# National Institute for Health and Care Excellence

**ANTIMICROBIAL HEALTH TECHNOLOGY EVALUATION**

**Pro-forma Response**

**Executable Model**

**Ceftazidime with avibactam for treating severe aerobic Gram-negative bacterial infections**

**Cefiderocol for treating** **severe aerobic** **Gram-negative** **bacterial infections**

The economic model enclosed and its contents are confidential and are protected by intellectual property rights, which are owned by **the Policy Research Unit in Economic Methods of Evaluation in Health & Social Care Interventions (EEPRU)**. It has been sent to you for information only. It cannot be used for any other purpose than to inform your understanding of the evaluations. Accordingly, neither the model nor its contents should be divulged to anyone other than those individuals within your organisation who need to see to them to enable you to prepare your response. Those to whom you do show the documents must be advised they are bound by the terms of the Confidentiality Agreement Form that has already been signed and returned to NICE by your organisation.

You may not make copies of the files and you must delete the files from your records when the evaluation process is complete. If asked, you must confirm to us in writing that you have done so. You may not publish it in whole or part, or use it to inform the development of other economic models.

**The model must not be re-run for purposes other than the testing of its reliability**.

Please set out your comments on reliability in writing providing separate justification, with supporting information, for each specific comment made. Where you have made an alteration to the model, details of how this alteration was implemented in the model (e.g. in terms of programme code) must be given in sufficient detail to enable your changes to be replicated from the information provided. Please use the attached pro-forma to present your response.

Please prepare your response carefully. Responses which contain errors or are internally inconsistent (for example where we are unable to replicate the results claimed by implementing the changes said to have been made to the model) will be rejected without further consideration.

Results from amended versions of the model will only be accepted if their purpose is to test robustness and reliability of the economic model. Results calculated purely for the purpose of using alternative inputs will not be accepted.

No electronic versions of the economic model will be accepted with your response.

Responses should be provided in tabular format as suggested below (please add further rows if necessary).

