Guideline for building cost-effectiveness models in R

1 Introduction

Microsoft Excel is currently the dominant software to build cost-effectiveness (CE) models within health technology assessments (HTA). Excel has many advantages, since it is an easily understandable, intuitive program with abundant features and documentation necessary for CE analysis. Additionally, HTA organizations have extensive experience with Excel. In some cases, Excel may suffice as the software of choice, but as the complexity of CE-models is increasing, the cases when Excel is impractical or even infeasible for the purpose of CE models are becoming more common. Limitations to the use of Excel for decision modelling have been well documented, and include the increasing use of probabilistic elements, the difficulties with having many separate but linked sheets, the introduction of increasingly complex Visual Basic for Applications (VBA) code, and the more complex demands that are required in terms of statistical calculations and visualizations.[1, 2] To remain future-proof, HTA organizations need to facilitate options beyond Microsoft Excel.

An often mentioned alternative to Excel is R.[2, 1] R is already used by many people for the purpose of building CE-models. It is advocated as the software of choice due to its possibilities to integrate statistical analyses, state-of-the-art decision modeling methods and proper documentation. When well executed, models in R have the potential to be more efficient, more transparent, more adaptable, and better reusable than models in Excel.

This document outlines guidance for CE-model building and submission in R for reimbursement requests to Zorginstituut Nederland (National Health Care Institute; ZIN). This guideline is mainly based on published work from two consortia in this field: DARTH (Decision Analysis in R for Technologies in Health) and the 'R for HTA' consortium.

2 Guidance

2.1 Purpose of this guidance

The freedom in model building inherent to R may lead to complications if not properly managed. To ensure that CE-models in R are transparent, valid, auditable, adaptable, and generally fit-for-purpose, some measure of standardization is considered necessary. The purpose of this guideline is to set the conditions that CE-models built in R and meant for HTA decision-making need to fulfill. Budget impact analyses are outside the scope if this guideline. This guideline is complementary to existing ZIN guidance on performing economic evaluations and is not meant to replace any existing guidance. When recommendations are conflicting, the ZIN general guidance for economic evaluations should be followed.

2.2 General principles for CE-models in R

General principles for models in R are essentially equivalent to models in any other type of software and include that the model is:

- Appropriately structured;
- Entirely transparent;
- Sufficiently validated;
- Fully executable and adaptable;
- Clearly and comprehensively documented.

It should be checked whether users other than the original author can run the model without errors. The model should facilitate that assessors can easily change parameters and rerun analyses in a Windows environment.

2.3 End-to-end functionalities

R provides the benefit that it facilitates end-to-end functionality, meaning that all statistical analyses usually performed separately from the CE-model can be coded in R. An example is survival curve extrapolations which cannot be performed in Excel. The current position of ZIN is that submissions with end-to-end functionality are encouraged but not obligatory. CE-models in R are required to include at least all functionalities that would otherwise be included in Excel.

2.4 Data requirements

All data necessary should be included with the submission in order to enable ZIN to run the model. Data may be aggregated to a level as it would also be aggregated within a model built in Excel (for example individual patient costing data is generally not required). In line with the options for end-to-end functionalities, data are only required to the extent as it would also be required for a model built in Excel.

2.5 Folder structure and readme

A 'readme' file or metadata file should be provided to serve as the starting point for users. These files should explain at least the folder structure, the purpose of the files within each folder and how these files are linked, the coding convention that was used, how errors are handled, and any other point relevant to the user to understand the model. The full project should be saved as an Rstudio project. An Rstudio project is a standalone working environment. This removes the trouble of having to specify where files are located when the model is used on different computers. All base case input parameters (with their distribution and ranges) should be defined in data files (.csv recommended, including also information about the sources of the input values), which should be loaded into the model (see '03 Loading of input parameters' in section 2.6). It must be possible to check input data files without the use of R (e.g. it is not allowed to include input data as an .Rdata file).

