REEEVR: Excel-to-R converter release manual
V2026.1.0.0-beta

Michael O'Donnell, Qian Xin, Howard Thom

February 2026

Contents

1 Acknowledgements **1**

2 How to Cite **1**

3 Introduction **1**
3.1 Beta Warning . . . 1
3.2 What is REEEVR? . . . 2
3.3 How does conversion work? . . . 2

4 How to use REEEVR: Excel-to-R converter **2**
4.1 Worked Example . . . 2
4.1.1 The program interface . . . 2
4.1.2 Pulling the required cells from the exemplar workbook . . . 3
4.1.3 Running the converter . . . 5
4.1.4 Validating the R code . . . 5
4.2 Instructions on how to run the converter . . . 5
4.3 Instructions on how to run the output . . . 6

5 Tips, Tricks and Caveats **6**
5.1 Caveats . . . 6
5.2 Tips and Tricks . . . 7
5.3 FAQ . . . 7

1 Acknowledgements

Code for project entitled "Reliable and Efficient Estimation of the Economic Value of medical Research (REEEVR)" funded by UK Medical Research Council grant MR/W029855/1.

2 How to Cite

Should you use this program in your work, or you wish to perform analysis on the program itself, please cite this work. Once the technical paper is released we will direct the citations to that paper, for now please cite the Github repository. We will aim to notify when the paper is released (which will hopefully occur before someone else writes a paper using this!).

3 Introduction

If you just want to get on with running the code, head to section 4.2

3.1 Beta Warning

The REEEVR conversion software is currently in a beta state. This means the following:

- Use of the conversion tool should be less clunky than in the alpha, and will likely streamline in later releases.

- Not all Excel functions are available to be converted. Any function that is not converted will be listed in the file *missing-func.log* and replaced in the output ".R" file with UNKNOWN_FUNCTION_SEE_LOGFILE(
- The status of all functions are listed in "Excel function list.xlsx"
- Code that has claimed to have successfully converted may not run succesfully in the R environment.
- Code that does run in the R environment may not give the correct results. Always check the output with the original input Excel file.
- Many other usability issues may exist.

3.2 What is REEEVR?

REEEVR is a multi-faceted project that aims to achieve two separate goals in the the field of health technology assessment (HTA). First, the project aims to create a converter able of taking Excel HTA models and output equivalent models in another programming language - in this case the R programming language. The second aspect of the project aims to improve the use of value of information (VOI) techniques by performing characterisation studies and implementing tools to aid in deciding which kind of VOI you may wish to use, and any considerations that are required when performing those decisions.

This document only concerns itself with the first aspect of the project: taking an Excel model and converting it to an R model. This process has several benefits. First, the models we have tested will run faster than the original Excel model, allowing for longer PSA runs or many short PSA runs to be gathered, which in turn makes using VOI more feasible. Secondly, by having the model in R, we no longer need to consider how to perform further analysis in Excel, and can instead take advantage of the wealth of R packages available on CRAN.

3.3 How does conversion work?

A full technical paper will be released at a future date describing the process in detail, but briefly, the converter is provided the details of important cells, such as costs, QALYs and willingness to pay, and creates a tree of cells that are required in the Excel sheet to give an answer. Each of the cells contains either a function or data that is required in the final answer. This allows us to convert each individual cell from Excel to R, and then arrange the R code such that it is in the correct order. Finally, the generated R code hooks into the REEEVR R library that provides R equivalents to the various Excel functions encountered during the conversion step. For a more in-depth explanation, we encourage you to watch the proceeding titled REEEVR - Automated Conversion of Excel to R from the R for HTA 2024 workshop.

4 How to use REEEVR: Excel-to-R converter

As previously mentioned some of these steps are a little clunky, but we are hoping your feedback will allow us to shape how the process works, and make this a usable piece of software. The program has been provided to you as a .exe file, and should run on Windows. While the program does rely on python to run, it *should* work even without python installed by yourselves due to the way we generated this executable file. If this is not the case please reach out ASAP.