**November 2021**

1. General points on implementation and transparency

|  |  |  |
| --- | --- | --- |
| **Description of problem** | **Description of proposed amendment** | **Result of amended model or expected impact on the result (if applicable)** |
| The R code is not implemented efficiently or transparently, and that this has prevented EEPRU from being able to run probabilistic scenario analyses. It has also prevented us from being confident in our review of the model code as it is difficult to follow. We have identified errors which are detailed in subsequent issues, yet we are not confident that all errors were identified during the QC process. | We would be happy to offer assistance or advice in reorganising the model in an efficient and transparent manner, which would then facilitate both modifications to the code functionality and higher quality code review / QC. We hope that the following suggestions are useful to EEPRU:   1. Using an excel file to house all of the input data instead of several csvs scattered across different folders (which led to errors in the code with naming of those csvs). The *openxlsx* package can be used to extract data and write results from and to excel files, even by named range. Such named ranges could then be kept consistent between Excel and R to allow easy cross-referencing during QC. This would allow explanation of data sources and values within the Excel file (where it is easier to do so in text boxes, comments, etc), improving transparency greatly, and the conduciveness between the EEPRU report and model. We have a single function which can extract named ranges from an excel file into an R list, which we can make available to EEPRU if desired. 2. Establishing all inputs outside of the core model functions, including those that are scenario-specific. This avoids hard-coding values deep within the code within multi-nested loops and function definitions within those loops. Many of the scenarios are currently obfuscated within two or more layers of for loop with single-letter index, within custom functions without annotation or explanation, and are also hard-coded value entries. It is better practice to establish all of the inputs outside of the function call to run the model, and feed those inputs into the function, rather than to have typed-in values hidden from immediate view. 3. Using a centralized data normalization paradigm. This is simple to achieve in the context of a script-based model. Create one or a few list object(s) that house(s) all of the data and analysis results required to run the entire model. This object can then be the argument to a function which runs the model, ensuring consistency of environment within that function (which avoids a lot of errors). These objects (named perhaps inputs, analysis, INHE, population, PNHE) can be backed up, saved to rds file, and replicated for all scenarios easily, passing through the alternative values for each scenario. This would keep all information neat and traceable at all times located within just a handful of objects, allowing easier debugging and tracking of which arguments are entered throughout the calculation chain. This would also remove the need for loading from CSV files within loops (considerably improving computational efficiency), and would considerably reduce the number of required named objects. We would recommend doing this for scenarios as well, and passing an altered version of the same object into a genericised RUN\_MODEL() type function to generate results. This is also useful in improving the overall layout of the master script, which would then be a series of well organised, well named, well documented calls to a sequential set of coherent functions. The final advantage of this is avoidance of the use of the global environment within iterative calls. This avoids errors like the one detailed in issue 2 where an object in the global environment is being called within a loop by mistake. 4. Define a function or set of functions for running the model. Currently, this is partially done. There is a function for the Markov and outcomes simulations, but the running of the model is a 400 line for loop which calls all of these functions several times repeatedly, with many lines of individual arithmetic operations between. For instance, a separate function calculating each of the “stages” that the code is separated into would simplify the Run\_models.R code considerably, would also ease review, whilst creating a platform to thoroughly explain each of the stages in detail in its own R file (or in one file with sections and extensive annotation). As the functional method would also ensure that only the relevant data is passed to the function, this also helps avoid errors. The model should be a neat set of well defined, well documented, functions which have been individually QCd. At the moment, the code appears to be directly translated from VBA, as the current implementation is more conducive to that software. 5. Considerable improvement to the annotation within the model generally, as it opaque currently, even for experienced R programmers that have built several R-based cost-effectiveness models in the past. There are large blocks of code with no explanation, reading from files with no labelling with no explanation in the code, and no readme or similar to explain the overall computational design/implementation of the model. The description in the EEPRU report is inadequate to explain the low-level operations within the model code. 6. Avoidance of for() loops for all but simple arithmetic functions in favour of functional programming and apply family functions (apply rarely, lapply mostly, mapply if necessary, Reduce for Markov traces). These considarably improve efficiency and readability in comparison to base for/while etc loops. We would be happy to offer advice to EEPRU in this aspect. 7. Avoidance of “typewriter” data population (which R is not designed for) in favour of vectorised operations (which R is designed for). That is, in R it is more usual to populate objects one object at a time not e.g. one cell within one object at a time. For instance, the population of object “tm” in the Markovian function does not require a loop as all functions used are vectorised already (there are many other examples of this throughout the code). 8. (optional but gold standard for publishing) Unit testing – It is possible in code-based modelling to “prove” that a function is working correctly or generating the correct results. Unit testing using the “testthat” R package can achieve this. However, this is a considerable undertaking and may not be feasible in the NICE timelines. Yet constructing and implementing unit testing in an improved model framework (composed of a series of functions rather than many lines of individual code) would generally improve confidence in the model results, particularly when these become public. A simple middle-ground would be to take several example datasets containing inputs to each stage of the model, then double checking (I.e. computing by hand) and storing the results. If the function takes the data and produces those results then it must be functioning correctly. This, along with breaking the code down into separate functions would then break any QC down into manageable pieces, reducing the overall need to read through thousands of individual lines of code (as the model would then reduce in size and the majority of the QC would be to QC the approximately 20-30 well defined, explained, pre-tested functions required to generate the model results).   We suggest that following at least a few of these simple steps will allow any issues with the model to be exposed at the same time as improving the model’s usability, adaptability, efficiciency and transparency. | Insert Incremental Net Health Effects (INHE) resulting from amended model.  If the model has not been re-run, if appropriate, describe your expectations of how the problem might have an impact on the result |
