The Probabilistic Analysis Check R Package

X. Pouwels & H. Koffijberg

2024-10-04

Target journal: Pharmacoeconomics  
Type of paper: Practical application

# Introduction

Health economic (HE) models are routinely developed to inform health policy decisions such as including (new) healthcare interventions in insurance packages or restricting their use to specific subgroups**[REF]**. Validating health economic models is an important step within the development of health economic models to increase their credibility, and reduce the risk of suboptimal policy decisions **[REF CARO]**. However, validation efforts on HE models are not always systematically performed and/or reported.  
Validation of HE models is a multifaceted concept. The ISPOR-SMDM Modeling Good Research Practices Task Force defined different aspects of HE model validity: face validity, verification (also called internal validity), cross validity, and external validity of the model structure, inputs, and outputs [<https://doi.org/10.1016/j.jval.2012.04.012>], but does not provide an easy-to-use tool to report validation efforts for each of these aspects. Following this initial definition of HE model validation, multiple generic tools have been developed to structure the (reporting of) the validation efforts performed on HE models. These tools are, among other, the AdviSHE tool, the TECH-VER checklist, and the CADTH’s Model Validation Tool to assist in the Conduct of Economic Evaluations**[REFs]**. Both AdviSHE and CADTH’s tool assess multiple aspects of validity, such as face validity and technical verification while TECH-VER solely focuses on the latter.  
Besides the availability of these validation tools, the (increasing) use of script-based software, such as ‘R’**[REF]**, to develop HE models **[REF Jalal]** facilitates the automated execution of systematic and generic validation tests and may be key to improve HE modelling validation practices . For instance, the hesim package**[REF]**, *an R package for health economic simulation modeling and decision analysis*, contains a function to easily check the summary statistics of the parameter distributions used for the probabilistic analysis (PA). Similarly, the darthpack package**[REF]**, *an R package that showcases the Decision Analysis in R for Technologies in Health (DARTH) coding framework*, contains functions to assess the validity of the transition matrices and arrays used to populate health state transition models developed according to this framework. These examples show the feasibility of integrating simple validation checks during the development of script-based health economic models. Even though the nature of these validation tests is generic, their current implementation within coding frameworks for HE models may limit their usefulness beyond HE models developed without these coding frameworks.  
Hence, generic (R) software tools to easily and systematically validate HE models are required to improve HE model validation practices. Such tools may be particularly useful for developers who are new to script-based HE model development. Examples of such tools are the assertHE R package and the probabilistic check analysis dashboard (PACBOARD). assertHE depicts the network of functions defined and used within a HE model developed in R. This package is useful for HE model developers and reviewers since one gets an overview of the backbone of the HE model. This package may further improve the communication of the workings of a developed HE models to a less technically-oriented audience. PACBOARD is a web-based dashboard which allows to systematically validate HE model input parameter and output values and allows to explore the relation between HE model input and outputs values. This dashboard is partly powered by the pacheck R package, which offers a suite of functions aiming at validating HE models and exploring their workings using metamodelling techniques. Metamodels are statistical models, such as a linear regression model, fitted to the (probabilistic) inputs and outputs of a HE models. These metamodels allow to rapidly estimate the output of a HE models without requiring access to the source code, but also allow to assess the direction and magnitude of the relationships between inputs and outputs.  
The current paper describes the functionalities of the pacheck R package using the probabilistic inputs and outputs of two published open-source HE models**[REF]**. The aim is to provide a tutorial for the use of the pacheck R package, including example codes, for novice R users.

# Illustrations of functionalities

## Case studies

The functionalities of pacheck are illustrated using the cdx2cea R package. This package has been developed to perform a *“cost-effectiveness analysis (CEA) of testing average-risk Stage II colon cancer patients for the absence of CDX2 biomarker expression followed by adjuvant chemotherap which contains”* **[REF]**. The CEA incorporated in this package is performed using a cohort-based health state transition model. It compares two strategies: No CDX2 testing and CDX2 testing. The PA datasets contains 1000 iterations of the HE models. The HE model contains 18 input parameters and provides discounted quality-adjusted life years and discounted costs for both strategies which are stored in the l\_psa.RData object of the cdx2cea R package. This case study demonstrates that the functionalities of pacheck can be used on a HE model developed according to the darth coding framework. The other case study is a mock partitioned survival model that has been developed to demonstrate the functionalities of pacheck and PACBOARD. A short description of this three-state oncology partitioned survival model is provided in Pouwels et al.[**REF PACBOARD**] and by following this link: **ADD**. The prtobabilistic inputs and outputs of this HE model are available within pacheck as example dataset.