Input data files should facilitate that alternative base case values and ranges can be tested. For this purpose, data input files should specify base case values with references, the ranges that are used, and the type of distributions that are applied, but generally they should not specify the distribution parameters themselves (e.g. alpha and beta for the beta distribution). The distribution parameters themselves should be generated within the script, so that they can be recalculated based on alternative base case values and ranges specified by the user.

It is mandatory to use a folder structure that separates at least data, code and results. The recommended folder structure is:

I. Readme
II. Data: in this folder all data should be stored. This may contain (in subfolders) raw data, processed data and a file with externally derived parameter values, and could be stored as '.csv' files, R data files (with .RData, .rds, or .rda extensions) or other data file extensions. Input data should be stored with a file extension that can be opened without R (see above), there is no such requirement for processed data. The data files should be numbered, according to the numbers of the parts of the code in which they are used (see section 2.6).
III. Scripts: in this folder the main R script(s) should be stored. Functions and unit tests should be in separate scripts. Scripts should be numbered in their order of execution.
IV. Functions: in this folder '.R' files containing user-defined functions should be stored. Functions may be wrapped in a package.
V. Output: in this folder intermediate or final output data should be stored in subfolders. Sometimes output data generated by the model needs to be used in other parts of the model, and these intermediate outcomes may be saved.
VI. Tables and figures: in this folder tables and figures should be stored that are generated to be used in the final dossier.
VII. Tests: this folder should contain at least one R script in which unit tests are coded. In section 2.9, a descripting of the recommended unit tests can be found.

2.6 Code structure and R Markdown

The R-code should be able to function stand-alone. All model code (except additional functions not part of pre-existing packages and the unit tests, see section 2.9) should be in a single file. R Markdown should be used to structure the code in coding blocks and explanatory text. Despite the option to generate reports using R markdown, the official report for ZIN should still be written in the word-template from ZIN. The following paragraphs should be defined in the R Markdown script:

01 Installation
   Containing all information to make the model operational. This section should also state the version of R and the packages used at the time of submission.
02 Loading functions
03 Loading of input parameters
   This paragraph should be divided in the following sections (sections may be added if necessary):
   - 03.a Patient characteristics
   - 03.b Effectiveness parameters (e.g. transition probabilities or survival curve extrapolation)
   - 03.c Probabilities of adverse events
   - 03.d Utilities
   - 03.e Costs
04 Model settings
   This part should contain the settings for the model type, the number of health states, the time horizon, the cycle length, the perspective, etc.
05 Base case analysis and results
06 Deterministic sensitivity analyses and results
07 Probabilistic sensitivity analysis and results
08 Scenario analyses and results
09 VOI (optional)
10 Validation

2.7 Coding conventions and commenting

Commenting

Commenting should be extensive, comprehensive and clear. Each function, and/or block of code should be clearly commented to describe the purpose of the code and to make it easy to check whether the code functions as intended. Examples of clear R code and adequate commenting are for example provided on the websites of the R for HTA and DARTH consortia (https://r-hta.org/publication/; http://darthworkgroup.com/r-code/).

Coding conventions

The tidyverse style guide (available from https://style.tidyverse.org) provides general recommendations for good coding practices that should be adhered to. The recommendations in the tidyverse style guide can be complemented with specific styles dedicated to CE-modelling. The DARTH coding framework presents file and variable naming conventions that are strongly recommended.[3] Using other style guides is acceptable but they need to be fully explained in the README/metadata file (see section 2.5) alongside the model submission and variable names should include at least the specification of the data type, variable type and a clear name.

Following DARTH, variable names listed in Table 1 should be used.[3] Variable and function names should contain underscores to separate words and should only contain lower case letters. Use the following naming structure: <x>_<y>_<var_name>, where <x> indicates the data type, <y> indicates the variable type, and <var_name> is a short but clear name indicating the variable.[3] An example is: v_r_mort_by_age (vector, rate, mortality by age).