| A for loop is not required to do dependent calculations in R, and for loops are highly inefficient for all but basic operations, and struggle when nested. They also have environment issues, in that within a for loop assignments are to the global environment, and not to a sub-environment specific to that set of iterations. Instead, we recommend the use of apply family functional programming functions wherever possible, as these do not slow down when nested and avoid errors like the one in issue 2. As there are many layers of nesting in this model’s lowest point of iteration, the performance improvement should be considerable (perhaps as much as 100x quicker, potentially more with some other more convoluted approaches), and may therefore allow probabilistic scenario analyses within a reasonable runtime. | The function Reduce is very useful for dependent iterations like Markov traces. The below line works similarly to a for loop but only produces the final result as a list (rather than performing an action in the global environment on every iteration):  Reduce(x = cycle.v, accumulate = TRUE, init = 1, function(prev, cycle) {prev \* (1-tp\_selected[cycle])})  To expand to a Markov trace with the transition probability matrix used in this model:  Reduce(  x = 1:(cycles - 1),  accumulate = TRUE,  init = seed,  f = function(prev, cycle) {  prev %\*% tm[, , cycle]  }  )  These will create markov traces efficiently, allowing scalability and probabilistic scenario analysis. Note that the result of each iteration can be of any type, shape, or size, meaning that this approach can be used to iterate an entire row of a “patient flow sheet” in an R-based cost-effectiveness model, depending on the previous row of that sheet. As in the above case the result of each iteration is a vector of health state populations, one can use do.call(rbind, OutList) to stick them together rowwise, producing a data.frame. Alternatively matrix() can be used to achieve the same result with improved computational efficiency.  Another solution would be to code the model in VB. Given the way that it is currently implemented is more conducive to VB programming, the runtime should be considerably faster. This is because VBA is between 100 and 1000x faster at running for/while loops than R is. The R code running *flexsurv* only does so based on Kaplan-Meier data that appears to always remain the same, therefore the parameters and variance-covariance could simply be moved to Excel and a function to draw from multivariate normal distribution used in the probabilistic base-case. As far as we can tell, R is not required for this model as aside from the aforementioned survival analysis there is no essential within-model statistical analysis present. If it is in fact the case that only the flexsurv element strictly requires R, then it would be trivial to iterate on only that in R to produce N sets of parameters and variance-covariance matrices, which can then be imported to an Excel model similarly to an NMA CODA sample.  This approach would also allow the model inputs to be housed inside of an excel file (something which can be done with the R engine as well using *openxlsx*). This would help greatly in explaining all of the data to be used for the model. | Using these functions will increase run speed enough to run probabilistic scenarios rather than deterministic |
| The code uses reading and writing of files inside of looped environments. This is inefficient, meaning that running probabilistic scenarios would take an extended period unecessarily (Base case PSA takes several hours) | Follow best practices for writing R code:   * Try to avoid using base R for loops (See above) * Try to avoid reading and writing data in nested iterations as this slows things down (due to needing to read/write from disk), instead populate a list during iterations and then save the list once at the end * Try to perform operations within a functional environment. This is to avoid assigning objects to the global environment within a looped environment, which is conducive to error. | None, but will enable probabilistic scenario analysis and increasing iterations to ensure convergence whilst reducing runtime overall to a more reasonable level. |
| In ./R/Outcomefct\_PSA.R, the saving and loading of files is unnecessary. Instead, pass the data into the function using an argument. Note that arguments to functions can be lists of objects, a factor which can reduce the amount of arguments entering a function considerably. | A useful strategy for keeping data together is to make the return from a function a list of objects. This would be a good use of this strategy.  Further, an argument to simulate\_outcome() could be itself a list, which can keep the number of arguments down to a reasonable number.  Finally, a function (within the default R package ‘base’) list2env() can be used with the environment = environment() in order to avoid having to refer to sub-elements within the list entering simulate\_outcome() (see issue 9). The with() command makes it very difficult to review the code as it cannot be step-edited (see below).  Following these 2 strategies would avoid having to read any data from disk, and also having to save data to disk during simulate\_outcome. Simply create an empty list at the beginning of the function definition, and populate it along the way, then return() the list at the end of the function for full transparency and the ability to pull any of the parts from that operation henceforth. | None, but improves transparency and efficiency, allowing probabilistic scenarios, OWSA and so on. |
| Use of the with() function makes the model very difficult to review as it cannot be step-edited. | Make one of the arguments to simulate\_outcome a list similar to how the argument “pars” already is one. The simple function list2env() can then be used to assign the elements of that list as objects in the immediate environment, like so:  list2env(pars, environment())  This will achieve the same results whilst also meaning that a reviewer can step edit through the function simulate\_outcome without having to individually assign each element of “pars”. This also has the advantage of not assigning the contents of pars to the global environment (as the environment=environment() argument is used). The contents of pars are attached only within the loop in question, meaning they can be used but do not crowd the environment afterwards. | Improves transparency and allows review of the model, follows best coding practices. |
| In the “resemerg” scenarios, the following code is difficult for us to understand:  if (word(scenario, 1) == "resemerg") {  sus\_new\_rdm <-  (sus\_new\_rdm - (1 / 100) \*  (as.numeric(word(scenario, 4)) - 1) \*  as.numeric(word(scenario, 2)) / 19  )  }  Why is the 2nd word of the scenario name divided by 19? This is not explained or cross-referenced to the EEPRU report to clarify. Appears arbitrary at present. | The code must be annotated and explained appropriately for repeatability, replicability and transparency. | Without an understanding of how susceptibility changing over time is implemented in the model, it is impossible for us to assess the impact or even presence of any errors. |
| In function simulate\_outcome a large block of code is commented out labelled:  Step2: Scenario where update susceptibilities to DRUG Categories for those who fail Emptx and need MDS to reflect empiric tx outcomes  There is no explanation as to why this is commented out, or why there doesn’t appear to be a mechanism for patients to fail Emptx to be treated currently. Was this scenario abandoned in model production? | We would like EEPRU to explain this, and to provide documentation explaining the code and assumptions being made.  We would also like EEPRU to explain how successive lines of treatment are being handled in the ES, as it is currently not clear from the report or the model code. | It is not feasible in the review timeline to know the effect of if there are any errors. |
