variable value

3.5.3.3. Character variable

An integer variable is described with a single **AtomicDescription** of type `character`. Its value is described with an **AtomicValue**.

```
<atomicDescription name=”character_variable” valueType=”character” />
```

NuML character variable description

```
<atomicValue>3.141592653589793</atomicValue>
```

NuML character variable value

3.5.3.4. Vector variable

A vector variable is described with a **CompositeDescription** with the name of the variable and with `indexType` integer. This **CompositeDescription** involves an **AtomicDescription** with a `valueType`. The types may be `integer` for integer vectors, `numeric` for numeric vectors and `character` for character vectors.

```
<compositeDescription name=”vector_variable” indexType=”integer”>
  <atomicDescription name=”vector_item” ontologyTerm=”dummy”
    valueType=”integer”|”numeric”|”character” />
</compositeDescription>
```

NuML vector variable description

Its values are described by a **CompositeValue** coupled with a single **AtomicValue** of the same type as in its description. The values are contained in a **Dimension** corresponding to the **CompositeDescription** of the vector.

```
<dimension>
  <compositeValue indexValue=”0”><atomicValue>0</atomicValue></compositeValue>
  <compositeValue indexValue=”1”><atomicValue>1</atomicValue></compositeValue>
  <compositeValue indexValue=”2”><atomicValue>1</atomicValue></compositeValue>
  <compositeValue indexValue=”3”><atomicValue>2</atomicValue></compositeValue>
  <compositeValue indexValue=”4”><atomicValue>3</atomicValue></compositeValue>
  ...
</dimension>
```
NuML vector variable values

3.5.3.5. Matrix variable

A matrix variable is fairly similar to a vector variable but with further columns. It is described with a **CompositeDescription** providing the name of the variable and `indexType` integer. The **CompositeDescription** involves **AtomicDescriptions** with the same `valueType`. The types may be `integer` for integer vectors, `numeric` for numeric vectors and `character` for character vectors.

```
<compositeDescription name=”matrix_variable” indexType=”integer”>
  <atomicDescription name=”col1” ontologyTerm=”dummy” valueType=”integer”|”numeric”|”character”
/>
  <atomicDescription name=”col2” ontologyTerm=”dummy” valueType=”integer”|”numeric”|”character”
/>
</compositeDescription>
```
NuML matrix variable description

Its values are described by a **CompositeValue** consisting of **AtomicValues** of the same type as in its description. The values are contained in a **Dimension** corresponding to the **CompositeDescription** of the matrix.

```
<dimension>
  <compositeValue indexValue=”0”>
    <atomicValue>0</atomicValue><atomicValue>0</atomicValue>
  </compositeValue><compositeValue indexValue=”1”>
    <atomicValue>1</atomicValue><atomicValue>1</atomicValue>
  </compositeValue><compositeValue indexValue=”2”>
    <atomicValue>2</atomicValue><atomicValue>4</atomicValue>
  </compositeValue><compositeValue indexValue=”3”>
    <atomicValue>3</atomicValue><atomicValue>9</atomicValue>
  </compositeValue><compositeValue indexValue=”4”>
    <atomicValue>4</atomicValue><atomicValue>16</atomicValue>
  </compositeValue>
</dimension>
```
NuML matrix variable values

3.6. Other

3.6.1. manifest.xml

The FSK container is a COMBINE archive and the presence of this file is a requirement of the COMBINE archive.

One file must be present at the root of any COMBINE archive, named manifest.xml. This file contains an instantiation of the OmexManifest class. It contains a number of Content children, one of which represents the COMBINE archive itself. A valid manifest needs to have at least one entry, declaring the archive itself, but may contain as many entries as needed. All the files present in the archive must be listed in the manifest. The only optional entry describes the manifest.xml file. Indeed the presence of manifest.xml is mandatory and its declaration is not necessary for parsing.

```
<?xml version="1.0" encoding="UTF-8"?>
<omexManifest
       xmlns="http://identifiers.org/combine.specifications/omex-manifest">
  <content location="."
       format="http://identifiers.org/combine.specifications/omex" />
  <content location="./manifest.xml"
       format="http://identifiers.org/combine.specifications/omex-manifest" />
  <content location="./visualization.r"
       format="http://purl.org/net/mediatypes/application/text/x-r" />
  <content location="./metaData.pmf"
       format="http://purl.org/net/mediatypes/application/application/sbml+xml" />
  <content location="./workspace.r"
       format="http://purl.org/net/mediatypes/application/text/x-r" />
  <content location="./triangle_0.10.zip"
       format="http://purl.org/net/mediatypes/application/application/zip" />
  <content location="./model.r"
       format="http://purl.org/net/mediatypes/application/text/x-r" />
  <content location="./param.r"
       format="http://purl.org/net/mediatypes/application/text/x-r" />
  <content location=".\metadata.rdf"
       format="http://identifiers.org/combine.specifications/omex-metadata" />
</omexManifest>
```

