PROSIT
Disease Modelling
Community

PROSIT Technical Handbook

PROSIT Technical Document

Version 1.0

20.07.2012

Edited by: Wendelin Schramm and Christian Weiß

http://www.prosit.de

Preamble

The PROSIT Technical Handbook has been written to help PROSIT community members to find themselves through a disease model.

Its main goal is to provide the readers with the information to comprehend the model's logic in the numerous sheets of the file. At first sight it seems to be very difficult to understand, but keeping in mind that inputs are computed to produce outputs it is then not so hard whatsoever.

A basic knowledge of spread sheet programs and on the math of Markov models is presupposed. In case of questions on Open Office the reader is directed to the excellent online resources of the Open Office community on the internet. Excellent publications exist to t

First, the section *General remarks* guides the reader to basic issues such as how to get a model running or cell referencing.

The following sections describe the structure of a model and most important how results are computed within the model.

We do hope that readers find this guidance though short and in telegram style helpful to their work. Feedback is warmly welcomed for improving future versions of this document.

Heilbronn and Steinen, July 2012

Wendelin Schramm, Christian Weiß

How to cite this Handbook:

PROSIT Disease Modelling Community. PROSIT Publication Technical Handbook 1.0 [monograph on the Internet]. Heilbronn: Heilbronn University; July 2012 [cited 201X XYmonth XYday]. Available from: http://www.prosit.de/index.php/Technical_Handbook

Contents

Title of a PROSIT Disease Model

Why is a PROSIT model named as it is named? The main advantage of the versioning of PROSIT models is that any study or publication performed with a PROSIT model can easily be evaluated.

- It is custom for international peer-review publications in health-economics to provide information about patient characteristics and cost data. With this information and the correct downloaded model version any cost/effectiveness results should reproducible to readers and reviewers. This is a unique feature of the PROSIT Community models.
- Older publications can be assessed as all previous versions of a PROSIT model are available for download here. New models are cross-checked with older PROSIT model versions in order to see changes being introduced over time.

Here is the systematic behind the versioning:

EXAMPLE OFFICIAL NAME:

```
PROSIT Nephropathy Type 2 Diabetes Disease Model 2008A Romania
```

EXAMPLE SHORT NAME:

```
Danube 0.9A Germany
```

River names: are the code names or nick names for models with a defined structure. "Danube" is the first code name as from the very beginning a German and a Romanian country version existed. Two countries situated "on the shores" of river Danube. Thus river names are getting an official status. River names are only assigned by the forum administrators Wendelin Schramm and Monika Pobiruchin.

Capital letters: The current nephropathy model is called "Danube A" in order to specify that a defined set of transition probabilities has been used within the defined "Danube" model structure. If a new model uses the same model structure, but new algorithms were introduced, than a new capital letter will be assigned ("Danube B", "Danube C", etc.). These capital letters are getting an official status. Letters are therefore only assigned by the forum administrators Wendelin Schramm and Monika Pobiruchin.

Country versions: for use in different communities/countries the important mortality tables have to be replaced. Therefore, a lot of country versions of a model may exist. A country version is always labelled with the full country name (e.g. Romania, France, Austria, etc.).

Release versions: After reaching the 1.0 model version status, once a year an official release version is made public. These yearly releases carry the number of the year in their name.

Development versions: The PROSIT models are developed continuously. This means, development or interim versions of the model exist. These are labelled with numbers smaller than one (e.g. Danube A 0.9). Again these numbers have an official status. Version numbers are therefore only assigned by the forum administrators Wendelin Schramm and Monika Pobiruchin.

What is the reasoning for the name "PROSIT"?

latin (prosit = it shall be useful)

PROSIT: this has been for 13 years the (brand) name of one of the most successful medical quality management programs in Germany. PROSIT had a focus on nephropathy in diabetes patients. Therefore a direct medical link exists to our nephropathy model. After stopping the project in 2005 the web site and the brand name became free again and were donated to the open source community. If someone breaches the GNU public license, the brand name PROSIT allows the community to protect their work against unjustified abuse.

General remarks

How to get a PROSIT Model running?