| The line in simulate\_outcome starting with:outcome <- Profile %>%  Is opaque due to its length (an approximately 240 line (depending on formatting/whitespace) block of code which cannot be stepped through due to being doing in one mutate() and has very minimal annotation/explanation along the way).  This component is central to the model results, yet is not explained and is not trivial. | Put each step of these crucial calculations in a separate muate() call so that the code can be audited, or define this step of the calculations in a separate function to be called at this stage in the code.  We request that EEPRU define this as a well documented function that is called within the loop and defined elsewhere, as it is core to running the cost-effectiveness model so has high importance.  As a minimum, we would expect:   * All parameters defined to be individually explained in an annotation with a full sentence each to explain it * Assumptions being made to be spelled out and also included in the EEPRU report with cross-references in the code to the location in the report where the component is explained. * The multiple nested logical statements to be pulled out into separate steps (easier to do this in a series of simple functions than using dplyr/tidyverse syntax), example below (there are many other, more complex examples than this)   Secondline = case\_when(  str\_starts(Sus, "Snca\_") ~ "NCA",  (Sus == "Sca\_SN" | Sus == "Res\_SN") &  has\_new\_drug & Emptx != "New" ~ "New",  str\_starts(Sus, "Sca\_") & AKI1 == "N" ~ "CA",  str\_starts(Sus, "Sca\_") |  str\_starts(Sus, "Res\_") & AKI1 == "Y" ~ "Salv2",  T ~ "Salv"  )  This could be simplified or explained. | Unknown as not known if errors are present, would make the model more transparent and QC-able. |
| In the line  mutate(PrbgrdDeath = if\_else(Salvage == "N",  p\_bgrdD30d\_MDS\_S\_rdm,  p\_bgrdD30d\_MDS\_nonS\_rdm))  Could EEPRU explain what PrbgrdDeath is? | Improve the degree of explanation to at least the standard that would be expected of a company submission in Excel. | This would allow proper review of the model code and therefore identification of errors or problems to discuss with EEPRU going forward |
| mutate(Groupw = Pr\_NCA \* Pr\_CA \* Pr\_Res \* Pr\_New \* Pr\_AKI1)  Groupw (group weighting) is not well explained and it is not clear. Probabilities of 1 are assigned to some categories and this column is calculated as the product of all possibilities. Is this is not then P(NCA & CA & Res & New & AKI1) I.e. the probability of intersection? What about patients that go straight to CKD in hospital? These don’t appear to have been covered in the expand.grid(). Better annotation of explanation of these steps would help to alleviate the concern that this column is erroneous. | Better description of how population weightings are calculated in the EEPRU report. There currently does not appear to be any description of the validity of this calculation or any assumptions it makes. Distributions of groups do not appear to be presented in the report, so we cannot check this against the model or clinical expert advice. Yet, these values are central to the model results. | Unknown |
| Pr\_deltaAKIrisk1 = (Pr\_survandAKI / (Pr\_survandAKI + Pr\_survandnoAKI)) - p\_bgrdAKI\_rdm  Assumptions surrounding this are not stated and there is no explanation | Please explain and list out all of the modelling assumptions. | Unclear |
| In the line l2choice <- select(outcome, c(1:4, 6))  At the bottom of the definition of simulate\_outcome, this is only pulling the result of the dplyr chain for the last iteration as this is a for loop which assigns objects to the global environment. Is this the intended behaviour? | EEPRU should explain their code, and investigate whether this is an error | Unclear |
| In line 35 of 1\_pop\_level\_benefit.R, why is there no variation in surv\_uti\_psa or surv\_iai\_psa? | No explanation given for what seems like an important discussion about variation of survival probabilities. If the survival analysis is not varied in the PSA then there is no justification for the model to be in R, as there is just one set of survival parameters which can be varied using a multivariate normal distribution in Excel/VBA. | Unclear |
| In lines 105 and 106 of 1\_pop\_level\_benefit.R why are the values hard coded and not from a file? We do not understand where these come from or their justification.  Some of these numbers appear in Table 42 of the report but others do not.  105-106  #import PHE data  abs\_pop\_mds<-matrix(c(2,6,0,389,48,36,10,247,17,16,2,243,21,30,7,149),4,4)  abs\_pop\_es<-matrix(c(14,44,13,151,418,66,94,271,31,519,1539,47),3,4)  138-139  #import PHE data  abs\_pop\_mds<-matrix(c(19,14,7,2127,77,49,6,244,17,16,2,243,21,30,7,149),4,4)  abs\_pop\_es<-matrix(c(97,275,203,231,569,61,94,271,31,519,1539,47),3,4) | Improve clarity by stating the source of data being entered manually. We are currently unable to verify or check for factual inacuraccy as we do not know the source of this data and therefore cannot check it. Alternatively, house all of the inputs in an excel file with full descriptions and justifications to accompany them, thus alleviating these concerns. | unknowable |
| In the linesurv\_uti\_det <- surv\_iai\_det <- 0.854  This value appears in Table 21 of the EEPRU report, though this is not cross referenced, so we assume this is the 30 day survival of susceptible cUTI patients? | The code should include cross-references to the EEPRU report. Tables and/or section numbers would be helpful | Unknown, as unable to cross-reference all data inputs, and there are many in the code that are not linked to the report and have no explanation of source. |
| In the line  survey\_pops<-read.csv(file="data\_pop/bsac\_survey.csv", header=TRUE)  The numbers in this table do not appear in the EEPRU report, and are not explained in the code. It is therefore not possible to check for errors or factual inaccuracy | Data inputs determining the results of the population model should be explained and cross-referenced. | Unclear |
| The numbers in  pop\_change\_cpeoxa<-t(as.matrix(read.csv(file="data\_pop/Count\_OXA.csv", header=TRUE)))[,-nyear]  pop\_change\_cpembl<-t(as.matrix(read.csv(file="data\_pop/Count\_NDM.csv", header=TRUE)))[,-nyear]  pop\_change\_psambl<-t(as.matrix(read.csv(file="data\_pop/Count\_Pseud.csv", header=TRUE)))[,-nyear]  Are not explained, and the calculation of these numbers is not provided in the model, it is simply a csv file. | EEPRU should detail where these numbers come from in the code, with cross-references to position in the report.  Also, could EEPRU explain why the file populating the MBL population change are taken from the file Count\_NDM.csv? Is this describing MBL patients? | Potentially large, if the patient numbers are incorrect. |
| Unable to reconcile why this probability time conversion assuming exponential distribution is occuring here:  p\_bgrdD30d\_MDS\_S\_rdm <- ratetoprob(probtorate(bgrdproba\_rdm [j, 10], 30), 25)  p\_bgrdD30d\_MDS\_nonS\_rdm <- ratetoprob(probtorate(bgrdproba\_rdm [j, 11], 30), 25)  This is not explained in the code or the report – is the 30 day probability being transformed into 25 day probability? | The code should be thoroughly annotated explaining every calculation at every stage, at least to the expected standard within an Excel-based model. | Unknown, as the reason for this calculation is unclear and therefore its effect on model results is unknown to us. |
|  |  |  |