Example manifest

The `Content` class represents a file in the COMBINE archive. The format attribute in a content element describes the file type. It may take either a URI corresponding to Identifiers.org for COMBINE standards such as SBML or SedML or an Internet media type (Freed, 1996), previously known as "MIME type". FSK makes use of extra file types through the following Internet media types.

File type	Internet media type
Zip (.zip)	http://purl.org/NET/mediatypes/application/zip

R (.r)	http://purl.org/NET/mediatypes/text/x-r
PMF (.pmf)	http://purl.org/NET/mediatypes/application/x-pmf
SBML (.sbml)	http://purl.org/NET/mediatypes/application/sbml+xml
Matlab (.m)	http://purl.org/NET/mediatypes/text/x-matlab
PHP (.php)	http://purl.org/NET/mediatypes/text/x-php

Table 5 Formats used in the manifest

3.6.2. The Archive metadata

A reduced model definition is allowed by providing additional annotation in the RDF format. In the FSKX archive the use of Internet media types as formats allows the detection of different types of files within the archive file such as scripts and binary libraries. However, files might share the same media types, as e.g. for R scripts and R workspace files. Therefore the use of Internet media types is not enough.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
       xmlns:dcterms="http://purl.org/dc/terms/"
       xmlns:vCard="http://www.w3.org/2006/vcard/ns#">

  <rdf:Description rdf:about=".">
       <dc:type>mainScript</dc:type>
       <dc:source>model.r</dc:source>
  </rdf:Description>

  <rdf:Description rdf:about=".">
       <dc:type>paramScript</dc:type>
       <dc:source>param.r</dc:source>
  </rdf:Description>

  <rdf:Description rdf:about=".">
       <dc:type>visualizationScript</dc:type>
       <dc:source>visualization.r</dc:source>
  </rdf:Description>

  <rdf:Description rdf:about=".">
       <dc:type>workspace</dc:type>
       <dc:source>workspace.r</dc:source>
  </rdf:Description>
</rdf:RDF>
```

Example archive metadata

The Archive metadata in FSK addresses this issue, providing a lightweight annotation using the Resource Description Format, RDF (Cyganiak, Wood, Lanthaler, Klyne, Carroll, & and McBride, 2014). This annotation involves RDF description elements that describe the type and location of the resource with DC type and source elements respectively. The accepted types are:

- `mainScript`: filename of the model script
- `paramScript`: filename of the parameters script
- `visualizationScript`: filename of the visualization script

- `workspace`: filename of the workspace with the results of the simulation

4. FSK-SED-ML

FSK-SED-ML extends SED-ML to describe the simulation settings for FSK-ML models. FSK-SED-ML manages this introducing new **Simulation** and **Output** classes compatible with the FSK-ML models. FSK-SED-ML files contain the references to the used models, their parameterizations, the simulation setups, the output methods for the storage of results and the methods used for visualisation of results. Unlike SED-ML FSK-SED-ML does not require the methods to be in an environment independent Markup Language format but contains the capacity to reference to scripts in supported languages. The main structure of the FSK-SED-ML format is maintained (<listOfModels><listOfSimulations><listOfTasks><listOfDataGenerators><listOfOutputs>), but each node of these lists might refer to one or more <sourcescript> attributes.

4.1. <sourcescript> attribute

The attribute <sourcescript> contains the following fields:

Identifier	Format	Content
id	*character*	Name of script
language	See figure 9	Script language
src {use="optional"}	filename	Filename of the script

If no scr is given, the actual script can be provided here as free text.

`language` is a mandatory attribute that identifies the scripting language of code embedded within a sourcescript element or referenced through the `src` attribute. It is specified with MIME types. Examples of MIME types are `text/x-r` and `text/x-python` for the `R` and Python languages respectively.

Language	MIME type
R	`text/x-r`
Python	`text/x-python`
Matlab	`text/x-matlab`

Table 6 MIME types of interest

`src` is an optional attribute that indicates the URI of an external or the relative path and name to a locally given script file. It represents an alternative to embedding a simulation script directly in the FSK-SED-ML file. When the `src` attribute is specified in a simulation script, the Simulation element should not include script code within the <sourcescript> attribute.

4.2. Model

References the model file(s) used in simulation(s). However the execution of models in default or specific simulation settings is defined in the Task node(s). Therefore each FSKX file contains at least one FSK-SED-ML file providing the settings for executing each model with the default parameters.

