METHOD ARTICLE

REVISED Living HTA: Automating Health Economic Evaluation with R [version 2; peer review: 2 approved]

Robert A. Smith [1-3], Paul P. Schneider [1,3], Wael Mohammed [1,3]

[1]School of Health and Related Research, University of Sheffield, Sheffield, S1 4DA, UK
[2]Lumanity, Sheffield, S1 2GQ, UK
[3]Dark Peak Analytics, Sheffield, S11 7BA, UK

v2 **First published:** 21 Jul 2022, **7**:194
https://doi.org/10.12688/wellcomeopenres.17933.1
Latest published: 11 Oct 2022, **7**:194
https://doi.org/10.12688/wellcomeopenres.17933.2

Abstract

Background: Requiring access to sensitive data can be a significant obstacle for the development of health models in the Health Economics & Outcomes Research (HEOR) setting. We demonstrate how health economic evaluation can be conducted with minimal transfer of data between parties, while automating reporting as new information becomes available.
Methods: We developed an automated analysis and reporting pipeline for health economic modelling and made the source code openly available on a GitHub repository. The pipeline consists of three parts: An economic model is constructed by the *consultant* using pseudo data. On the data-owner side, an application programming interface (API) is hosted on a server. This API hosts all sensitive data, so that data does not have to be provided to the *consultant*. An automated workflow is created, which calls the API, retrieves results, and generates a report.
Results: The application of modern data science tools and practices allows analyses of data without the need for direct access – negating the need to send sensitive data. In addition, the entire workflow can be largely automated: the analysis can be scheduled to run at defined time points (e.g. monthly), or when triggered by an event (e.g. an update to the underlying data or model code); results can be generated automatically and then be exported into a report. Documents no longer need to be revised manually.
Conclusions: This example demonstrates that it is possible, within a HEOR setting, to separate the health economic model from the data, and automate the main steps of the analysis pipeline.

Keywords

HEOR, HTA, APIs, R, plumber

Open Peer Review

Approval Status ✓ ✓

	1	2
version 2 (revision) 11 Oct 2022	✓ view	
version 1 21 Jul 2022	? view	✓ view

1. **Mohsen Sadatsafavi**, University of British Columbia, Vancouver, Canada

2. **Devin Incerti**, EntityRisk Inc., Princeton, USA

Any reports and responses or comments on the article can be found at the end of the article.

Corresponding author: Robert A. Smith (rasmith3@sheffield.ac.uk)

Author roles: Smith RA: Conceptualization, Data Curation, Formal Analysis, Investigation, Methodology, Project Administration, Resources, Software, Visualization, Writing – Original Draft Preparation, Writing – Review & Editing; **Schneider PP**: Conceptualization, Methodology, Software, Writing – Original Draft Preparation, Writing – Review & Editing; **Mohammed W**: Methodology, Writing – Original Draft Preparation, Writing – Review & Editing

Competing interests: R.A.S. Is part of the Scientific Committee for R for HTA, an academic consortium whose main objective is to explore the use of R for cost-effectiveness analysis. P.P.S. and W.M have no competing interests to declare.

Grant information: This work was jointly supported by the Wellcome Trust Doctoral Training Centre in Public Health Economics and Decision Science (PHEDS) [108903, https://doi.org/10.35802/108903; 224853] and the University of Sheffield.
The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

How to cite this article: Smith RA, Schneider PP and Mohammed W. **Living HTA: Automating Health Economic Evaluation with R [version 2; peer review: 2 approved]** Wellcome Open Research 2022, **7**:194 https://doi.org/10.12688/wellcomeopenres.17933.2

First published: 21 Jul 2022, **7**:194 https://doi.org/10.12688/wellcomeopenres.17933.1

REVISED **Amendments from Version 1**

Since the previous version of the article we have done the following:

- Added additional text in the introduction section to outline the value of the contribution.
- Included a new figure, related to the above, which is more aesthetically pleasing than the one in the preprint.
- Refined the methods sections to improve the flow and clarity of the article, in particular we have added a paragraph to provide more information about the example use case for which we provide open source code.
- Added descriptions to the code chunks to make it clearer who is running the code chunks and where.
- Indented several lines of the code to make the code easier to follow.
- Added to the discussion section to include more information on deploying these types of APIs, and the limitations of different deployment methods.

The fundamental contribution of the paper, the general message and the open source code remains unchanged.