1. Errors in the distribution of "bug” probability in the empiric setting

|  |  |  |
| --- | --- | --- |
| **Description of problem** | **Description of proposed amendment** | **Result of amended model or expected impact on the result (if applicable)** |
| In the base-case (Cat 2 MBL) the beta distribution of bug defined at the top of “Step 6” looks unusual:    There appears to be an error in the following line:  p\_bug <- rbeta(  n = nsims,  shape1 = p\_bug \* prob\_bug[prob\_bug$Bug\_cat == bug\_cat & scenario == p\_bug\_scenario, "N"],  shape2 = (1 - p\_bug) \* prob\_bug[prob\_bug$Bug\_cat == bug\_cat & scenario == p\_bug\_scenario, "N"]  )  Whereby scenario == p\_bug\_scenario should in fact be prob\_bug$Scenario == p\_bug\_scenario to select from data.frame prob\_bug, and not the looping index scenario (which is just scenarios[s] with looping index “s”). This may coincidentally avoid error in the code because the 2nd logical statement in the row selections for prob\_bug is by definition TRUE in every scenario due to the lines above the statement in question.  This is clear from the value of mean(p\_bug) not being close to the mean value if the correct alpha and beta were fed into the beta distribution. The mean should be relatively converged onto that point, particularly in the base case with 2000 iterations.  (This is line 272 in original Run\_models.R code) | The error should be amended so that the correct rows in the prob\_bug table are being taken in each scenario. For Cat 2 base case, amending the code to the below:  p\_bug <- rbeta(  n = nsims,  shape1 = p\_bug \* prob\_bug[prob\_bug$Bug\_cat == bug\_cat & prob\_bug$Scenario == p\_bug\_scenario, "N"],  shape2 = (1 - p\_bug) \* prob\_bug[prob\_bug$Bug\_cat == bug\_cat & prob\_bug$Scenario == p\_bug\_scenario, "N"]  )  Results in the following distribution:    Which appears to be a more typical beta distribution.  When writing R code applying multiple sets of logic, it is often easier to read the code using convenience functions like all() and any(), rather than using punctuation alone. Dplyr syntax could also be useful here with the filter() function, as this may be easier to write and read. For example  prob\_bug %>% filter(Bug\_cat == bug\_cat) %>% filter(Scenario == p\_bug\_scenario) %>% dplyr::select(N)  Would pull out the 92 patients representing Cat2 base case. Given that this is then the correct data manipulation, this can be tested with base function calls to optimize performance. As this is quite a low level point in the code which is iterated many times, efficiencies could make a considerable difference to overall runtimes of the model.  Another approach which should help to considerably simplify the model and allow easier review and debugging would be to abstract the “steps” in the code into individual functions, which can then be tested in isolation (and in an environment only containing the arguments to the function, avoiding using data from .globalenv by mistake). This would allow the overall code to be read and interpreted easier, and would likely avoid errors like this one. | Unclear. Code errors should be resolved and then models re-run. |

1. Issue with data treatment in the survival analysis

|  |  |  |
| --- | --- | --- |
| **Description of problem** | **Description of proposed amendment** | **Result of amended model or expected impact on the result (if applicable)** |
| There is an error in data treatment following digitization and pseudo-PLD production. | There is 1 pseudo-patient with time of 0 after re-baselining by subtracting 30 days from the data and filtering out observations with time <0. This is because the digitized time points are rounded, presumably to the closest day (We are not certain whether or not this is the technically correct data treatment). In order for flexsurv() to run the survival models, a small number is added to the entire time column and not only the individual with 0 event time:  surv<-survfit(Surv(t+0.0001,c)~1)  model<-flexsurvreg(Surv(t+0.0001,c)~1, dist =surv\_dist)  As R is vectorised, this adds 0.0001 to all observations of t, not just the one that has time of 0.  This is incorrect treatment of the data as it is increasing all survival, horizontally shifting the entire survival curve to the right (allbeit by a small amount). There are 2 appropriate/precedented options in this instance:   1. Add a small time only to the observations with 0 time (I.e. t[which(t == 0)] <- 0.0001) 2. Exclude the observation with 0 time to avoid survival curve starting <100% in survival   We elected to do the former during review, which results in the following fit table:  (scenario “basecase” in the code)   |  |  |  | | --- | --- | --- | | dist | AIC | BIC | | exponential | 952.6848 | 955.5752 | | Weibull | 933.0287 | 938.8094 | | Gompertz | 952.2875 | 958.0682 | | llogis | 936.9018 | 942.6825 | | lognormal | 956.2611 | 962.0418 | | gengamma | 931.0847 | 939.7557 |   And similar visual fit.  EEPRU should explore the effect of excluding that data point as this may also be an appropriate approach in this case. | Negligible difference, just a small technical error |