```
<listOfModels>
    <model id="model1" name="initialize_parents_flocks">
      <sourcescript id="model" language="text/x-r" src="./model.r" />
    </model>
</listOfModels>
```

4.3. Simulation

The **Simulation** node define the settings under which models should be executed. In the simplest case this defines just new input parameters for execution of a given model, e.g. replacing default values given in the paramScript file. Each simulation scenario defined in a Task node has to refer to simulation setting defined in the Simulation node. Nevertheless it is possible that simulation settings defined as a script can already contain the script command that calls a specific model. In general the **Task** node is the place that specifies which simulation setting is combined how with which model.

In SED-ML the simulation classes act as container for defining the simulation experiments. FSK-SED-ML does not use the original simulation classes in SED-ML (**UniformTimeCourse**, **OneStep** and **SteadyState**). FSK-SED-ML introduces a new simulation class, **Simulation**, which can be used to provide metadata on the kind of simulation that will be performed. The following terms are proposed: `deterministic`, `statistic` and `probabilistic`.

- Deterministic simulations operate on definite values for all model input parameters
- Statistic simulations create descriptive analysis of observational data. These simulations are meant to describe and analyse data.
- Probabilistic simulations use probabilistic methods like Monte Carlo simulations to create model input parameters

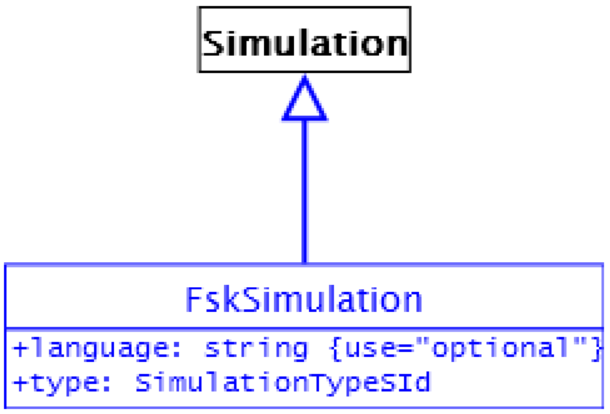

Figure 3 The definition of the FskSimulation class

The **Simulation** class derives two attributes from **Simulation**, id and name, which are mandatory and optional respectively. It also includes a new attribute: `type`.

`type`

`type` is a mandatory attribute that indicates the type of the simulation. The value of the attribute must be taken from the list of reserved words in … These reserved symbols are defined in the value space of the data type `SimulationTypeSId`.

deterministic
statistic
probabilistic

Table 7 Types for FskSimulation

```
<listOfSimulations>
    <fskSimulation id="Simulation_Default" name="normal_settings" type=”deterministic”>
        <sourcescript id="paramScript" language="text/x-r" src="param.r" />
    </fskSimulation>
</listOfSimulations>
```

Example of the FskSimulation class

```
<listOfSimulations>
    <fskSimulation id="Simulation1" name="script_settings" language="R" type=”deterministic”>
        <sourcescript id="simulation1" language="text/x-r">
           source("param.r")
           Npos <- 20
           source("model.r")
           result1 <- result

           Npos <- 30
           source("model.r")
           result2 <- result

           Npos <- 40
           source("model.r")
           result3 <- result
       </sourcescript>
    </fskSimulation>
</listOfSimulations>
```

Example of the FskSimulation class

4.4. Task

The **Task** class defines each simulation scenario through the combination of Models with Simulation setting considered in the current FSK-SED-ML file. It further defines the “execution order” for each simulation scenario. If no additional **Sourcescript** is defined the referenced **Simulation** will be executed.