Any further responses from the reviewers can be found at the end of the article

Introduction

The development of economic models sometimes involves the transfer of sensitive data (e.g. individual patient or price data) between parties. This paper demonstrates how the use of application programming interfaces (API) allows data-owners in the Health Economics & Outcomes Research (HEOR) industry to collaborate with multiple partners on health economic decision models, while, retaining full control of their data. The use of an API furthermore makes it possible to streamline and automate reporting as new information becomes available, significantly reducing the financial and administrative burden of economic model updates.

To our knowledge this is the first publication to outline a process for automated reporting in HEOR, which we term Living HTA, and the first to demonstrate the process of sending health economic model algorithms to sensitive data using APIs.

Two other bodies of work are particularly relevant. The first is the OpenSafely initiative, which inspired this work. Williamson *et al.*[1] describe the OpenSafely interface, which was developed to analyse electronic health records data without the need to share confidential patient information:

> "secure software interface that allows detailed pseudonymized primary care patient records to be analysed in near-real time where they already reside - hosted within the highly secure data centre of the electronic health records vendor — to minimize the reidentification risks when data are transported off-site".

The method described in this paper has a similar objective, but aims to protect sensitive information in the HEOR sector.

The second work, a publication by Adibi *et al.*[2], describes a cloud-based model accessibility platform for models developed in R. The authors make the case for cloud based platforms to improve the accessibility, transparency and standardization of health economic models, particularly highlighting the benefits of hosting computationally burdensome models on remote servers. The authors outline a framework for hosting models, contained within R packages, which are run using calls to an API. A set of standardized model call functions provide the user of the API with enough information to pass the necessary parameters to the model, run the model, and retrieve the necessary results directly into an R session. The publication is the first, to our knowledge, to discuss the enormous implications that remote model hosting could have in the HEOR industry.

We combine elements from both Adibi *et al.*[2] and the OpenSafely initiative, and provide an open-source code base which demonstrates the ease with which APIs can be deployed on remote servers to avoid the need to share sensitive data, and enabling automation of model updates. In short, we propose that *data-owners* (e.g. pharmaceutical companies or governments), with support from health economists, host their own model accessibility platforms. We therefore see the primary contribution of this paper as being the development of a system in which the health economic model and the data are two separate entities, and the health economic model is sent to the data rather than the other way around. By working in this way, it is theoretically easier to share the model without the sensitive data on which it is run, although making the model open source is not a requirement. Our hope is that providing these materials will encourage others to use these methods to improve the transparency, accessibility and efficiency of health economic models.

Methods

A common problem in health technology assessment is a situation in which a data-owner (e.g. a company in the pharmaceutical industry) holds sensitive data, but requires the services of a consultant to conduct health economic modeling. For example the data-owner may have interim clinical trial data at the patient level, and may need an external health economist (the consultant) to build a state transition model to determine the cost-effectiveness of the treatment. Currently the data-owner may be required to send the consultant cuts of data as the trial progresses (e.g. 12 month, 18 month, 24 month). This is burdensome and results in multiple iterations of sharing sensitive data. We propose a solution that does not require the sharing of data between parties and allows for automated updates to the analysis as data is updated.

This automated analysis and reporting pipeline for health economic modeling consists of three parts:

- An economic model. The model can initially be developed using pseudo data – that is, randomly generated data, which has the same format as the actual data, but does not contain any sensitive information.
- An API, hosted by the data-owner side. It can be generated using the R package plumber. An automated workflow is created. This workflow sends the economic model to the data-owner's API. The model is then run within the data-owner's server. The results are sent back to the *consultant*, and a (PDF) report is automatically generated using RMarkdown[3]. This API server hosts all sensitive data, so that data does not have to be sent between parties.
- All of these processes can be controlled with a web-based user-interface. We provide an example user-interface built in the R shiny package[4], based on the tutorial application in our previous paper[5]. This application allows users to select input parameters with which to query the API, and view the results. This allows non-technical stakeholders to interact with the model in real time, while allowing the data-owner to retain control of the data. The application will always reflect the data on the data-owner's server, and the model hosted by the *consultant* at the time of use.

Figure 1 shows a schematic of the interaction between the data-owner's API and the *consultant's* automated workflow. All of the methods discussed in this paper, as well as the code for the demonstration app can be found contained within an open access GitHub repository (see *Software availability*[6]).