1. First remember: the model is not a stand-alone software. It needs a spread sheet programme capable of reading and writing the Open Document Format ODS to run.
2. The most common software package and our development tool is OpenOffice.org (OOo). OOo like the PROSIT model is open source software. You can get it for free on the internet (e.g. at the international OpenOffice.org Community; or here for German version) and you will find it on the give-away CD-ROMs of most computer magazines. Many infos on OpenOffice are given here (in English) and here (in German)
3. Download the desired PROSIT model to your computer
4. Open the file. users of Versions < 2.3 most likely will be asked whether to activate the macros: Answer with YES.
5. **Users of OpenOffice.org 2.3 and later versions:** The security question does not appear. You have to change the makro security level of OpenOffice.org (Tools/Extras->Options->Security). The security level should be set to "medium", i.e. the program always asks you before allowing macros to run.
6. **Here you are:** the model should be running, if not post a message to the administrators.
7. We do not know of a security bug while using Open Office's macros. Nevertheless, to be on the safe side, always download PROSIT models from our web site. Do not load PROSIT models from other sites.

Get OpenOffice.org !

It should be stated though that a lot more programmes exist with support for the Open Document File format. We have not reached a level for systematic testing and validating the interoperability of PROSIT models with other software than OpenOffice.org Calc.

Cell referencing

PROSIT models make heavy use of referencing named cells. These names indicate either single cells or cell arrays (tables). These names are unique for the Calc file. One can easily jump to a named cell/array by clicking on the label in the drop-down list in the left-upper corner of the spread sheet (Figure 1).

Figure 1: drop-down list with labels used to reference cells or arrays

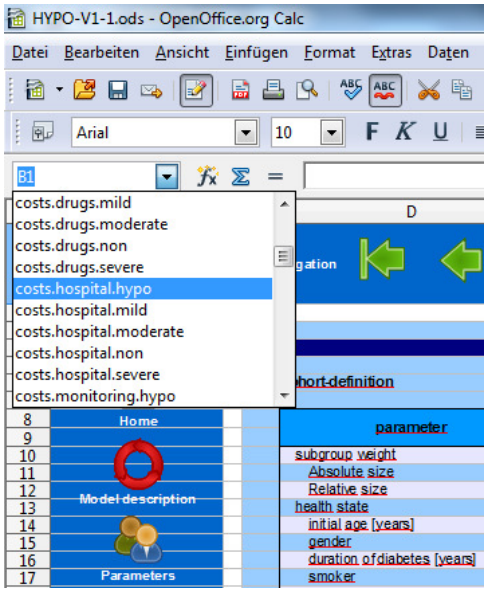
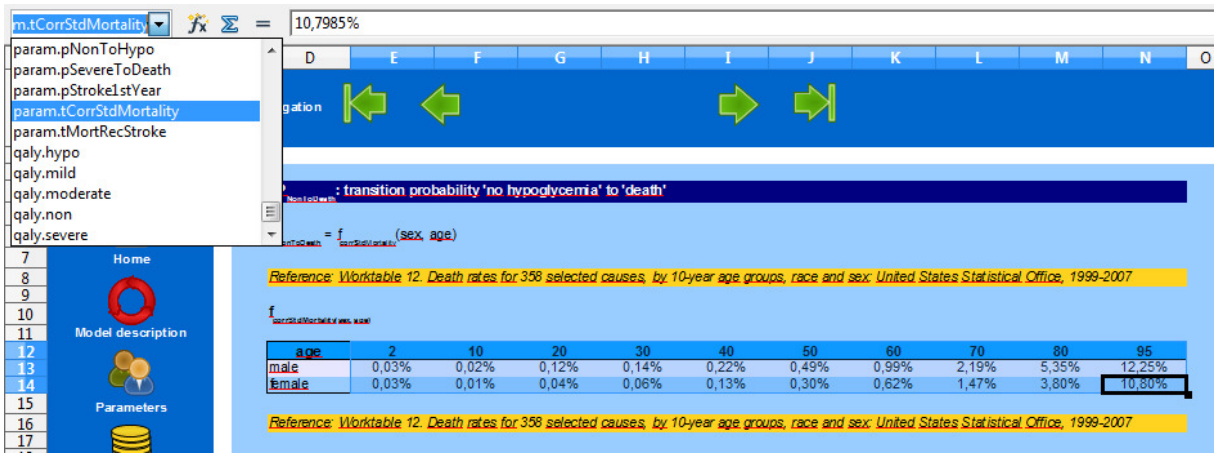


Figure 2: reference to an array



Time Horizon

All PROSIT diabetes models calculate over 80 cycles, normally each representing one year. It would be feasible to define cycles for instance as quarters, which then reduces the time horizon to 20 years (80/4 = 20). Using monthly cycles results in 6.7 years (80/12 = 6.666). The hypoglycaemia model is an exception: there, up to four hypoglycaemic events can be computed for each of the 80 cycles.