```
<listOfTasks>
    <task id="Task1" name="FlocksSimu" modelReference="model1"
         simulationReference="Simulation1">

    </task>
</listOfTasks>
```

```
<listOfTasks>
    <task id="Task2" name="Model_Default" modelReference="model1"
         simulationReference="Simulation_Default">
        <sourcescript id="task" language="text/x-r" >
            source("param.r")
            source("model.r")
       </sourcescript>
    </task>
</listOfTasks>
```

4.5. DataGenerator

The **DataGenerator** class defines which values from which tasks should be considered for output. The post-processing of values may also be defined in order to bring values in appropriate form for later output. It is possible to specify or reference scripts here as well.

4.6. Output

The **Output** class defines how the values specified in the Task or **DataGenerator** are plotted.

```
<listOfOutputs>
  <output id="plot1">
    <sourcescript id="script1" language="text/x-python" src="visualization-script.py" />
  </output>
  <output id="plot2">
    <sourcescript id="script2" language="text/x-r" >
       hist(result, breaks=50, main="PREVALENCE OF PARENTS FLOCKS", xlab="Prevalence",
            col="32")
       </sourcescript>
  </output>
</listOfOutputs>
```

Examples of the Output class with referenced and embedded scripts

4.7. Examples

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Written by Sascha Bulik,
     Guido Correia Carreia
     Miguel de-Alba-Aparicio,
     Carolina Plaza-Rodríguez,
     Matthias Filter
 -->
<sedFSKML xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://sed-ml.org/sed-ml-L1-V2.xsd"
        xmlns="http://sed-ml.org/sed-ml/level1/version2"
        level="1" version="2">

        <listOfModels>
                <model id="model1" name="initialize_parents_flocks">
                        <sourcescript id="model" language="text/x-r">
                                result <- 100 * rbeta(n.iter, shape1=Npos+1,
                                        shape2=Ntotal-Npos+1)
                        </sourcescript>
                </model>
        </listOfModels>

        <listOfSimulations>
                <fskSimulation id="Simulation1" name="script_settings" type="deterministic">
                        <sourcescript id="simulation1" language="text/x-r">
                                n.iter <- 200
                                Npos <- 30
                                Ntotal <- 100
                        </sourcescript>
                </fskSimulation>
        </listOfSimulations>

        <listOfTasks>
                <task id="Task1" name="FlocksSimu" modelReference="model1"
                        simulationReference="Simulation1">
                </task>
        </listOfTasks>

        <listOfOutputs>
                <output id="plot1">
                        <sourcescript id="visualization1" language="text/x-r">
                                hist(result, breaks=50,
                                        main="PREVALENCE AMONG FLOCKS IN PARENT GENERATION",
                                        xlab="Prevalence", col="32")
                        </sourcescript>
                </output>
        </listOfOutputs>
</sedFSKML>
```

5. Example models

5.1. Dose response example model

This example defines a parameterized dose-response model which gives the probability of infection after ingestion of PRRS virus by pigs for a given dose.

$$P(Infect|Dose) = 1 - \left(1 + \frac{Dose}{beta}\right)^{-alpha}$$

Here *alpha=0.3* and *beta=14400* (*log10 TCID50*) are PRRS and pig specific parameters. The equation represents a beta-poisson response function where the alpha describes the slope of the response curve and the value of beta shifts the curve along the abscissa (Dose axis).