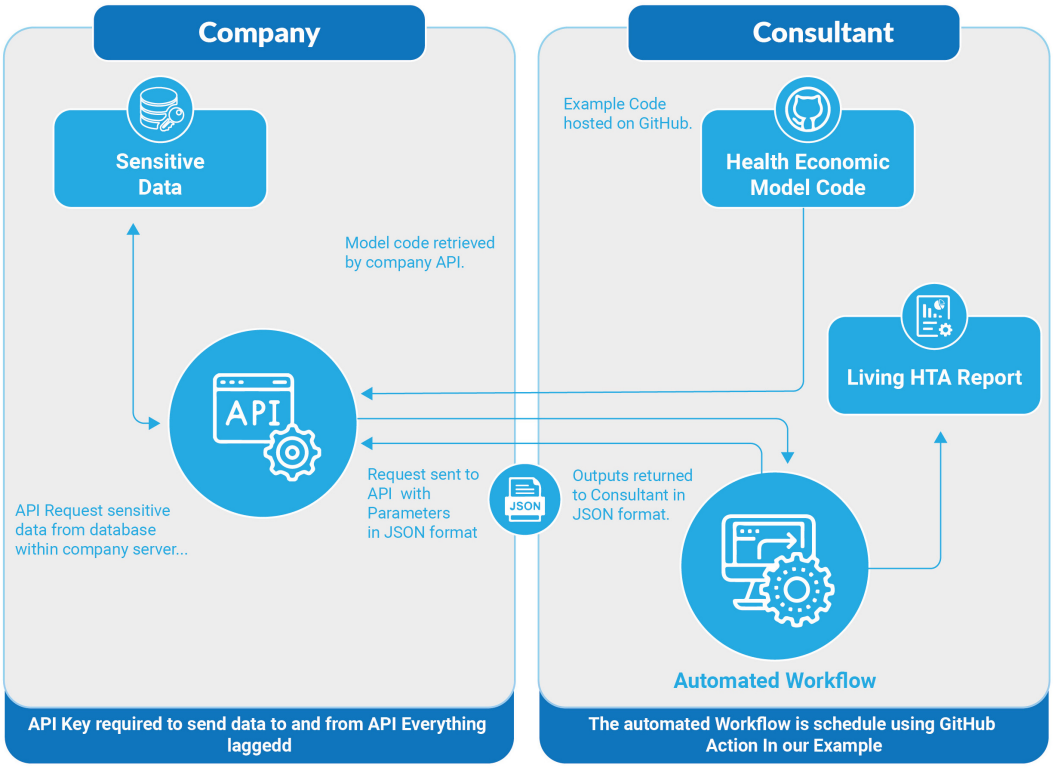

Figure 1. Schematic showing the interaction between the company API (application programming interface) and the consultant automated workflow. HTA, Health Technology Assessment; JSON, Javascript Object Notation.

The economic model

This model code has been adapted from the Decision Analysis in R for Technologies in Health (DARTH) group's open source Cohort state-transition model (the Sick-Sicker Model) which is discussed in Alarid-Escudero *et al.*[7] with open source code available online[8]. The code includes several functions, but for the purpose of this example we can treat the model as a black box, as a single function called *run_model* which runs the DARTH Sick Sicker model. The *run_model* function takes a single argument, *psa_inputs*, which is a data-frame containing Probabilistic Sensitivity Analysis parameter inputs for the model variables that are allowed to vary. Additional, sensitive parameters (including treatment costs and hazard ratio for treatment B) are not allowed to be varied by the API request and will be informed by the values held by the data-owner.

The data-frame has four columns:

- *parameter* - the name of the parameter (e.g. p_HS1)
- *distribution* - the distribution of that parameter (e.g. "beta")
- *V1* - the first parameter for the distribution in R (for beta this would be *shape1*, for normal this would be *mean*)
- *V2* - the second parameter for the distribution in R (for beta this would be *shape2*, for normal this would be *sd*)

The *run_model* function returns a data-frame with six columns. The first three columns are costs for each treatment option, and the second three columns are Quality Adjusted Life Years (QALYs) for each treatment option. Each row represents the result of the model run for a set of inputs.

The function described is designed as a simple reproducible example. The proposed method is flexible to any inputs, model structure, and outputs.

The API

An application programming interface is a set of rules, in the form of code, that allow different computers to interact with one another in real time. Whereas user-interfaces such as those generated by the R package *shiny* allow humans to interact with data, APIs are designed to enable computers to interact with data[4].