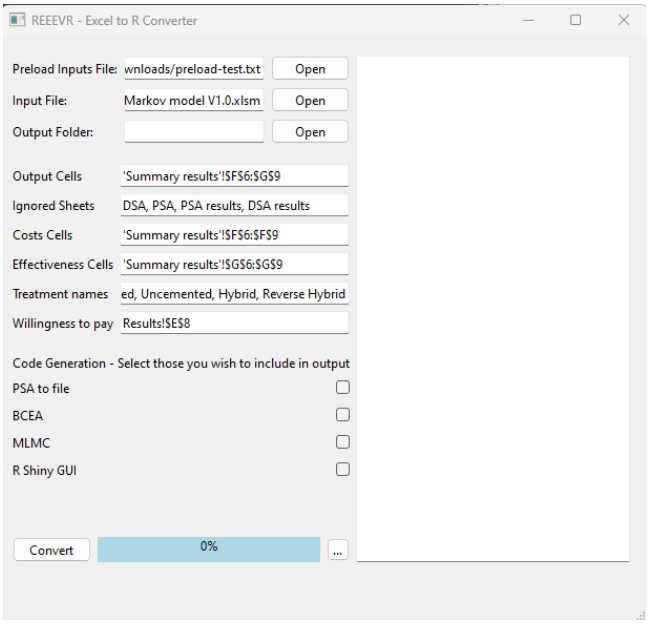
Mac users, I am sorry but I do not have access to a mac to both test this code and to then compile a mac equivalent executable. However, this program should be able to be converted trivially (oh how I hate that word!). Currently you will need to download the source package and either run directly from source or compile it yourself.

4.1 Worked Example

I personally find that a worked example is more useful than a set of instructions. But if this is not your kettle of fish then please skip the subsection for the following one.

4.1.1 The program interface

- Preload input file - A .txt file that contains all the values we wish to load into converter.
- Input file - The Excel workbook we wish to convert.
- Output folder - Output location of the conversion and subsequent PSA outputs.



- Output Cells - Cells we wish to be included in the PSA. **NOTE: Costs and QALYs are automatically added to this.** The cells need to have the sheet name prepended, and lists need to be comma separated. We can use Excel ranges if required. E.g. 'Summary results'!F6:G9, or 'Summary results'!$F$6:$G$9, or other valid variations.
- Ignored Sheets - Sheets which we don't care about. For example, PSA and DSA sheets, as we are calculating our own PSA and DSA.
- Costs Cells - The cells where the costs are finally calculated before being appended to the PSA sheet in Excel. Uses the same format as output cells, e.g. 'Summary results'!$F$6:$F$9.
- Effectiveness Cells - The cells where qualys are finally calculated before being appended to the PSA sheet in Excel. Uses the same format as output cells, e.g 'Summary results'!$G$6:$G$9.
- Treatment names - The list of treatment names in the order the costs/qualys are output. The list must be comma separated. In our example this is the types of hip replacements: Cemented, Uncemented, Hybrid, Reverse Hybrid.
- Willingness to pay - The cell containing the willingness to pay threshold e.g. 'Setup and run'!D14.
- Tick boxes - The tick box"BCEA" adds code to output graphs using the BCEA tool. This is still somewhat experimental so caution is advised.

4.1.2 Pulling the required cells from the exemplar workbook

In the future we plan on providing a VBA script to assist with this. For now this is a manual process.