1. Potential issue with discounting

|  |  |  |
| --- | --- | --- |
| **Description of problem** | **Description of proposed amendment** | **Result of amended model or expected impact on the result (if applicable)** |
| Discounting is not done in the typical way in the model code. Usually this is done by year unless continuous discounting using 1/ert is used. | Discounting is applied in cycle 0 in the following lines:  disc.fact.c <- 1 / (1 + dr.c) ^ (cycle.v / 4)  disc.fact.o <- 1 / (1 + dr.o) ^ (cycle.v / 4)  Such that the first result at discount of 3.5% is 0.9914365 and not 1. Further, when using step discounting, it is typical to use the floor of the year:  disc.fact.c <- 1 / (1 + dr.c) ^ floor(cycle.v / 4)  disc.fact.o <- 1 / (1 + dr.o) ^ floor(cycle.v / 4)  This avoids discounting at time 0 but still has an error of only having 3 quarterly cycles at discount factor of 1. Instead cycle.v should start at 0:  disc.fact.c <- 1/(1 + dr.c)^floor((cycle.v-1)/4)  disc.fact.o <- 1/(1 + dr.o)^floor((cycle.v-1)/4)  Alternatively 1/e^(rt) could be used for continuous discounting, which would then more accurately reflect using smaller increments of time. | Small influence, but makes the model in line with NICE STA reference case. |

1. EEPRU appear to have struggled with dirichlet distributions – we offer a solution

|  |  |  |
| --- | --- | --- |
| **Description of problem** | **Description of proposed amendment** | **Result of amended model or expected impact on the result (if applicable)** |
| It is evident from the commented out attempts to simulate a dirichlet distribution that EEPRU was unable to do this. | To assist with this, the package gtools contains an rdirichlet function which is quick and can be vectorised like so:  gtools::rdirichlet(1000, c(0.2,0.3,0.5) \* 1000)  The above example creates 1000 draws of the distribution that is commented out in the function at the bottom of FittingDistr.R (the probabilities are those that are commented out in EEPRU’s code). Note that when entering probabilities into a dirichlet distribution (which is intended to distribute numbers of observations, not probabilities) the probability values should be multiplied by N to avoid overestimating the uncertainty surrounding the probability vector in question. In the above example N of 1000 is used. We hope that EEPRU find this useful, and that the above allows them to draw from these distributions as intended.  If required, this function is very simple and can be reverse-engineered to take a fixed set of random numbers, allowing pre-generation of all uniform draws to ensure repeatability of analysis through separation of probability and distribution draws into different steps:  Source code for rdirichlet (in Rstudio you can get to this by highlighting the function call and pressing F2, or by calling the function without brackets):  l <- length(alpha)  x <- matrix(rgamma(l \* n, alpha), ncol = l, byrow = TRUE)  sm <- x %\*% rep(1, l)  x/as.vector(sm)  Simply change the rgamma call to a qgamma call, allowing entry of a set of uniform draws, allowing consistency between PSA runs. Note that the number of uniform draws required is a function of the vector length. | Unclear until EEPRU re-write the model to include the dirichlet draws in question. |

1. Random seed is repeatedly set at different stages throughout the code

|  |  |  |
| --- | --- | --- |
| **Description of problem** | **Description of proposed amendment** | **Result of amended model or expected impact on the result (if applicable)** |
| The random seed is set as a hard coded value in several places “set.seed(123)”  This also means that changing the amount of values that are generated (e.g. adding a parameter to the model) will change the entire PSA result (for all number generation subsequent to the additional element), as the same random numbers are not used in either run. | The gold standard would be to make a function to generate all of the required uniform/probability draws and orgamise them neatly into a list structure to inform all of the different quantile draws from the different distributions required for the PSA (in the form of 2 functions, one to generate random uniform draws, one to convert them into draws from different distributions). This would ensure repeatable PSA as the same random numbers would be used for the same parameters when the same seed is used (with some effort to ensure this).  As the code is long and difficult to follow, we have refrained from testing the impact of changing the random seed on the results to examine whether seed 123 is an outlier or not. | Unclear |