When a 'client' application wants to access data, it initiates an API call (*request*) via a web-server, to retrieve the data. If this request is deemed valid, the API makes a call to an external program/server, the server sends a response to the API with the data, and the API transfers the data to the 'client' application. In a sense, the API is the broker (or middle-man) between two systems.

There are numerous benefits to APIs:

- in supporting programmatic access. In contrast to what web applications offer (for example shiny apps), APIs allow users to access data, or other utilities (for example, proprietary applications) programmatically. Programmatic access enables users to invoke actions through an application or third-party tool. For example, R users can write a function that fetches or analyses data via an API and use it in their workflow as any other user-defined function.
- in allowing cross-platform communications. Statisticians and decision-model developers can use different programming languages or packages. For example, APIs can allow a decision analytic model, developed in C++ to programmatically utilise data from a bayesian meta-analysis performed using the Python programming language.

- in aiding speed of collaboration between institutions, ensuring inputs and outputs are standardised so that applications can 'talk' to one another. Users from one institution need not to take into account the software or package used by their partners, but focus on how they would interact with the expected data.
- in security, eliminating the necessity to share data manually (e.g. via email). All interaction with data can be logged and access can be restricted by passwords and by limiting IP address access. For example, APIs can safely allow statisticians to programmatically accumulate sub-group summary-statistics from securely stored trial-data to inform a network meta-analysis.
- in expanding sharing avenues. For example, APIs can allow institutions to give limited access to their proprietary tools such as in-house decision-analytic models. Users of such tools can pass their data to the model and receive the respective outputs via the API.
- eliminating computational burden on the client side (since all computation is done on the API owner side).

There are lots of different implementations of APIs, but the main focus of this paper is on *Partner APIs*, which are created to allow data transfer between two different institutions. This requires a medium level of security, usually through the creation of access keys that are shared with partners.

In the examples below we use Javascript Object Notation (JSON), a data interchange format that is commonly used to transfer information between computers, to pass information to and from our API. Since the model is written in R, we convert back and forth between JSON and R data formats using the *jsonlite* R package[9].

Creating the API using plumber

The R package *plumber* allows programmers to create web APIs by decorating R source code with roxygen-like comments[10,11]. These functions are then made available as API endpoints by plumber.

The API can be called using a number of HTTP request methods (also known as HTTP verbs). The most-commonly used methods POST, GET, PUT, PATCH, and DELETE correspond to create (POST and PUT), read (GET), update (PATCH), and delete (DELETE) operations. These annotations generate the API's endpoint(s) and specify the operation(s) or response(s) the respective R function is responsible for generating. The below example shows the 'GET' request (the default for web-browsers).

The code below gives an example function which echos a message. The function takes one input, a string with the message, and outputs the message contained within a list. If this function was created in R it would return a list containing some text, like this: The message is: 'example_msg'.

```
#* Echo back the input
#* @param msg The message to echo
#* @get /echo
function(msg="") {
  list(msg = paste0("The message is: '", msg, "'"))
}
```

The code for the model function uses the same principles, but is much more developed. There are three arguments to the model API; *path_to_psa_inputs*, *model_functions* and *param_updates*.

The core API function created by plumber sources the model functions from software development website GitHub, obtains the model parameter data from within the API, and then overwrites the rows of the parameter updates that exist in *param_updates*. It then runs the model functions using the updated parameters, post-processes the results, checks that no sensitive data is included in the results, and then returns a data-frame of results. This entire process occurs in the server on which the API is hosted, with inputs and outputs passed to the API over the web in JSON format.

Code chunk 1 - Generating the API **(this code is run on the data-owner's server)**

```
library(dampack)
library(readr)
library(assertthat)

#* @apiTitle API hosting sensitive data
#*
#* @apiDescription This API contains sensitive data, the data-owner does not
#* want to share this data but does want a consultant to build a health
#* economic model using it, and wants that consultant to be able to run
#* the model for various inputs
#* (while holding certain inputs fixed and leaving them unknown).

#* Run the DARTH model
#* @serializer csv
#* @param path_to_psa_inputs is the path of the csv file containing the PSA parameters
#* @param model_functions gives the GitHub repository to source the model code
#* @param param_updates gives the replacement values of the editable parameters
#* @post /runDARTHmodel
function(path_to_psa_inputs = "parameter_distributions.csv",
         model_functions = paste0("https://raw.githubusercontent.com/",
                                  "BresMed/plumberHE/main/R/darth_funcs.R"),
         param_updates = data.frame(
           parameter = c("p_HS1", "p_S1H"),
           distribution = c("beta", "beta"),
           v1 = c(25, 50),
           v2 = c(150, 70)
         )) {


  # source the model functions from the shared GitHub repo...
  source(model_functions)

  # read in the csv containing parameter inputs
  psa_inputs <- as.data.frame(readr::read_csv(path_to_psa_inputs))

  # for each row of the data-frame containing the variables to be changed...
  for(n in 1:nrow(param_updates)){

     # update parameters from API input
     psa_inputs <- overwrite_parameter_value(
                               existing_df = psa_inputs,
                               parameter = param_updates[n,"parameter"],
                               distribution = param_updates[n,"distribution"],
                               v1 = param_updates[n,"v1"],
                               v2 = param_updates[n,"v2"])
  }