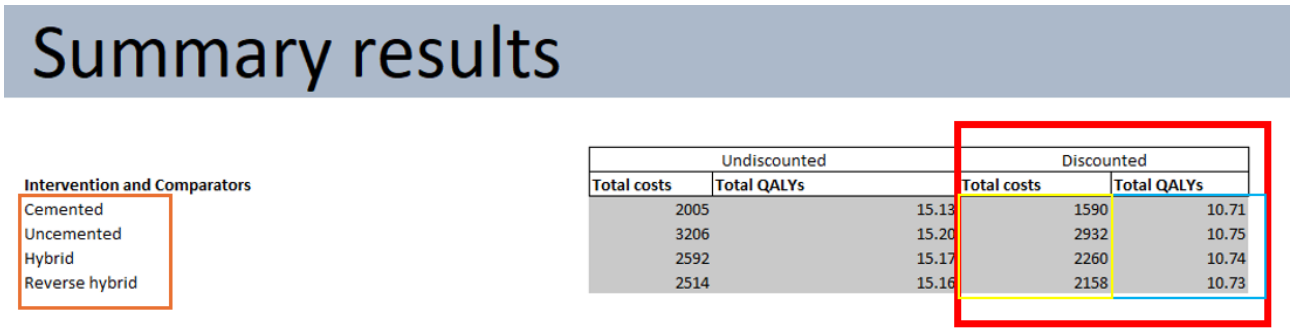


Figure 1: The costs and effectiveness/qualy/utilities are found in the summary results tab. These are updated every PSA cycle, so we know that these are suitable for calculating our PSA. The treatment names are also found here.
To fill out the converter, we type in 'Summary results'!F6:F9 for costs, 'Summary results'!F6:F9 for qualys, and 'Summary results'!F6:G9 for outputs. The treatment names are explicitly written out as Cemented, Uncemented, Hybrid, Reverse Hybrid. Note the order of the treatments matches the order of the costs/qualys

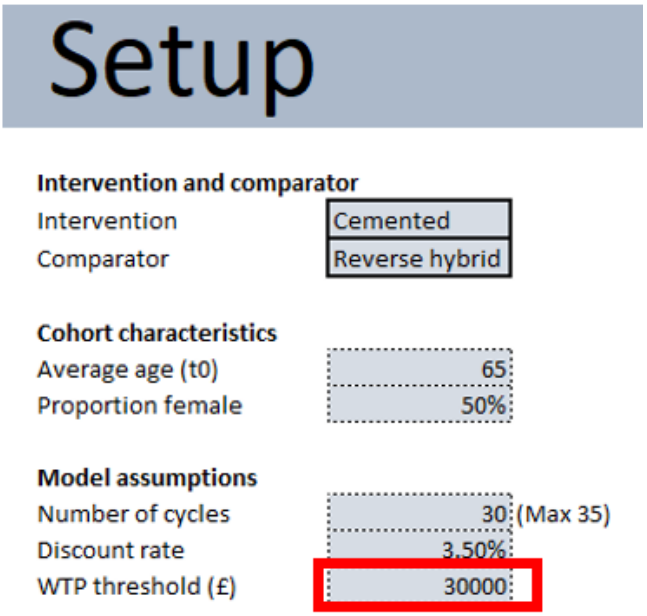


Figure 2: The willingness to pay threshold can be found on the Setup and Run page. Unfortunately the CEAC plane will not exceed this value. It is recommended to set this much higher to get the full CEAC graph.

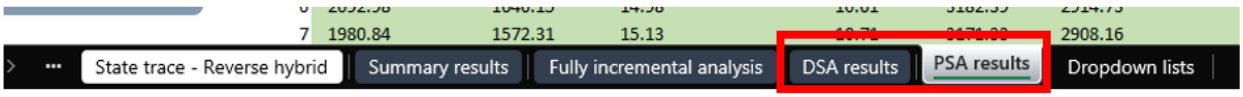


Figure 3: We do not need to include out DSA and PSA tabs. As such we can choose to ignore them. This saves on computation time for the converter.