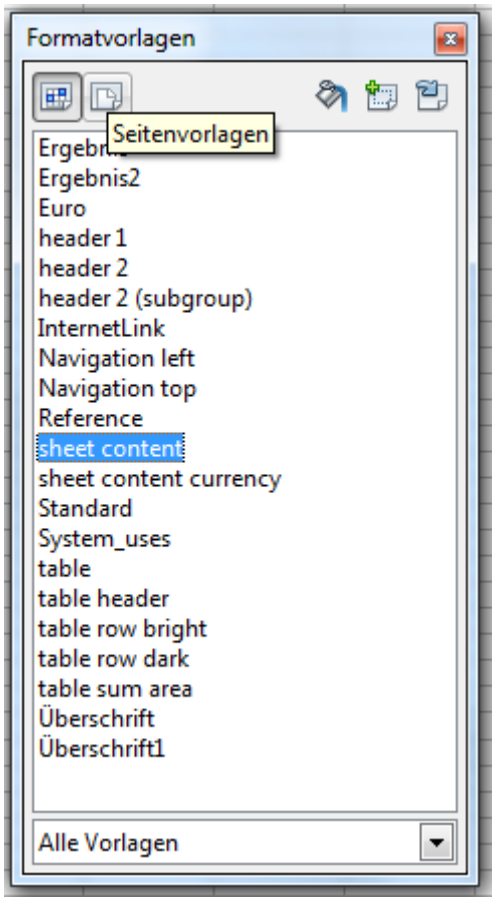
Click behaviour

If a mouse click on an icon e.g. in the navigation bar should not work. Then click into a free area in the sheet and then try it again.

Templates

The PROSIT models makes use of templates to format the appearance of the model. So if users make changes to the model, they should not forget about this feature. Changing one of the templates changes the “look” for the rest of the model regardless of a specific single sheet.

Figure 3: the template palette



The template palette can be called up with the **F11 KEY** or the button left to the font selection in the formatting menu.

Figure 4: Starting the template palette

Interpolation

Sometimes interpolation of data values within a defined range is helpful to achieve a better precision in the computations than realised with the last observation carried forward approach.

As Open Office does not know an interpolation function one was developed by Christian Weiß (herr.christian.weiss@googlemail.com) and the code is implemented in most PROSIT models using Calc's *OpenOffice.Basic* macro functions.

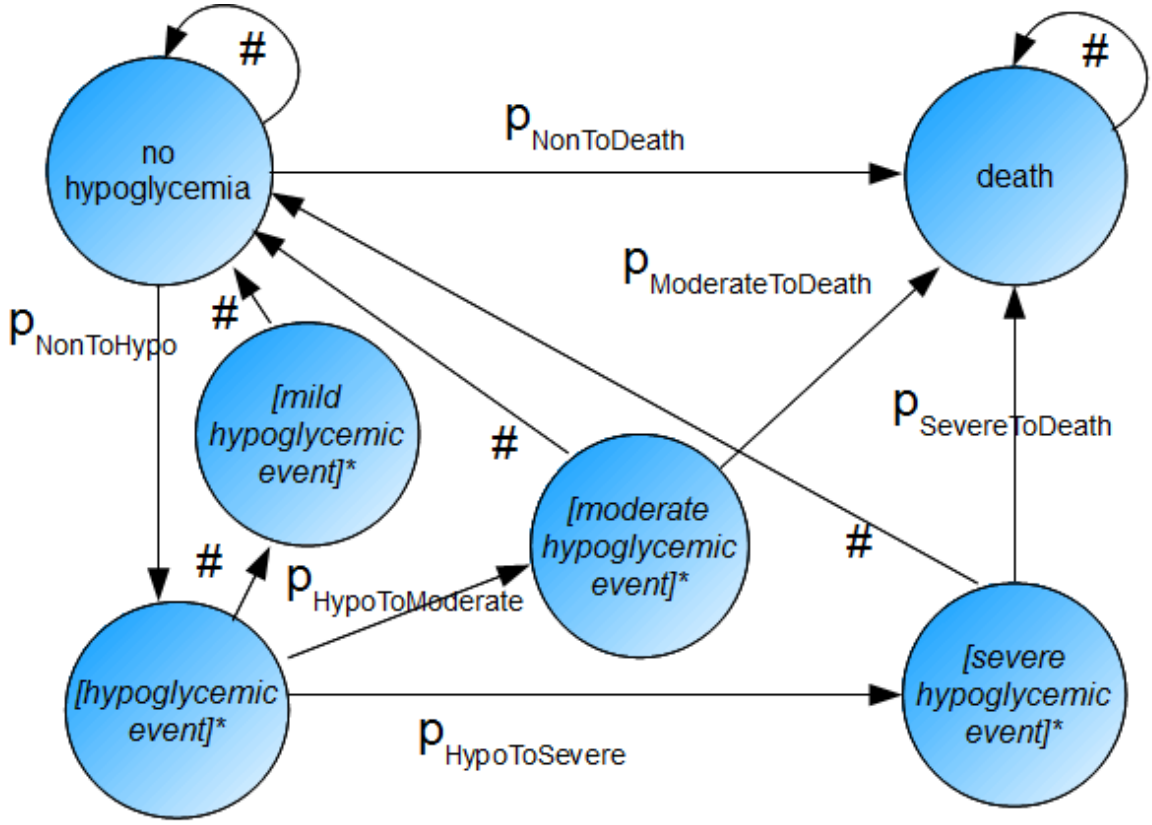
Additional program code has been implanted to realise the look-up of interpolated values from either a horizontal (WINTERPOLATION) or vertical (SINTERPOLATION) data table that consists of two columns and a flexible number of rows or vice-versa. A sample file on the PROSIT homepage illustrates the functions.