1. We are not sure if some patients go untreated in this model

|  |  |  |
| --- | --- | --- |
| **Description of problem** | **Description of proposed amendment** | **Result of amended model or expected impact on the result (if applicable)** |
| In the line  mutate(Salvage = if\_else(Secondline == "Salv" | Secondline == "Salv2", "Y", "N"))  Are we correct to assume that EEPRU is assuming that some treatments that fail in the first line do not have salvage options and the patient is simply left to die with no treatment? This does not appear to align with the efficacy that is applied to such patients (who in the no-cefiderocol state of the world would indeed have no effective treatment option, but the model appears to assign the efficacy from a colistin-inclusive regimen to them) | Assumption is clinically implausible, doctors will always attempt to treat patients, even if the chance of efficacy is low. However, also doesn’t algin with our understanding of the modelling assumptions.  Could EEPRU explain this, and the framework of assumptions on which the model is based? The table in the EEPRU report suggests no need for change in some circumstances where currently there is no effective medicine, and we are having difficulty reconciling this. | Unclear, but potentially large |

1. Standard NICE precedent days in a year not used

|  |  |  |
| --- | --- | --- |
| **Description of problem** | **Description of proposed amendment** | **Result of amended model or expected impact on the result (if applicable)** |
| On average there are 365.25 days in a year, so lines like the below are incorrect:  st\_selected <- summary(model, t = seq(0, (365) \* 40 - 365 / 4, 365 / 4), ci = FALSE)[[1]]$est | Use the correct number of days in an average year | Likely small |

1. There is an error in the code assigning rownames

|  |  |  |
| --- | --- | --- |
| **Description of problem** | **Description of proposed amendment** | **Result of amended model or expected impact on the result (if applicable)** |
| In the below line in Run\_models.R, the colnames of outAll are set twice in two successive commands, and the second one is for an incorrect number of rows (21 vs the 12 rows in OutAll)  rownames(OutAll) <- c(  "Death1",  "TxSucc1",  "FtxNeeded ",  "TxSucc1Deathat30",  "TxSucc1Survat30\_noAKI",  "TxSucc1Survat30\_AKI",  "Death2",  "Disch2\_noAKI",  "Disch2\_AKI1or2",  "Death\_all",  "SurvnoAKI\_all " ,  "AKI\_all " ,  "CKD\_all" ,  "hosp\_loS" ,  "Tx\_cost " ,  "AKI\_cost\_inhosp " ,  "Tot\_cost\_inhosp" ,  "tot.disc.LTcost.pp" ,  "tot.disc.lys.pp" ,  "tot.disc.qalys.pp" ,  "ppNHE "  ) | Errors in the code should be fixed | Most likely none, potential for downstream errors |

1. Patient incidence is assumed the same month to month

|  |  |  |
| --- | --- | --- |
| **Description of problem** | **Description of proposed amendment** | **Result of amended model or expected impact on the result (if applicable)** |
| In lines 109 and 110 of 1\_pop\_level\_benefit.R the following is done  #adjust for time (from 7 to 12 months)  temp1a<-abs\_pop\_mds/7\*12  temp1b<-abs\_pop\_es/7\*12  Yet all other rate conversions in the model are done using an exponential assumption. Is it correct to simply divide up incidence, particularly when non-linear growth is a factor? | EEPRU should reevaluate their approach to calculating patient numbers as numbers change each month, and growth is non-linear. | Unclear but potentially large. |

1. No indication that the final growth rate was validated

|  |  |  |
| --- | --- | --- |
| **Description of problem** | **Description of proposed amendment** | **Result of amended model or expected impact on the result (if applicable)** |
| The growth projection assigned in around line 220 looks strange. Is this the annual growth rate in a damped trend?  (based on pop\_change\_cpembl when pop\_grow == 1)    Is a growth rate of 9.3% in year 2 and 1.4% in year 20 realistic? Were these values validated directly with clinical experts? | EEPRU should validate that the growth rate in their base case is assumed to be very low 20 years from now. Could EEPRU confirm that the 1.4% growth rate in 2041 despite a 9.3% rate in 2021 has been directly presented to experts for validation? | Potentially large if damped, or even standard trends include unrealistically low projections of incidence growth in the future. Such a scenario would mean that EEPRU have underestimated PNHE, potentially considerably. |

1. Cost of treatment does not appear to be a function of length of stay in the model

|  |  |  |
| --- | --- | --- |
| **Description of problem** | **Description of proposed amendment** | **Result of amended model or expected impact on the result (if applicable)** |
| In the below code in Outcomefct\_PSA.R  tx1\_cost\_S = case\_when(  empiric & Emptx == "NCA" ~ c\_nca\_E\_S + c\_admin\_rdm ,  empiric & Emptx == "CA" ~ c\_ca\_E\_S + c\_admin\_rdm,  empiric & Emptx == "New" ~ c\_admin\_rdm,  T ~ 0  )    tx1\_cost\_F = case\_when(  empiric & Emptx == "NCA" ~ c\_nca\_E\_F + c\_admin\_rdm ,  empiric & Emptx == "CA" ~ c\_ca\_E\_F + c\_admin\_rdm,  empiric & Emptx == "New" ~ c\_admin\_rdm,  T ~ 0  )  It is clear that cost of treatment is not linked to length of stay, despite length of stay being a parameter with 2nd order uncertainty as presented in Figure 7 within the report. These values are then put into the long dplyr chain mutate call like so:  Tx\_cost = round(  TP\_TxSucc1 \* tx1\_cost\_S + (1 - TP\_TxSucc1) \* tx1\_cost\_F + TP\_FtxNeeded \*  tx2\_cost,  digits = 2  )  Therefore just a weighted average of a fixed cost for sucessful and failed treatment, therefore no variation of LoS (a highly uncertain value) in the PSA. Consequently, it is evident that cost of treatment is not a function of duration of treatment received. | Important factors such as this one should be varied in the probabilistic base case. | Unclear |