This model has one variable and two parameters with the following default values:

- Input:

Variables	Dose = 4 $\log_{10}$ $TCID_{50}$
Parameter	Alpha = 0.3
	Beta = 14400 $\log_{10}$ $TCID_{50}$

- Output

Variables	P_Infect_Dose

Example metadata:

General Information	**Model Name**		Dose Response Model for Porcine Reproductive And Respiratory Syndrome Virus
	Source		PUBLICHED SCIENTIFIC STUDIES: Dose-response data and models
	Identifier		*Dose-Response_Brookes_Pork_001*
	Creator(s)	*vCard 4.0 standard*	V.J. Brookes; Charles Sturt University – Wagga Wagga; vbrookes@csu.edu.au
	Date	*Creation date*	10.29.2013
	Rights	*Rights*	Copyright © 2014 Elsevier B.V.
	Availability		*Public*
	URL		URI / URL of openFSMR or other model repository
	References	*Is_reference_description?*	*Yes*
		Publication type	*Journal*
		Publication date	2014 Mar 1
		PubMed ID	24502944
		Publication DOI	https://doi.org/10.1016/j.prevetmed.2014.01.016
		Publication Author List	Brookes VJ, Hernández-Jover M, Holyoake P, Ward MP.
		Publication Title	Import risk assessment incorporating a dose-response model: introduction of highly pathogenic porcine reproductive and respiratory syndrome into Australia via illegally imported raw pork.
		Publication Abstract	Highly pathogenic porcine reproductive and respiratory syndrome (PRRS) has spread through parts of south-east Asia, posing a risk to Australia. The objective of this study was to assess the probability of infection of a feral or domestic pig in Australia with highly pathogenic PRRS following ingestion of illegally imported raw pork. A conservative scenario was considered in which 500 g of raw pork was

			imported from the Philippines into Australia without being detected by border security, then discarded from a household and potentially accessed by a pig. Monte Carlo simulation of a two-dimensional, stochastic model was used to estimate the probability of entry and exposure, and the probability of infection was assessed by incorporating a virus-decay and mechanistic dose-response model. Results indicated that the probability of infection of a feral pig after ingestion of raw meat was higher than the probability of infection of a domestic pig. Sensitivity analysis was used to assess the influence of input parameters on model output probability estimates, and extension of the virus-decay and dose-response model was used to explore the impact of different temperatures and time from slaughter to ingestion of the meat, different weights of meat, and the level of viraemia at slaughter on the infectivity of meat. Parameters with the highest influence on the model output were the level of viraemia of a pig prior to slaughter and the probability of access by a feral pig to food-waste discarded on property surrounding a household. Extension of the decay and dose-response model showed that small pieces of meat (10 g) from a highly pathogenic PRRS viraemic pig could contain enough virus to have a high probability of infection of a pig, and that routes to Australia by sea or air from all highly pathogenic PRRS virus endemic countries were of interest dependent on the temperature of the raw meat during transport. This study highlighted the importance of mitigation strategies such as disposal of food-waste from international traffic as quarantine waste, and the need for further research into the probability of access to food-waste on properties by feral pigs.
		Publication Journal / Vol / Issue, etc.	Prev. Vet. Med. 113, 565-579
		Publication Status	*Accepted*
		Publication website	*http://www.sciencedirect.com/science/article/pii/S0167587714000178?via%3Dihub*
		Comment	*0:1*
	Language		*English*
	Software		R
	Programing language		*R/knime*
	Model category	*Model Class*	*Dose-response model*
		Model Sub-Class	*Other*
		Basic process	Infection with PRRS by ingestion of 0.5 kg of Pork (raw)
	Status		Uncurated
	Description		This is a dose-response model which describes the probability that pigs get infected with PRRSV. The model is based on a beta-poisson response function. It is considered to hold for a PRRSV dose from 0 to 8 $\log_{10}$ TCID50/ml in residual blood serum in raw pork.
Scope	**Product / matrix**	*Product/matrix name*	Swine Meat
		Product/matrix description	Pork raw meat
		Product/matrix unit	*Kg*
		Country of origin	Philippines
	Hazard	*Hazard type*	Microorganisms
		Hazard name	Porcine Reproductive and Respiratory Syndrome Virus (PRRS)
		Hazard description	PRRS is caused by an RNA virus that has a high mutation rate, allowing rapid gain of viral fitness when freely transmitted in large host populations.
		Hazard unit	$\log_{10}$ TCID50/ml
		Adverse effect	In 2006, a genetically distinct strain now known as “highly pathogenic PRRS virus” emerged in China, and caused 50–100% morbidity and 20–100% mortality in affected herds
		Source of contamination	Transmission of PRRS virus is possible via a variety of routes including orally by eating infected raw pork.
	Population Group	*Population name*	Feral or domestic pig
		Country	*Australia*
Model math / Data definition	**Parameter / Factor / Input / Output / "Data column"**	*Parameter ID*	*1* *2* *3* *4*
		Parameter classification	*1-Output* *2-Input-hazard* *3-Constant* *4-Constant*
		Parameter name	*1*- PInfectDose 2-Dose 3-Alpha 4-Beta
		Parameter description	*1.* PInfectDose is the probability that a pig gets infected given that it ingests a certain dose of PRRS virus. 2.Concentration of PRRS virus in residual blood in pork when ingested 3. Alpha = Parameter that describes the slope of the beta-poisson dose response function 4.Beta = Parameter which shifts the beta-poisson dose response function
		Parameter type	1.Number 2.Number 3.Number 4.Number
		Parameter unit	1. [] (no name)