4.1.3 Running the converter

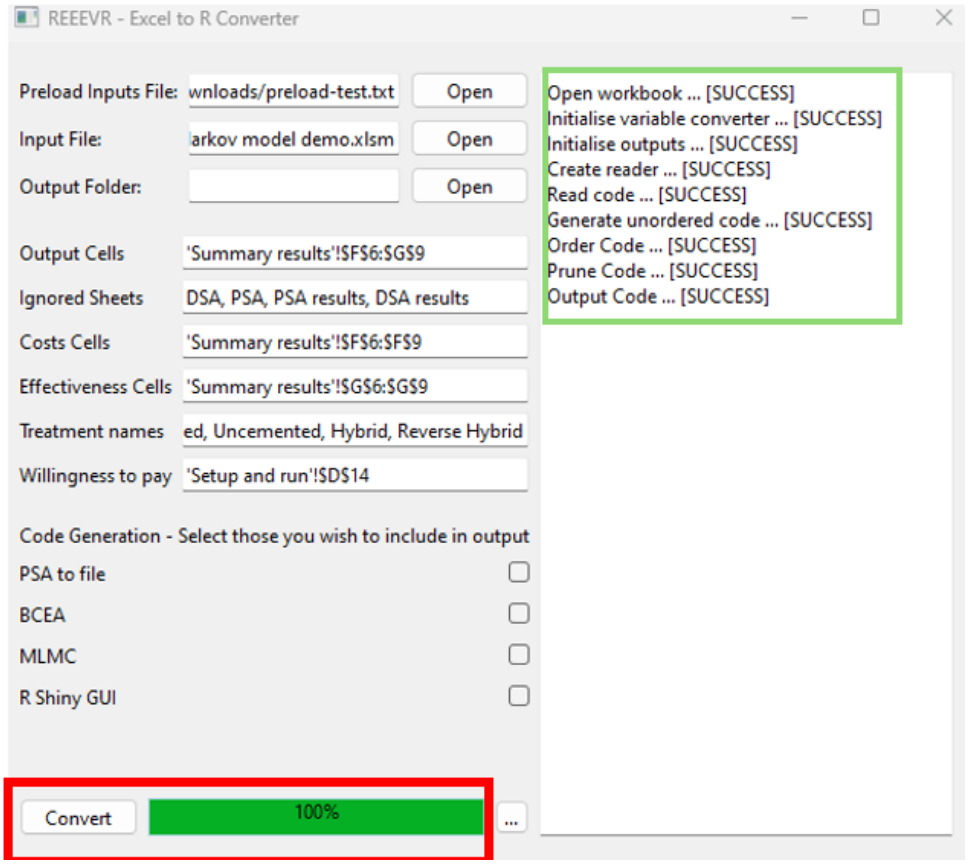


Figure 4: To run the converter, press the convert button highlighted in red. The current process the converter is working on is displayed on the right in the green box. If the converter fails the progress bar will turn red.

4.1.4 Validating the R code

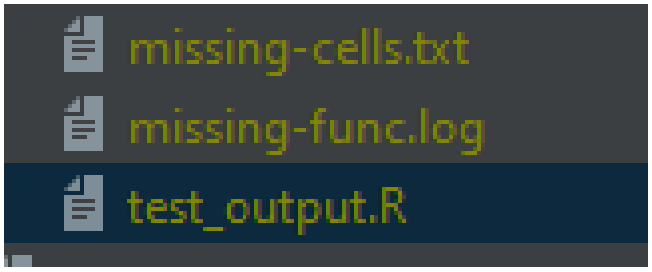


Figure 5: The converter will output up to three files. The R code is found in file test_output.R. The other two files state the functions not successfully converted (missing-func.log) and cells where they could not be successfully ordered (missing-cells.txt). The existence of these final two folders is not always cause for concern. For example MATCH is in the missing function log, but this is fine because the cells where MATCH was called were not needed in the final calculations, and so were discarded without impacting the program.

The output R code will require the reeevr package to be manually installed, and will default to the validation state. Head to section §4.3 to move through those instructions to run the code.

4.2 Instructions on how to run the converter

1. Run reeevrconverter.exe. **This may cause a blue windows popup saying this is an unknown file**. Press more info then run anyway.

2. Press open next to input file and select the exemplar workbook.

3. Go to the exemplar workbook and note down the sheet name and cells needed for the PSA. In the exemplar this is 'Summary results'!F6:G9. We could also have written: 'Summary results'!F6:F9, 'Summary results'!G6:G9. Note the comma separation.
4. Enter the cells for the costs. For the exemplar this is 'Summary results'!F6:F9
5. Enter the cells for the effectiveness (QALYs). For the exemplar this is 'Summary results'!G6:G9
6. Enter the treatment names in the order of the outputs. For the exemplar this is: Cemented, Uncemented, Hybrid, Reverse Hybrid.
7. Enter the willingness to pay cell. For the exemplar this is 'Setup and run'!D14
8. Add the sheet names of those sheets we do not wish to handle/convert. This is usually the DSA and PSA sheets.
9. Press convert - The output will be the name of the Excel workbook but with the extension .R

4.3 Instructions on how to run the output

1. The output will default to the validation state.
2. Replace boolean value TRUE with value FALSE for the variable converter_validate. This validation variable will be in the top few lines.
3. We have provided a folder called reeevr which contains the R source for the reeevr library. This will need to be manually built and installed and should have come in the download.
4. Open the reeevr R project file.
5. Press build at the top of Rstudio to reveal the dropdown.
6. Press install package.
   (a) If it complains you don't have Rtools installed, continue in this sublist, else skip this sublist.
   (b) Head to https://cran.rstudio.com/bin/windows/Rtools/rtools44/rtools.html and press the Rtools44 installer link.
   (c) Follow the Rtools installer instructions.
   (d) Once installed, go back to step 5
7. Once R has restarted, REEEVR is installed and we can run the R code.
8. Output files/folders will appear in the directory this is ran inside. If using the exemplar model these will be titled "Cemented", "Uncemeneted", "Hybrid", and "Reverse Hybrid".