All PROSIT Models try to avoid advanced programming and macros except for convenience features like navigation by clicking on icons apart from this one exception.

Structure of a PROSIT Disease Model

The models have a sheet **MODELOVERVIEW**. There a state transition diagram illustrates the basic composition of the model structure. It shows the model's disease states ("bubbles"), the transitions (arrows) and the labels of the transition probabilities (e.g. p NonToDeath). The Hash sign "#" indicates probabilities that are calculated as the remainder to 100% of the sum of the other probabilities leaving a disease state. Squared brackets "[ ]" indicate states that can be repeated during the computation: e.g. so called tunnel states.

Figure 5: state transition diagram



*** temporary state**
 # = 1.0 – sum of other transitions leaving this state

How transitions are calculated in the Markov models

Preface: a basic knowledge of the use and of the math of Markov models for disease modelling in medicine are presupposed. In case of questions excellent publications and text books exist to support the user (1-3). Twice a year the PROSIT Community holds workshops to demonstrate how it may work.

The tables are numerous in a PROSIT model, but the logic is simplified to allow easy bug detection and maintenance of the models.

Input

In the **INPUT SHEET** basic characteristics of up to 16 patient groups can be defined and other variables are set for the model calculations.

The 16 groups are used to calculate different risks within a cohort. Frequently males and females or smokers and non-smokers differ in the rates of certain events. And, may be smoking females are different from non-smoking males. Provided that sufficient data is available subgroups now can be defined in the manner illustrated in Table 1. According to the scheme $2^4 = 16$ combinations are possible. Apart from these differences the absolute or relative weight of a subgroup can be different and this can be defined also in the patients' subgroup table of the **INPUT SHEET**. It would be possible to implement more subgroups, but that would increase the size of the model tables enormously.

Table 1: Defining subgroups for the model

male	male	male	male	female	female	female	female
smoker	smoker	non-smoker	non-smoker	smoker	smoker	non-smoker	non-smoker
atrial fibrillation	no fibrillation	atrial fibrillation	no fibrillation	atrial fibrillation	no fibrillation	atrial fibrillation	no fibrillation

In the **INPUT SHEET** QALYs can be entered for the defined disease states. Also discount rates for costs and effects can be entered here.

"p-Tables"

The sheet **OVERVIEWPROBABILITES** lists all transitions of the disease model. By definition in a model with n states n^2 transitions are possible (e.g. 5 states makes 25 transitions), but as many transitions are not necessary the number of transitions is determined by the medical nature of the condition or certain treatment capabilities.

Figure 6: Probabilities Overview

All sheets labelled with a small letter **P** in their names are used to define probabilities used in the model. These may be constant values, tables, or formulas. The probabilities are illustrated also in the state transition diagram (

Figure 5 on page 11).

Probability Subtraces

The table **PROBABILITYSUBTRACES** is a long helper table (scroll down or zoom-out to see the entire dimension!) collecting for each of the 16 subgroups the probabilities for all cycles (time slices) of the model.

This table looks impressive and numerous formulas are used, yet the logic is simple. For all conditions of a patient subgroup the corresponding probability is determined from the **P SHEETS** and noted down. The regular composition of the patient characteristics table allows simple repetition over all 15 other subgroups.

Normally, you can find the formulas used by the model here. The table also builds up on numerous *look-ups* and *if,then,else* statements.