			2.log10 TCID50 3. [] (no name) 4. log10 TCID50
		Parameter unit category	1. Dimensionless Quantity 2. Number 3. Dimensionless Quantity 2.Number
		Parameter data type	1.Double 2.Double 3.Double 4.Double
		Parameter source	3. Hermann et al. (2005) 4. Hermann et al. (2005)
		Parameter subject	0:1
		Parameter distribution	0:1
		Parameter value	2. 4 log10 TCID50 3. 0.3 4. 14400 log10 TCID50
	Model equation	*Model equation name*	Beta-poisson response function
		Model equation / Script	$P(Infect\|Dose) = 1 - \left(1+\frac{Dose}{beta}\right)^{-alpha}$
		Level of contamination after left-censored data treatment	4 log10 TCID50
		type of exposure	Acute
		Scenario	An individual in the Philippines acquires 500 g of raw pork that may be infected with highly pathogenic PRRS virus, and transports the meat by air from Manila to Darwin, Australia. On arrival at a private household, all the meat is discarded and may be eaten by a domestic or feral pig.

Table 8 Example metadata

Corresponding JSON encoded metadata to Table 8.

```
{
 "generalInformation": {
  "name": "Dose Response Model for Porcine Reproductive And Respiratory Syndrome Virus",
  "source": "PUBLICHED SCIENTIFIC STUDIES: Dose-response data and models",
  "identifier": "Dose-Response_Brookes_Pork_001",
  "creators": [
   {
    "creator": "[\"vcard\",[[\"version\",{},\"text\",\"4.0\"],[\"prodid\",{},\"text\",\"ez-vcard 
0.10.2\"],[\"nickname\",{},\"text\",\"V.J. Brookes\"],[\"email\",{},\"text\",\"vbrookes@csu.edu.au\"]]]"
   }
  ],
  "creationDate": 61372681200000,
  "modificationDate": [],
  "rights": "Copyright Ã‚Â© 2014 Elsevier B.V.",
  "isAvailable": true,
  "url": null,
  "format": null,
  "reference": [
   {
    "reference": "TY  - JOUR\r\nAB  - Highly pathogenic porcine reproductive and respiratory syndrome 
(PRRS) has spread through parts of south-east Asia, posing a risk to Australia. The objective of this 
study was to assess the probability of infection of a feral or domestic pig in Australia with highly 
pathogenic PRRS following ingestion of illegally imported raw pork. A conservative scenario was considered 
in which 500 g of raw pork was imported from the Philippines into Australia without being detected by 
border security, then discarded from a household and potentially accessed by a pig. Monte Carlo simulation 
of a two-dimensional, stochastic model was used to estimate the probability of entry and exposure, and the 
probability of infection was assessed by incorporating a virus-decay and mechanistic dose-response model. 
Results indicated that the probability of infection of a feral pig after ingestion of raw meat was higher 
than the probability of infection of a domestic pig. Sensitivity analysis was used to assess the influence 
of input parameters on model output probability estimates, and extension of the virus-decay and dose-
response model was used to explore the impact of different temperatures and time from slaughter to 
ingestion of the meat, different weights of meat, and the level of viraemia at slaughter on the 
infectivity of meat. Parameters with the highest influence on the model output were the level of viraemia 
of a pig prior to slaughter and the probability of access by a feral pig to food-waste discarded on 
property surrounding a household. Extension of the decay and dose-response model showed that small pieces 
of meat (10 g) from a highly pathogenic PRRS viraemic pig could contain enough virus to have a high 
probability of infection of a pig, and that routes to Australia by sea or air from all highly pathogenic 
PRRS virus endemic countries were of interest dependent on the temperature of the raw meat during 
transport. This study highlighted the importance of mitigation strategies such as disposal of food-waste
from international traffic as quarantine waste, and the need for further research into the probability of
access to food-waste on properties by feral pigs.\r\nAU  - Brookes VJ\r\nAU  - HernÃƒÂ¡ndez-Jover M\r\nAU
- Holyoake P\r\nAU  - Ward MP\r\nDA  - 2014-03-01\r\nDO  -
https://doi.org/10.1016/j.prevetmed.2014.01.016\r\nEP  - 579\r\nLK  -
http://www.sciencedirect.com/science/article/pii/S0167587714000178?via%3Dihub\r\nSP  - 565\r\nT2  - Prev.
Vet. Med.\r\nTI  - Import risk assessment incorporating a dose-response model: introduction of highly
pathogenic porcine reproductive and respiratory syndrome into Australia via illegally imported raw
pork.\r\nVL  - 113\r\nER  - \r\n\r\n"
    }
   ],
   "language": "English",
   "software": "R",
   "languageWrittenIn": "R/KNIME",
   "modelCategory": {
    "modelClass": "Dose-response model",
    "modelSubClass": [
     "Other"
    ],
    "modelClassComment": null,
    "modelSubSubClass": [],
    "basicProcess": [
     "Infection with PRRS by ingestion of 0.5 kg of Pork (raw)"