## Validating health economic model inputs and outputs

Functionalities of pacheck to assess the validity of the model input parameters concern the plausibility of the range in which model input parameters vary. For instance, pacheck contains simple validity assessments such as assessing whether utility values and probabilities remain between 0 and 1, and whether there are no occurrence of negative costs or other strictly positive parameters in each probabilistic iteration. pacheck also contains a function to assess and visualise whether two survival curves cross (check\_surv\_mod and plot\_surv\_mod). This assessment of plausibility is especially relevant when developing partitioned survival models in which a progression-free survival (PFS) curve should always result in lower probabilities than an overall survival curve (OS). Concerning the validation of model outputs, users are able to check whether total costs and effects are positive, whether discounted results are lower than undiscounted results, and whether the mean quality of life of. All validation efforts are also performed for each iteration of the probabilistic analysis and the pacheck functions mention in which iteration an erroneous input or output has been identified. pacheck can be used to visually assess the convergence of HE model outputs**[REF]**. Finally, pacheck contains a function to assess the plausibility of probabilistic model inputs and outputs estimated and structured using the DARTH coding framework and naming conventions (check\_psa\_darth). Box 1 illustrates how to perform these validation efforts using different functions of pacheck.

*Box 1: example R code of using pacheck to validate the health economic model inputs and outputs*

# Install and load packages  
# install.packages("devtools")  
# devtools::install\_github("feralaes/cdx2cea")  
# devtools::install\_github("DARTH-git/dampack")  
# devtools::install\_github("Xa4P/pacheck")  
# install.packages("foreach")  
require(cdx2cea)  
require(dampack)

## Loading required package: dampack

## Loading required package: ggplot2

require(pacheck)

## Loading required package: pacheck

data("l\_psa", package = "cdx2cea")  
data("df\_pa\_psm", package = "pacheck")  
  
# Inspect parameter values - limited to first 5 for the sake of brievety   
pacheck::generate\_sum\_stats(l\_psa$parameters[ ,1:5])

## Parameter Mean SD Percentile\_2.5th Percentile\_97.5th Minimum  
## 1 r\_DieMets 0.047 0.005 0.038 0.058 0.031  
## 2 r\_RecurCDX2pos 0.004 0.001 0.003 0.006 0.002  
## 3 hr\_RecurCDX2neg 3.090 0.595 2.075 4.328 1.703  
## 4 p\_Mets 0.961 0.025 0.899 0.990 0.812  
## 5 p\_CDX2neg 0.072 0.011 0.052 0.095 0.041  
## Maximum Median Skewness Kurtosis  
## 1 0.065 0.046 0.400 3.316  
## 2 0.007 0.004 0.159 3.104  
## 3 6.247 3.047 0.616 3.992  
## 4 0.995 0.968 -2.343 11.511  
## 5 0.106 0.071 0.300 3.176

# Check whether utility values are within the 0-1   
pacheck::check\_binary(c("u\_Stg2", "u\_Stg2Chemo", "u\_Mets"),   
 df = l\_psa$parameters)

## Input Negative\_values Values\_above\_1  
## 1 u\_Stg2 None None  
## 2 u\_Stg2Chemo None None  
## 3 u\_Mets None None

# Introduce utility values below 0 and above 1 to illustrate how pacheck identifies it  
df\_inputs\_error <- l\_psa$parameters  
df\_inputs\_error[c(4, 444, 754), "u\_Stg2"] <- -1  
df\_inputs\_error[c(3, 333, 681), "u\_Stg2Chemo"] <- 99  
df\_inputs\_error[c(5, 554, 153), "u\_Mets"] <- -1  
df\_inputs\_error[c(6, 146, 538), "u\_Mets"] <- 99  
  
pacheck::check\_binary(c("u\_Stg2", "u\_Stg2Chemo", "u\_Mets"),   
 df = df\_inputs\_error)

## Input Negative\_values Values\_above\_1  
## 1 u\_Stg2 4, 444, 754 None  
## 2 u\_Stg2Chemo None 3, 333, 681  
## 3 u\_Mets 5, 153, 554 6, 146, 538