Table 1: Recommended prefixes. This list is indicative and not necessarily comprehensive.

Data type	Prefix	Variable type	Prefix
Vector	v	Number	n
Matrix	m	Probability	p
Array	a	Rate	r
Data frame	df	Utility	u
Data table	dtb	Cost	c
List	l	Hazard ratio	hr
Scalar	(no prefix)	Relative risk	rr
		Life years	ly
		QALYs	q
		Standard error	se

Source: Alarid-Escudero et al.[3]

2.8 Packages and functions

As an extension of the standard functionality of R (base R), various packages are available. These packages include a collection of functions, data and documentation. Before these packages can be used, they need to be installed. The list with acceptable packages by ZIN to conduct the various types of analyses to estimate the cost-effectiveness of a new pharmaceutical can be found in Appendix A. Within this list, the available packages are categorized according to their role in the estimation of cost-effectiveness, i.e. supporting packages, preparatory packages, cost-effectiveness analysis (CEA), deterministic sensitivity analysis (DSA) and probabilistic sensitivity analysis (PSA) packages, value of information (VOI) analysis packages and model calibration packages. Usually, these packages include all functions for the required parts of a CEA. The use of existing packages is strongly encouraged. When existing packages are not used, it must be fully justified why the available functions from these packages are not adequate. In the case existing packages are truly inefficient, additional functions may be written using base R for actions that are repetitive.

ZIN only allows the use of packages that are available from the Comprehensive R Archive Network (CRAN: https://cran.r-project.org/). ZIN strives to provide a comprehensive list of packages. Therefore, packages outside of the list below are in principle not allowed. However, packages developed in the future may be added, and particularly in the early versions of this guideline it is expected that important packages are missing. Therefore, when there is an intention to use packages that are not listed below, ZIN should be contacted.

2.9 Testing

Unit tests should be included to make sure parts of the model function as expected, and their interdependence is valid. There should be a set of tests for each additional complicated function that is written. Additionally, tests may be included that test larger components of the model. We recommend to use the package *testthat* to perform unit testing (https://cran.r-project.org/web/packages/testthat/). Use of the *testthat* package is not mandatory, model components may be checked using other functions. Worked examples of unit tests within the context of cost-effectiveness models have been provided by DARTH.[3]

2.10 Additional features

R shiny interfaces are not required for submitted CE-models, but they are allowed. However, all underlying code should be submitted. Any additional user functionalities will not be checked by assessors and will not be included in final HTA dossiers either.

2.11 Submission procedure

The general submission procedure applies. In addition, all files and data required to run the model should be submitted in a single folder (with subfolders according to folder structure section 2.5). All descriptions on the folders and code do not need to be part of the submitted report but can be included in the readme file included in the dedicated R folder. The written

dossier will thus closely resemble dossiers submitted alongside models built in Excel, excluding any technical specifications.

2.12 Issues not addressed by this guidance

In case issues are not addressed by this guidance, in case of ambiguous or confusing statements, or if there exists a wish to implement additional features not described here, the general principles described in section 2.2 of this guideline serve as the basic criteria that all CE-models in R should fully adhere to. Caution should be taken with the introduction of features outside of the scope of this guidance, and any major modifications (e.g. the use of additional packages) should be discussed with ZIN in advance to prevent delays in HTA procedures.

3 References

1. Baio G and Heath A. When simple becomes complicated: why Excel should lose its place at the top table. Glob Reg Health Technol Assess 2016; 4: e3-e6.
2. Incerti D, Thom H, Baio G, et al. R You Still Using Excel? The Advantages of Modern Software Tools for Health Technology Assessment. Value Health 2019; 22: 575-9.
3. Alarid-Escudero F, Krijkamp EM, Pechlivanoglou P, et al. A Need for Change! A Coding Framework for Improving Transparency in Decision Modeling. Pharmacoeconomics 2019; 37: 1329-39.