    ]
   },
   "status": "Uncurated",
   "objective": null,
   "description": "This is a dose-response model which describes the probability that pigs get infected
with PRRSV. The model is based on a beta-poisson response function. It is considered to hold for a PRRSV
dose from 0 to 8 log10 TClD50/ml in residual blood serum in raw pork."
  },
  "scope": {
   "product": {
    "environmentName": "Swine Meat",
    "environmentDescription": "Pork raw meat",
    "environmentUnit": "Kg",
    "productionMethod": [],
    "packaging": [],
    "productTreatment": [],
    "originCountry": "Philippines",
    "originArea": null,
    "fisheriesArea": null,
    "productionDate": null,
    "expirationDate": null
   },
   "hazard": {
    "hazardType": "Microorganisms",
    "hazardName": "Porcine Reproductive and Respiratory Syndrome Virus (PRRS)",
    "hazardDescription": "PRRS is caused by an RNA virus that has a high mutation rate, allowing rapid gain
of viral fitness when freely transmitted in large host populations.",
    "hazardUnit": null,
    "adverseEffect": "In 2006, a genetically distinct strain now known as Ã¢â‚¬Å“highly pathogenic PRRS
virusÃ¢â‚¬ï¿½ emerged in China, and caused 50Ã¢â‚¬â€œ100% morbidity and 20Ã¢â‚¬â€œ100% mortality in
affected herds",
    "origin": "Transmission of PRRS virus is possible via a variety of routes including orally by eating
infected raw pork.",
    "benchmarkDose": null,
    "maximumResidueLimit": null,
    "noObservedAdverse": null,
    "lowestObservedAdverse": null,
    "acceptableOperator": null,
    "acuteReferenceDose": null,
    "acceptableDailyIntake": null,
    "hazardIndSum": null,
    "laboratoryName": null,
    "laboratoryCountry": null,
    "detectionLimit": null,
    "quantificationLimit": null,
    "leftCensoredData": null,
    "rangeOfContamination": null
   },
  "populationGroup": {
   "populationName": "Feral or domestic pig",
   "targetPopulation": null,
   "populationSpan": [],
   "populationDescription": [],
   "populationAge": [],
   "populationGender": null,
   "bmi": [],
   "specialDietGroups": [],
   "patternConsumption": [],
   "region": [],
   "country": [
    "Australia"
   ],
   "populationRiskFactor": [],
   "season": []
  },
  "generalComment": null,
  "temporalInformation": null,
  "region": [],
  "country": []
 },
 "dataBackground": null,
 "modelMath": {
  "parameter": [
   {
    "id": "1",
    "classification": "output",
    "name": "PInfectDose",
    "description": "PInfectDose is the probability that a pig gets infected given that it ingests a
certain dose of PRRS virus.",
    "unit": "[] (no name)",
    "unitCategory": "Dimensionless Quantity",
    "dataType": "Double",
    "source": null,
    "subject": null,
    "distribution": null,
    "value": null,
    "reference": null,
    "variabilitySubject": null,
    "modelApplicability": [],
    "error": null
   },
   {
    "id": "2",
    "classification": "input",
    "name": "Dose",
    "description": "Concentration of PRRS virus in residual blood in pork when ingested",
    "unit": "log10 TCLD50",
    "unitCategory": "Number",
    "dataType": "Double",
    "source": null,
    "subject": null,
    "distribution": null,
    "value": "4 log10 TCLD50",
    "reference": null,
    "variabilitySubject": null,
    "modelApplicability": [],
    "error": null
   },
   {
    "id": "3",
    "classification": "constant",
    "name": "Alpha",
    "description": "Alpha = Parameter that describes the slope of the beta-poisson dose response
function",
    "unit": "[] (no name)",
    "unitCategory": "Dimensionless Quantity",
    "dataType": "Double",
    "source": "Hermann et al. (2005)",
      "subject": null,
      "distribution": null,
      "value": "0.3",
      "reference": null,
      "variabilitySubject": null,
      "modelApplicability": [],
      "error": null
     },
     {
      "id": "4",
      "classification": "constant",
      "name": "Beta",
      "description": "Beta = Parameter which shifts the beta-poisson dose response function",
      "unit": "log10 TCLD50",
      "unitCategory": "Number",
      "dataType": "Double",
      "source": "Hermann et al. (2005)",
      "subject": null,
      "distribution": null,
      "value": "14400 log10 TCLD50",
      "reference": null,
      "variabilitySubject": null,
      "modelApplicability": [],
      "error": null
     }
    ],
    "sse": null,
    "mse": null,
    "rmse": null,
    "rSquared": null,
    "aic": null,
    "bic": null,
    "modelEquation": {
     "equationName": "Beta-poisson response function",
     "equationClass": null,
     "equationReference": [],
     "equation": "PInfectDose = 1 - (1 + Dose/Beta) ^ (-Alpha)"
    },
    "fittingProcedure": null,
    "exposure": null,
    "event": []
   },
  }
```