## pacheck identified all erroneous utility values  
  
# Check whether costs and hazard ratios are strictly positive   
v\_names\_positive <- c(l\_psa$parnames[grep("^c\_", l\_psa$parnames)],  
 l\_psa$parnames[grep("^hr\_", l\_psa$parnames)])  
pacheck::check\_positive(v\_names\_positive, df = l\_psa$parameters) # No negative values within these parameters

## Input Negative\_values  
## 1 c\_Chemo None  
## 2 c\_ChemoAdmin None  
## 3 c\_CRCStg2\_init None  
## 4 c\_CRCStg2\_cont None  
## 5 c\_CRCStg4\_cont None  
## 6 c\_Test None  
## 7 hr\_RecurCDX2neg None  
## 8 hr\_Recurr\_CDXneg\_Rx None  
## 9 hr\_Recurr\_CDXpos\_Rx None

# Check whether survival curves for PFS and OS do cross each other  
pacheck::check\_surv\_mod(df = df\_pa\_psm,  
 surv\_mod\_1 = "exp",  
 surv\_mod\_2 = "weibull",  
 label\_surv\_1 = "PFS",  
 v\_names\_param\_mod\_1 = c("r\_exp\_pfs\_comp"),  
 v\_names\_param\_mod\_2 = c("shape\_weib\_os",  
 "scale\_weib\_os\_comp"),  
 label\_surv\_2 = "OS")

## $message  
## [1] "The PFS curve is lower than the OS curve in all iterations."  
##   
## $v\_n\_cross  
## integer(0)

## Introduce low rate for PFS  
df\_pa\_psm\_error <- df\_pa\_psm  
df\_pa\_psm\_error$r\_exp\_pfs\_comp[c(1, 4, 6)] <- 0.01  
pacheck::check\_surv\_mod(df = df\_pa\_psm\_error,  
 surv\_mod\_1 = "exp",  
 surv\_mod\_2 = "weibull",  
 label\_surv\_1 = "PFS",  
 v\_names\_param\_mod\_1 = c("r\_exp\_pfs\_comp"),  
 v\_names\_param\_mod\_2 = c("shape\_weib\_os",  
 "scale\_weib\_os\_comp"),  
 label\_surv\_2 = "OS")$message # errors are identified

## [1] "Pay attention, the PFS curve is higher than the OS curve in iterations 1, 4, 6"

# Check crossing in iteration 4  
pacheck::plot\_surv\_mod(df = df\_pa\_psm\_error,  
 surv\_mod\_1 = "exp",  
 surv\_mod\_2 = "weibull",  
 label\_surv\_1 = "PFS",  
 v\_names\_param\_mod\_1 = c("r\_exp\_pfs\_comp"),  
 v\_names\_param\_mod\_2 = c("shape\_weib\_os",  
 "scale\_weib\_os\_comp"),  
 label\_surv\_2 = "OS",  
 iteration = 4,  
 time = seq(0, 5, 0.1))



# Perform multiple plausibility checks on a psa object constructed using the darth coding framework - also on 'effectiveness' and 'cost' outcomes  
data.frame(pacheck::check\_psa\_darth(l\_psa))

## Parameter Iterations\_error  
## 1 u\_Stg2 none  
## 2 u\_Stg2Chemo none  
## 3 u\_Mets none  
## 4 p\_Mets none  
## 5 p\_CDX2neg none  
## 6 c\_Chemo none  
## 7 c\_ChemoAdmin none  
## 8 c\_CRCStg2\_init none  
## 9 c\_CRCStg2\_cont none  
## 10 c\_CRCStg4\_cont none  
## 11 c\_Test none  
## 12 hr\_RecurCDX2neg none  
## 13 hr\_Recurr\_CDXneg\_Rx none  
## 14 hr\_Recurr\_CDXpos\_Rx none  
## 15 effectiveness\_No CDX2 testing and no FOLFOX none  
## 16 effectiveness\_CDX2 testing and FOLFOX if CDX2-negative none  
## 17 cost\_No CDX2 testing and no FOLFOX none  
## 18 cost\_CDX2 testing and FOLFOX if CDX2-negative none