State Subtraces

The table **STATESUBTRACES** has the same style of the previous **PROBABILITYSUBTRACES** table. Now here the number/proportion of the patients under risk is multiplied with the probabilities for each sub-group and for each cycle of the model.

State Traces

The **STATETRACES** sheet does not have 16 subgroups any more: just one table that adds up the results of the 16 subgroups of the **STATESUBTRACES**.

The **STATETRACESHALFSTEP** sheet applies a half-cycle correction to the results, if needed (see also the corresponding section on page 15). Thus, the model has two result tables as not all results should be adjusted by a half-cycle correction(3).

Markov table

The Markov table shows results from the **STATETRACES** or **STATETRACESHALFSTEP** sheet, but also adds up values to produce cumulative results, which are then used by the model's diagrams.

Half-cycle correction

The half-cycle correction is implemented as the mean of the addition of the results of two following cycles in the manner (StateTrace.previous.cycle+StateTrace.following.cycle/2.

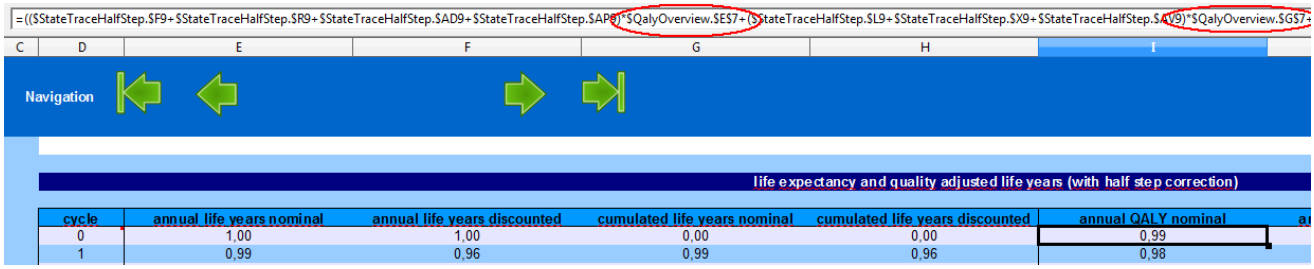
Thus the time point to collect the results of the computation is shifted from the end of a cycle to the middle.

Earlier PROSIT models halved the first and last cycle. This procedure has been abandoned since the 2012 relaunch of the PROSIT models.

QALYS and their implementation

QALYs are easily implemented by multiplying the number/proportion of patients in certain disease states by the QALY discount value defined previously in the **INPUT SHEET**.