5 Tips, Tricks and Caveats

As with all things in life, there are tips and tricks for making life a little bit easier, but there are also a myriad of caveats that can leave you more confused than not. In this sections we will aim to highlight these areas while providing the rational behind this process.

5.1 Caveats

- This converter CANNOT convert VBA code. It is beyond the scope of this project and unfortunately is quite difficult (I did look into this and dealing with 3 languages is hard enough, let alone 4!). At this point you may ask "how can it do a PSA then?". The short answer is that it looks for RAND(functions and replaces them with a vector of random numbers. The long answer can be found in the linked proceedings in §3.3.
- The converter cannot perform a DSA. There is the possibility this will be included in future updates, but we expect it is unlikely as it is both difficult to program and has little benefit, as most Excel workbooks can trivially provide the DSA as standard.
- Not all functions are implemented. This will be rectified with time.

- Clever Excel will likely break the converter. Some Excel users, sometimes referred to as Excel wizards or power users, keep up-to-date with the latest and greatest in Excel usage, and utilise niche or new functionality. Unfortunately, I am not a Excel wizard, and so likely have not included these use cases, and some of them would be impossible to include anyway (looking at you users putting python into the workbooks).

5.2 Tips and Tricks

- Preload files - If you plan on converting the same workbook over and over, it can get tedious to find the relevant cells. Use the preload file functionality that is shown in the worked example.
- Preload files part 2 - If you have multiple excel workbooks where all of cell locations are the same (say, in a standardised set of workbooks) the same preload script can be used so long as the file location is changed. Please double check everything is correct though!
- VBA generation of preload file - We have provided a small VBA script that, if copied into your Excel document, will run a macro that will attempt to create the preload file for you. **We highly recommend using this on a copy of your Excel workbook in case it goes wrong!**
- Changing cells in the Excel document - If you perform a conversion, then decide you want to change some values in the cells (say, updating a life table), so long as the converter hasn't been closed it will be able to run the conversion again without re-filling out the input details.
- Values can be changed in the R code - Lets say you set a willingness to pay in the Excel document of £20000, but you wish to change this to £30000. You could change the value in the Excel document, and rerun the conversion, or, you can open the R code, and use ctrl+f to find the variable/cell name that is the willingness to pay.
- The R code has structure (even when it doesn't look like it!) - Say you wish to change the value of cell E6 on sheet lifetables. If you do not wish to run the converter again, the variable in R will be of the form SHEETNAME_CELL, so in this case will be called lifetables_E6.

5.3 FAQ

Q: When running the R code, Rstudio gives an error saying can't find last parenthesis (closing bracket) when it loads the life table array.
A: Rstudio has a maximum character limit that it's console can run. In this case you have likely hit this limit. Try running instead using the "source" button. This will run the code without first entering it into the console.

Q: Does this program steal my data?
A: Nope, the program is completely self contained and local to your computer. While this is obviously an exercise in trust, the source code is available on the Github repository. Should you wish you can recompile directly from the source.

Q: Can I use this on sensitive data?
A: This depends. If you trust that we do not connect to the internet, or forward such data, then yes. However, your IT department may disagree without first seeing the source code. As mentioned above, this has been provided on the Github repository.

Q: Should I make a copy of my Excel document first?
A: Yes. While we have every confidence that we do not edit the Excel document in any way, it is prudent to be distrustful and keep a backup.

Q: The converter doesn't work with the error "Permission denied".
A: Close the Excel workbook. When open the file is sometimes protected and cannot be accessed by any other program. If you wish to look at both, I suggest opening the original and working on the copy.

Q: The converter runs, but gives columns with only 9 results
A: Your file is still in validation mode. Head to the top of the file and find the variable converter_validate. Set the boolean value to FALSE.