# Check whether the mean quality of life of simulated individuals remain within the lowest en highest utility values used in each iteration  
## In the comparator group (+ introducing errors)  
df\_pa\_psm\_error$t\_qaly\_comp[1:10] <- 2.01 # introducing total QALYs that will lead to mean quality of life below lowest utility value used in the iteration  
pacheck::check\_mean\_qol(df = df\_pa\_psm\_error,  
 t\_ly = "t\_ly\_comp", # undiscounted total life years of the comparator  
 t\_qaly = "t\_qaly\_comp", # undiscounted total quality-adjusted life years of the comparator  
 u\_values = c("u\_pfs", "u\_pd")  
 )

## Mean\_QoL\_below\_min   
## Number of iteration with issue "10"   
## Iteration number with issue "1, 2, 3, 4, 5, 6, 7, 8, 9, 10"  
## Mean\_QoL\_above\_max  
## Number of iteration with issue "0"   
## Iteration number with issue "0"

# Check convergence of a model output  
## Calculate incremental effectiveness  
l\_psa$effectiveness$Incremental\_effectiveness <- l\_psa$effectiveness$`CDX2 testing and FOLFOX if CDX2-negative` - l\_psa$effectiveness$`No CDX2 testing and no FOLFOX`  
## Plot moving average incremental effectiveness - per 100 iterations  
pacheck::plot\_convergence(l\_psa$effectiveness, param = "Incremental\_effectiveness", block\_size = 100) # Incremental difference varies within 0.002 QALYs over blocks of 100 iterations



## Investigating the relationships between inputs and outputs

The pacheck package also contains diverse functions to investigate the relation between HE model inputs and outputs. For instance, the correlation matrix or plot between inputs and outputs can be calculated using the generate\_cor function. Using the probabilistic inputs and outputs of the cdx2cea, one can see that cancer mortality rate, r\_DieMets, is negatively correlated with the total costs of the intervention. This result seems logical since a higher probability of death would lead to shorter survival and thus lower costs. Linear regression metamodelling is also available through pacheck (using the lm function for the linear regression modelling). When estimating the incremental difference in QALYs between the intervention and the comparator, we can see that an increase in utility values of stage II cancer with and without chemotherapy, u\_Stg2 and u\_Stg2Chemo, leads to higher incremental QALYs while an increase in the utility value of the metastatic recurrent state leads to lower incremental QALYs. Most of the paramaters included in the linear regression metamodel are not statistically significant (lm\_metamod). This metamodel can be validated and, if deemed valid, used to perform sensitivity analysis. This is especially useful for computationally intensive HE models. As mentioned by Jalal et al. metamodel’s parameters may be subject to scale effects and normalisation of the HE model inputs may be useful before applying metamodelling, to facilitate their interpretation in relation to each other**[REF]**. Normalisation of inputs has been implemented in pacheck within the lm\_metamod. One can validate the fitted metamodel using the user-defined train-test split proportions or cross-validation using a user-defined number of folds**[REF]**. In Box 2, we show how to validate the linear regression metamodel using the test-train split method. The high R2 value (0.97) and the position of the prediction versus observation dots in the calibration dots near the 45 degree line show that the metamodel may be deemed sufficiently valid to estimate the incremental QALYs. Finally, this metamodel can be used to predict these incremental QALYs using an alternative set of parameters in sensitivity analyses. In Box 2, we show that increasing the utility value for the metastatic state, u\_Mets, to 0.5 (mean value in the probabilistic set was 0.25) increases the incremental QALYs by 0.01, from 0.035 in the basecase to 0.036. Box 2 illustrates how to use the functions of pacheck to investigate the relationships between HE model inputs and outputs.

*Box 2: Example R code of how to use the pacheck package to assess the relationships between inputs and outputs.*

# Generate correlation matrix between inputs and outputs  
## Transform l\_psa parameters and outcomes in one dataframe  
df\_psa\_cdx2cea <- cbind(l\_psa$parameters,   
 l\_psa$effectiveness,   
 l\_psa$cost)  
tbl\_cor <- pacheck::generate\_cor(df\_psa\_cdx2cea[, c(1:5, ncol(df\_psa\_cdx2cea))]) # only 5 first parameters and costs of the intervention  
tbl\_cor[, "CDX2 testing and FOLFOX if CDX2-negative"] # only correlation with outcome