References

Bergmann, F., Adams, R., Moodie, S., Cooper, J., Glont, M., Golebiewski, M., et al. (2014). *COMBINE archive and OMEX format: one file to share all information to reproduce a modeling project.* (B. Bioinformatics, Ed.) Retrieved from http://www.ncbi.nlm.nih.gov/pubmed/25494900

Bergmann, F., Cooper, J., Le Novère, N., Nickerson, D., & Waltermath, D. (2015). Simulation Experiment Description Markup Language (SED-ML) Level 1 Version 2. *Journal of intergrative Bioinformatics*, 262.

Bergmann, F., Rodriguez, N., & Le Novère, N. (2015). *COMBINE Archive Specification Version 1.* Retrieved from http://www.ncbi.nlm.nih.gov/pubmed/26528559

Cyganiak, R., Wood, D., Lanthaler, M., Klyne, G., Carroll, J. J., & and McBride, B. (2014). *RDF 1.1 concepts and abstract syntax.* Retrieved from https://www.w3.org/TR/rdf11-concepts/

Freed, N. a. (1996). *Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types.* Retrieved from www.iana.org/assignments/media-types/media-types.xhtml

Hucka, M., Bergmann, F. T., Hoops, S., Keatin, S., Sahle, S., Schaff, J., et al. (2010). *The Systems Biology Markup Language (SBML): Language Specification for Level 3 Version 1 Core.* Retrieved from http://identifiers.org/combine.specifications/sbml.level-3.version-1.core.release-1

O' Dada, J. (2011). *Numerical Markup Language.* Retrieved from http://code.google.com

OMG. (2009, February). *UML 2.2 Superstructure and Infrastructure.* Retrieved from http://www.omg.org/spec/uml/2.2

Waltemath, D., Adams, R., Beard, D., Bergmann, F., Bhalla, U., Britten, R., et al. (2011). *Minimum Information About a Simulation Experiment (MIASE).* Retrieved from http://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1001122