```

```
  # run the model using the single run-model function.
  results <- run_model(psa_inputs)

  # check that the model results being returned are the correct dimensions
  # here we expect a single dataframe with 6 columns and 1000 rows
  assertthat::assert_that(
    all(dim(x = results) == c(1000, 6)),
    class(results) == "data.frame",
    msg = "Dimensions or type of data are incorrect,
  please check the model code is correct or contact an administrator.
  This has been logged"
  )

  # check that no data matching the sensitive csv data is included in the output
  # searches through the results data-frame for any of the parameter names,
  # if any exist they will flag a TRUE, therefore we assert that all = F
  assertthat::assert_that(all(psa_inputs[, 1] %in%
        as.character(unlist(x = results,
                            recursive = T)) == F))

  return(results)

}
```

Deploying an API

There are numerous providers of cloud computing services. The most convenient, yet not the cheapest, service is offered by RStudio Connect. An account is required for this, but provides the benefit of being able to deploy the API directly from the Rstudio integrated development environment. RStudio have a blog on how to publish an API created using plumber to RStudio connect here.

Interacting with the API

We first show how to run the model from an R script, calling the API and retrieving the results of the model run. We then show how to use GitHub actions to automate the process, running the R script when triggered by an event (e.g. a data-update) or a scheduled time (e.g. the 1st of each month).

Interact with the API from an RScript. We use the *POST* function from the *httr* package to query the API[12] - as shown in the code chunk below. This function requires an internet connection. We provide values for several arguments:

- *url* - the URL of the RStudio Connect server hosting the API we have created using plumber.
- *path* - the path to the API within the server URL.
- *query* & *body* - objects passed to the API in list format, with names matching the plumber function arguments.
- *config* - allows the user to specify the KEY needed to access the API.

The *content* function attempts to determine the correct format for the output from the API based upon the content type. This function ensures that the result object is a dataframe.

The script then then goes on to save the data and generate a PDF report from the outputs using the RMarkdown package[3], the code for which can be found here. The R-Markdown report uses functions adapted from the *darkpeak* R package.

Code chunk 2 - Query the API, retrieve model results and generate report **(this code is run by the consultant)**

```
# remove all existing data from the environment.
rm(list = ls())

library(ggplot2)
library(jsonlite)
library(httr)

# run the model using the connect server API
results <- httr::content(
  httr::POST(
    # the Server URL can also be kept confidential, but will leave here for now
    url = "https://connect.bresmed.com",
    # path for the API within the server URL
    path = "rhta2022/runDARTHmodel",
    # code is passed to the data-owner API from GitHub.
    query = list(model_functions =
                   paste0("https://raw.githubusercontent.com/",
                          "BresMed/plumberHE/main/R/darth_funcs.R")),
    # set of parameters to be changed ...
    # we are allowed to change these but not some others
    body = list(
      param_updates = jsonlite::toJSON(
        data.frame(parameter = c("p_HS1","p_S1H"),
                   distribution = c("beta","beta"),
                   v1 = c(25, 50),
                   v2 = c(150, 100))
      )
    ),
    # we include a key here to access the API here the key is a env variable
    config = httr::add_headers(Authorization = paste0("Key ",
                                                      Sys.getenv("CONNECT_KEY")))
  )
)

# write the results as a csv to the outputs folder...
write.csv(x = results,
          file = "outputs/darth_model_results.csv")

source("report/makeCEAC.R")
source("report/makeCEPlane.R")

# render the markdown document from the report folder,
# passing the results dataframe to the report.
rmarkdown::render(input = "report/darthreport.Rmd",
                  params = list("df_results" = results),
                  output_dir = "outputs")
```

Living HTA - scheduling model report updates. Once the API is created and hosted online, it can be called any time. The advantage of this is that any updates to either the model code, or the data used by the model, can be undertaken separately and the model re-run by either party. Calls to the API can also be scheduled at routine intervals. This would enable the health economic evaluation model report to be updated, without human interaction, at regular intervals to reflect the most up-to-date data.