## r\_DieMets   
## -0.508554559   
## r\_RecurCDX2pos   
## 0.750189815   
## hr\_RecurCDX2neg   
## -0.312984661   
## p\_Mets   
## -0.001900022   
## p\_CDX2neg   
## 0.049150333   
## CDX2 testing and FOLFOX if CDX2-negative   
## 1.000000000

p\_cor <- pacheck::generate\_cor(df\_psa\_cdx2cea,  
 figure = T) # correlation matrix using 'tile' plot from ggplot2  
  
# Fit linear metamodel to predict the difference in QALYs between strategies using all parameters, except costs  
v\_x\_vars <- l\_psa$parnames[grep("^c\_", l\_psa$parnames, invert = T)]  
y\_var <- names(l\_psa$effectiveness)[3]  
lm\_metamod <- pacheck::fit\_lm\_metamodel(y\_var = y\_var,   
 x\_vars = v\_x\_vars, # altenatively, the names of the parameters can be used  
 df = df\_psa\_cdx2cea,   
 seed\_num = 123 # seed number for reproducibility  
 )  
summary(lm\_metamod$fit) # provides an overview of estimated linear metamodel parameters

##   
## Call:  
## lm(formula = form, data = df)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.0097410 -0.0017664 -0.0007761 0.0009259 0.0250413   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)  
## (Intercept) 0.1684922598 0.0828378588 2.034 0.0422  
## r\_DieMets 0.0088003017 0.0229356484 0.384 0.7013  
## r\_RecurCDX2pos -1.9742925140 0.1951303413 -10.118 < 0.0000000000000002  
## hr\_RecurCDX2neg -0.0031447830 0.0002095364 -15.008 < 0.0000000000000002  
## p\_Mets -0.0030930124 0.0044799734 -0.690 0.4901  
## p\_CDX2neg 0.4919905564 0.0103636969 47.472 < 0.0000000000000002  
## hr\_Recurr\_CDXneg\_Rx -0.2310806122 0.0012971089 -178.151 < 0.0000000000000002  
## hr\_Recurr\_CDXpos\_Rx -0.0001457265 0.0085688076 -0.017 0.9864  
## ic\_DeathCRCStg2 -0.0000001826 0.0000014571 -0.125 0.9003  
## ic\_DeathOCStg2 0.0000016267 0.0000012873 1.264 0.2067  
## u\_Stg2 0.0204151331 0.0046984042 4.345 0.0000153600652  
## u\_Stg2Chemo 0.0284825885 0.0041731900 6.825 0.0000000000153  
## u\_Mets -0.0039898544 0.0039401775 -1.013 0.3115  
##   
## (Intercept) \*   
## r\_DieMets   
## r\_RecurCDX2pos \*\*\*  
## hr\_RecurCDX2neg \*\*\*  
## p\_Mets   
## p\_CDX2neg \*\*\*  
## hr\_Recurr\_CDXneg\_Rx \*\*\*  
## hr\_Recurr\_CDXpos\_Rx   
## ic\_DeathCRCStg2   
## ic\_DeathOCStg2   
## u\_Stg2 \*\*\*  
## u\_Stg2Chemo \*\*\*  
## u\_Mets   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.003448 on 987 degrees of freedom  
## Multiple R-squared: 0.9724, Adjusted R-squared: 0.972   
## F-statistic: 2895 on 12 and 987 DF, p-value: < 0.00000000000000022

# Normalise the inputs  
lm\_metamod\_standardised <- pacheck::fit\_lm\_metamodel(y\_var = y\_var,   
 x\_vars = v\_x\_vars,  
 df = df\_psa\_cdx2cea,  
 seed\_num = 123,  
 standardise = TRUE)  
summary(lm\_metamod\_standardised$fit) # provides an overview of estimated linear metamodel parameters