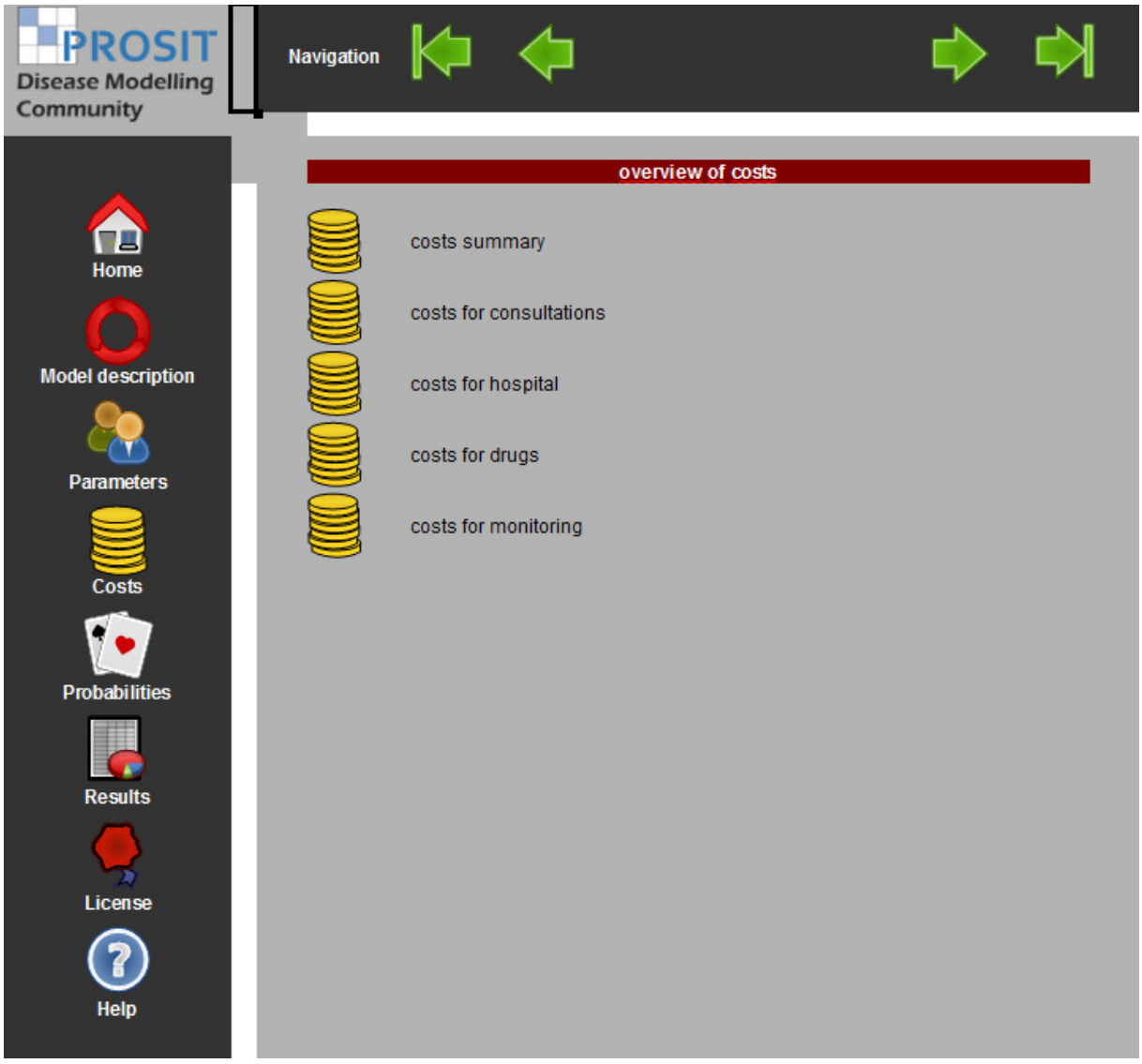
Figure 7: Implementing QALYs



Handling of Costs

PROSIT models by default are prepared for the data entry of costs. All data in a native model are sample data reflecting the great variation of how to document and calculate costs in different sectors of the health care system and from different viewpoints of the involved stakeholders. Cost tables can be adapted easily to the specific need of a health-economic study. An **OVERVIEWCOSTS** section lists the cost categories: monitoring, consultations, drugs/treatments, hospital spending (Figure 8).

Figure 8: Cost overview



The user has to respect the *summary fields* for each costing category indicated by defined labels following the scheme *label.category.item* (see Figure 1 on page 7). The labels must not be overwritten by user input, or the costs will not be calculated correctly.

In principal, the cost sheets calculate the allocation of monetary resources in the manner:

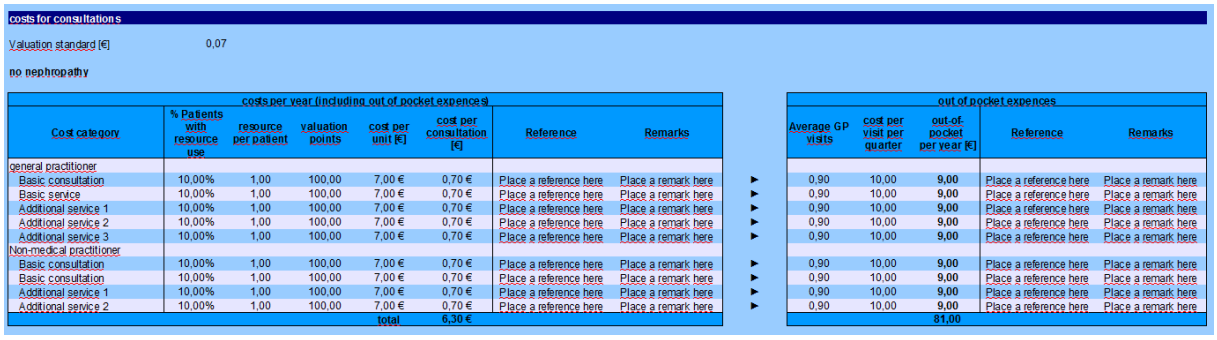
Number/per cent of patients receiving n-times a resource that has a certain price per model cycle (Figure 9).

$$Costs\ per\ cycle = patients \bullet frequency \bullet price$$

A **COSTSUMMARY** sheet provides a compilation of the overall costs per category taken from the detailed cost sheets.

Out of pocket expenses of patients are also recorded in the cost sheets.