1. Some data read in as text

|  |  |  |
| --- | --- | --- |
| **Description of problem** | **Description of proposed amendment** | **Result of amended model or expected impact on the result (if applicable)** |
| In the line  susceptibility <- as.matrix(read.csv(file="data\_pop/susceptibility.csv", header=TRUE))  This reads the data in as text as some columns are not numeric, which as.matrix() then interprets as character.  Simply reading in as a data.frame would then preclude all the as.numeric() calls in the following lines, reducing the amount of time R spends converting data types.  res\_nca\_cpeoxa<-(1-as.numeric(susceptibility[which(susceptibility[,"Bug\_cat"]=="Cat1" & susceptibility[,"Scenario"]=="basecase"),"susc\_nca"]))  sus\_new\_cpeoxa<-as.numeric(susceptibility[which(susceptibility[,"Bug\_cat"]=="Cat1" & susceptibility[,"Scenario"]=="basecase"),"susc\_new"])  res\_nca\_cpembl<-(1-as.numeric(susceptibility[which(susceptibility[,"Bug\_cat"]=="Cat2" & susceptibility[,"Scenario"]=="basecase"),"susc\_nca"]))  sus\_new\_cpembl<-as.numeric(susceptibility[which(susceptibility[,"Bug\_cat"]=="Cat2" & susceptibility[,"Scenario"]=="basecase"),"susc\_new"])  res\_nca\_psambl<-(1-as.numeric(susceptibility[which(susceptibility[,"Bug\_cat"]=="Cat3" & susceptibility[,"Scenario"]=="basecase"),"susc\_nca"]))  sus\_new\_psambl<-as.numeric(susceptibility[which(susceptibility[,"Bug\_cat"]=="Cat3" & susceptibility[,"Scenario"]=="basecase"),"susc\_new"]) | Improving computational efficiency could make a considerable difference to the usability of the model in general, and may allow probabilistic scenario analyses | Most likely none, potential changes to scenario analysis results as could be probabilistic rather than deterministic |

1. Some scenarios are excluded from presentation with no justification or notification within the report

|  |  |  |
| --- | --- | --- |
| **Description of problem** | **Description of proposed amendment** | **Result of amended model or expected impact on the result (if applicable)** |
| Can EEPRU please explain why the following scenarios are deleted from the results:  cazavi\_all<-cazavi\_all[!rownames(cazavi\_all)%in%c("p\_bug\_0","p\_bug\_100","thresh15","thresh30","dr1.5"),]  cefide\_all<-cefide\_all[!rownames(cefide\_all)%in%c("p\_bug\_0","p\_bug\_100","thresh15","thresh30","dr1.5"),]  These scenarios clearly have a large PNHE impact, and move the results counterintuitively, both of which making them important to present. | Improve clarity throughout the cost-effectiveness model, and explain both decisions made and justifications either inside the code or in accompanying documentation. | Unclear – omission of relevant scenario analyses from the results presented to decision makers, leading to misrepresentation of face validity, clinical plausibility, and decision uncertainty. |

1. Parameters are varied in the code without stating the distribution used in the report

|  |  |  |
| --- | --- | --- |
| **Description of problem** | **Description of proposed amendment** | **Result of amended model or expected impact on the result (if applicable)** |
| We notice that EERPU vary the LoS as a Gamma distribution:  cost\_rdm = cbind(  rep(NA, nsim),  rep(NA, nsim),  rep(c\_admin, nsim),  Gammafit(nsim, los\_MDS\_S, los\_MDS\_S\_se),  Gammafit(nsim, los\_MDS\_nonS, los\_MDS\_nonS\_se)  )  This was done without announcing explicitly within the report the distribution that was being assumed for different parameters.  Could EEPRU explain why this does not follow the distribution in the SEE performed on LoS exactly? | Distributional selection of parameters should be justified, particularly when there is a distribution as a result of the SEE, which should be replicated exactly in the probabilistic base case. | Unclear as related to uncertainty analysis. |



|  |  |  |
| --- | --- | --- |
| **Description of problem** | **Description of proposed amendment** | **Result of amended model or expected impact on the result (if applicable)** |
| No indication of model convergence is provided for the patient-level model. The only mention of convergence was in Appendix 16, which is not concerned with the convergence of the patient-level model. | Could EEPRU please present evidence that the 2000 iteration base case has converged? Could EEPRU also provide these results using a few different seeds? Please explore whether just presenting INHE is sufficient to test convergence as it is not a ratio like an ICER. | Potentially large if the patient level model is biased. |

(please cut and paste further rows as necessary)