##   
## Call:  
## lm(formula = form, data = df)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.0097410 -0.0017664 -0.0007761 0.0009259 0.0250413   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 0.03580038 0.00010903 328.356 < 0.0000000000000002 \*\*\*  
## r\_DieMets 0.00004445 0.00011586 0.384 0.701   
## r\_RecurCDX2pos -0.00135079 0.00013351 -10.118 < 0.0000000000000002 \*\*\*  
## hr\_RecurCDX2neg -0.00187099 0.00012466 -15.008 < 0.0000000000000002 \*\*\*  
## p\_Mets -0.00007734 0.00011202 -0.690 0.490   
## p\_CDX2neg 0.00519457 0.00010942 47.472 < 0.0000000000000002 \*\*\*  
## hr\_Recurr\_CDXneg\_Rx -0.01956199 0.00010981 -178.151 < 0.0000000000000002 \*\*\*  
## hr\_Recurr\_CDXpos\_Rx -0.00000187 0.00010995 -0.017 0.986   
## ic\_DeathCRCStg2 -0.00001377 0.00010987 -0.125 0.900   
## ic\_DeathOCStg2 0.00013806 0.00010926 1.264 0.207   
## u\_Stg2 0.00047474 0.00010926 4.345 0.0000153600652 \*\*\*  
## u\_Stg2Chemo 0.00074829 0.00010964 6.825 0.0000000000153 \*\*\*  
## u\_Mets -0.00011111 0.00010973 -1.013 0.311   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.003448 on 987 degrees of freedom  
## Multiple R-squared: 0.9724, Adjusted R-squared: 0.972   
## F-statistic: 2895 on 12 and 987 DF, p-value: < 0.00000000000000022

# Validation metamodel using the train-test approach  
lm\_metamod\_valid <- pacheck::fit\_lm\_metamodel(y\_var = y\_var,   
 x\_vars = v\_x\_vars,  
 df = df\_psa\_cdx2cea,  
 seed\_num = 123,  
 validation = "train\_test\_split",  
 partition = 0.75) # in combination with the "train\_test\_split" approach, the proportion of observation used to fit the metamodel (`partition`) should be provided. The remainder is used as validation check  
lm\_metamod\_valid$stats\_validation

## Statistic Value (method: train/test split)  
## 1 R-squared 0.971  
## 2 Mean absolute error 0.002  
## 3 Mean relative error 0.253  
## 4 Mean squared error 0.000

lm\_metamod\_valid$calibration\_plot



# Prediction metamodel  
df\_params <- data.frame(t(colMeans(df\_psa\_cdx2cea[, v\_x\_vars]))) # calculate mean values of each parameter  
df\_params$u\_Mets <- 0.5 # increase the utility value of the metastatic state  
v\_pred <- predict\_metamodel(model = lm\_metamod\_valid,  
 inputs = df\_params) # inputs has to be a dataframe  
v\_pred # 0.036 QALYs, basecase incremental QALYs was 0.035, hence

## [1] 0.03600197

# Discussion

The current paper presents functionalities of the pacheck R package that focuses on validating probabilistic HE model inputs and outputs and allows to investigate the relation between these model inputs and outputs using metamodelling. Additionally, sample R codes are provided to illustrate how to use pacheck on available probabilistic inputs and outputs from published HE models.  
The validation tests included in pacheck are based on a pragmatic literature review described in Pouwels et al.**[REF]**. Most of the validation tests included in pacheck were identified in the TECH-VER checklist, emphasising their relevance for HE models. In comparison with other R package dedicated to HE model development and validation**[REF]**, pacheck may be applied to inputs and outputs of HE models developed with other software package than R once the probabilistic inputs and outputs are loaded within the R session of interest. This increases the accessibility of these validation tests to HE models developed in other software packages than R and potentially to non-R-users.  
By no means we aimed to provide a complete list of validation tests. To our opinion, pacheck is a package that requires regular updates and addition of validation tests over time to ensure it remains relevant for HE model developers and reviewers.  
From a technical point of view, pacheck is most likely not coded in the most efficient way. This is a choice we made to ensure transparency of the implemented validation tests and to encourage external contributors to review the code base and to contribute to the further development of the package. The source code of pacheck is openly available on GitHub: <https://github.com/Xa4P/pacheck>. External contributors can raise “Issues” concerning the package and propose new validation tests via “Issues” and “Pull requests”.  
Finally, pacheck focuses on the technical verification of HE model which is only a single aspect of validity. Hence, passing the validation tests included in pacheck supports - but does not guarantee - the validity of HE models. HE model developers and reviewers are therefore encouraged to also use additional validation tools such as AdviSHE and the CADTH’s tool to report and review other aspects of model validation**[REFs]**.

# Conclusion

The current paper presents the R package pacheck, which aims at validating probabilistic HE model inputs and outputs and at investigating their relationships using metamodelling. We provide sample R code to encourage novice R users to use this package to systematically validate their HE models. Via this paper, we want to encourage external contributors to engage in the development and update of pacheck.