Figure 9: cost sheets calculate the economic input to the model



costs for consultations

Valuation standard [€] 0,07

no nephropathy

	costs per year (including out of pocket expences)								out of pocket expences				
Cost category	% Patients with resource use	resource per patient	valuation points	cost per unit [€]	cost per consultation [€]	Reference	Remarks		Average GP visits	cost per visit per quarter	out-of-pocket per year [€]	Reference	Remarks
general practitioner													
Basic consultation	10,00%	1,00	100,00	7,00 €	0,70 €	Place a reference here	Place a remark here	►	0,90	10,00	9,00	Place a reference here	Place a remark here
Basic service	10,00%	1,00	100,00	7,00 €	0,70 €	Place a reference here	Place a remark here	►	0,90	10,00	9,00	Place a reference here	Place a remark here
Additional service 1	10,00%	1,00	100,00	7,00 €	0,70 €	Place a reference here	Place a remark here	►	0,90	10,00	9,00	Place a reference here	Place a remark here
Additional service 2	10,00%	1,00	100,00	7,00 €	0,70 €	Place a reference here	Place a remark here	►	0,90	10,00	9,00	Place a reference here	Place a remark here
Additional service 3	10,00%	1,00	100,00	7,00 €	0,70 €	Place a reference here	Place a remark here	►	0,90	10,00	9,00	Place a reference here	Place a remark here
Non-medical practitioner													
Basic consultation	10,00%	1,00	100,00	7,00 €	0,70 €	Place a reference here	Place a remark here	►	0,90	10,00	9,00	Place a reference here	Place a remark here
Basic consultation	10,00%	1,00	100,00	7,00 €	0,70 €	Place a reference here	Place a remark here	►	0,90	10,00	9,00	Place a reference here	Place a remark here
Additional service 1	10,00%	1,00	100,00	7,00 €	0,70 €	Place a reference here	Place a remark here	►	0,90	10,00	9,00	Place a reference here	Place a remark here
Additional service 2	10,00%	1,00	100,00	7,00 €	0,70 €	Place a reference here	Place a remark here	►	0,90	10,00	9,00	Place a reference here	Place a remark here
				total	6,30 €						81,00		

How to perform a sensitivity analysis

It is mandatory when using Markov disease models to perform sensitivity analyses. One-way sensitivity analysis is commonly accepted (3). In rare cases, where unclear sensitivity relations exist a two-way sensitivity analysis should be performed.

Basically, all influencing variables should be modified by changing them within a fixed range of e.g. ±10% (which is the most frequent way to do this). The procedure is simple

1. Define a base case scenario and calculate all results and note them.

2. One after the other, change the influencing inputs, variables and probabilities by +10%. Then calculate all results of step 1 and note them.
3. Repeat step 2 with -10% and record the results.
4. Create a common chart with the results of step 1 to 3 and draw tornado diagrams illustrating either the absolute or the relative change of variables for the modelling outcomes.

Results in this context are the medical outcomes (e.g. number events over the time horizon), the cost results and the combined health-economic results such as the CLYGs.

References

1. Briggs A, Sculpher M. An introduction to Markov modelling for economic evaluation. Pharmacoeconomics. 1998;13(4):397-409.

2. Sonnenberg FA, Beck JR. Markov models in medical decision making: a practical guide. Med Decis Making. 1993;13(4):322-38.

3. Gold MR, Siegel JE, Russel LB, Weinstein MC. Cost-effectiveness in health and medicine. NY: Oxford University Press; 1996.

Appendix

Code of Linear Interpolation

All PROSIT Models try to avoid advanced programming and macros except for convenience features like navigation by clicking on icons.

There is one exception.

As OpenOffice does not know an interpolation function it was developed by Christian Weiß (herr.christian.weiss@googlemail.com) and the code is implemented in most PROSIT models using Calc's *OpenOffice.Basic* macro functions. Following code has been implanted to realise the look-up of interpolated values from either a horizontal (WINTERPOLATION) or vertical (SINTERPOLATION) data table that consists of two columns and a flexible number of rows or vice-versa two rows and a flexible number of columns:

SINTERPOLATION

Interpolation of a vertical table.

```
'-------------------------------------------------------------------------


FUNCTION SINTERPOLATION(searchValue As Double, s As Range, sourceColumn As Long, OPTIONAL
mode AS INTEGER = 1)


  DIM result AS DOUBLE

  DIM numberOfRows AS LONG

  DIM numberOfColumns AS LONG

  DIM lowerIndex AS LONG

  DIM upperIndex AS LONG
```

```
DIM i AS LONG

numberOfRows    = UBound(s,1) - LBound(s,1) + 1
numberOfColumns = UBound(s,2) - LBound(s,2) + 1
lowerIndex = LBound(s, 1)
upperIndex = LBound(s, 1)
Result = 0.0

IF ((numberOfColumns > 0) AND (numberOfRows >0) AND (sourceColumn <= numberOfColumns))
THEN

  FOR i=LBound(s, 1) to UBound(s, 1)

    IF s(lowerIndex, 1) > searchValue THEN
      IF s(i, 1) < s(lowerIndex, 1) THEN
        lowerIndex = i
      ENDIF
    ELSE
      IF (s(i, 1) <= searchValue) AND (s(i, 1) > s(lowerIndex, 1)) THEN
        lowerIndex = i
      ENDIF
    ENDIF
```

```
 IF s(upperIndex, 1) < searchValue THEN
  IF (s(i, 1) > s(upperIndex, 1)) THEN
   upperIndex = i
  ENDIF
 ELSE
  IF (s(i, 1) >= searchValue) AND (s(i, 1) < s(upperIndex, 1)) THEN
   upperIndex = i
  ENDIF
 ENDIF

NEXT i

IF searchValue > s(upperIndex, 1) THEN
 result = s(upperIndex, sourceColumn)
ELSEIF searchValue < s(lowerIndex, 1) THEN
 result = s(lowerIndex, sourceColumn)
ELSEIF s(upperIndex, 1) = s(lowerIndex, 1) THEN
 result = s(lowerIndex, sourceColumn)
ELSE
 SELECT CASE mode
  CASE 0
   result = s(lowerIndex, sourceColumn)
  CASE 1
```

```
        result = s(lowerIndex, sourceColumn) + (s(upperIndex, sourceColumn) - s(lowerIndex,
sourceColumn))/(s(upperIndex, 1) - s(lowerIndex, 1)) * (searchValue - s(lowerIndex, 1))

      CASE ELSE

        result = CVERR(1)

    END SELECT

  END IF


  SINTERPOLATION = result

ELSE

  SINTERPOLATION = "#Error"

ENDIF


END FUNCTION
```

WINTERPOLATION

Interpolation of a horizontal table.

```
'-------------------------------------------------------------------------------


FUNCTION SINTERPOLATION(searchValue As Double, s As Range, sourceColumn As Long, OPTIONAL
mode AS INTEGER = 1)
```

```
DIM result AS DOUBLE

DIM numberOfRows AS LONG

DIM numberOfColumns AS LONG

DIM lowerIndex AS LONG

DIM upperIndex AS LONG

DIM i AS LONG


numberOfRows    = UBound(s,1) - LBound(s,1) + 1

numberOfColumns = UBound(s,2) - LBound(s,2) + 1

lowerIndex = LBound(s, 1)

upperIndex = LBound(s, 1)

Result = 0.0


IF ((numberOfColumns > 0) AND (numberOfRows >0) AND (sourceColumn <= numberOfColumns))
THEN


  FOR i=LBound(s, 1) to UBound(s, 1)


    IF s(lowerIndex, 1) > searchValue THEN

      IF s(i, 1) < s(lowerIndex, 1) THEN

        lowerIndex = i

      ENDIF

    ELSE
```

```
  IF (s(i, 1) <= searchValue) AND (s(i, 1) > s(lowerIndex, 1)) THEN
    lowerIndex = i
  ENDIF
 ENDIF

 IF s(upperIndex, 1) < searchValue THEN
  IF (s(i, 1) > s(upperIndex, 1)) THEN
    upperIndex = i
  ENDIF
 ELSE
  IF (s(i, 1) >= searchValue) AND (s(i, 1) < s(upperIndex, 1)) THEN
    upperIndex = i
  ENDIF
 ENDIF

NEXT i

IF searchValue > s(upperIndex, 1) THEN
 result = s(upperIndex, sourceColumn)
ELSEIF searchValue < s(lowerIndex, 1) THEN
 result = s(lowerIndex, sourceColumn)
ELSEIF s(upperIndex, 1) = s(lowerIndex, 1) THEN
 result = s(lowerIndex, sourceColumn)
```

```
  ELSE
    SELECT CASE mode
      CASE 0
        result = s(lowerIndex, sourceColumn)
      CASE 1
        result = s(lowerIndex, sourceColumn) + (s(upperIndex, sourceColumn) - s(lowerIndex,
sourceColumn))/(s(upperIndex, 1) - s(lowerIndex, 1)) * (searchValue - s(lowerIndex, 1))
      CASE ELSE
        result = CVERR(1)
    END SELECT
  END IF

  SINTERPOLATION = result
ELSE
  SINTERPOLATION = "#Error"
ENDIF

END FUNCTION
```