In the example below we show how a GitHub Actions (other providers available) workflow can be used to automate an update to a health economic evaluation[13]. The workflow runs at 0:01 on the first day of every month or any time there are changes made to the source code. It first clones the GitHub repository on a GitHub actions Windows 2019 server, then install the necessary dependencies, before running the script described above to generate the model report. It creates a pull request to the repo with this new updated report. If GitHub is not the preferred location of report storage, it is possible to send the report via email or save to cloud storage solutions such as Google Drive or Dropbox.

Code chunk 3 - Automated report updates

```
on:
  push:
    branches:
    - main
  schedule:
    - cron: '1 1 1 * *'

name: Run DARTH model via API
jobs:
  createPullRequest:
    runs-on: windows-2019
    env:
      GITHUB_PAT: ${{ secrets.GITHUB_TOKEN }}
  # Load repo and install R
    steps:
    - uses: actions/checkout@master
    - uses: r-lib/actions/setup-r@master

    - name: Setup pandoc
      uses: r-lib/actions/setup-pandoc@v2
      with:
        pandoc-version: '2.17.1.1'

    - name: Install TinyTeX
      uses: r-lib/actions/setup-tinytex@v2
      env:
          # install full prebuilt version
          TINYTEX_INSTALLER: TinyTeX

    - name: Install dependencies
      run: |
          install.packages(
          c("reshape2", "jsonlite", "httr", "readr", "rmarkdown", "markdown")
          )
          install.packages(
          "scales", dependencies = TRUE, repos = 'http://cran.rstudio.com/'
          )
          install.packages(
          "ggplot2", dependencies = TRUE, repos = 'http://cran.rstudio.com/'
          )
      shell: Rscript {0}

    - name: Run the model from API and create report
      env:
         CONNECT_KEY: ${{secrets.PLUMBER_SECRET}}
      run: |
          source("scripts/run_darthAPI.R")
      shell: Rscript {0}

    - name: Create Pull Request
      uses: peter-evans/create-pull-request@v3
      with:
        token: ${{ secrets.GITHUB_TOKEN }}
        commit-message: Automated Model Run from API
        title: 'Living HTA Automated Model Run'
        body: >
          Automated model run
        labels: report, automated pr
```

Results

All source code for the API, the economic model, the automated model update framework, and the example dataset are available online (see *Software availability*[6] and *Underlying data*[14]).

The most up to date automated report, based on the data held on the exemplar API (hosted on RStudio Connect), can always be found here.

The method has been validated by two co-authors using Windows and MAC with example data (see *Underlying data*[14]). Those validating the method were able to run the model with updated parameter values without access to sensitive data, were able to trigger the automated report generation based on existing sensitive data, and were able to query the model through an example R-Shiny application, hosted on GitHub (see *Software availability*[6]). However we are keen to validate the method further, and invite collaboration. A live exemplar API is currently hosted by Lumanity (using the exact source code provided open access). If the reader is interested to test the functionality of the API please contact the corresponding author, who can provide the key.

Discussion

As the collection and storage of large data sets has become more commonplace in health & health care settings, this data is increasingly being used to inform decision making. However, concerns about the security of this data, and the ethical implications about linked data sets, make the owners of this valuable resource particularly reluctant to share data with health economic modelling teams. The ability to host APIs on data-owners' servers, and send the model to the data rather than the data to the model, is one potential solution to this problem. The example described in this paper may be relatively simple, but gives a tech savvy health economist everything they need to set up a modelling framework which does not rely on the sharing of data by a pharmaceutical company (or other *data-owner*).

The framework described has a number of benefits.

- Firstly, no data needs to leave the data-owner's server. This is likely to significantly reduce administrative burden for both the *data-owner* and the *consultant*, and reduce the number of data-leaks.

- Separating the data from the model has significantly improved the transparency of the health economic model. Allowing others to critique methods & hidden structural assumptions, test the code and identify bugs should improve the quality of models in the long run. It also enables the pool of people working on developing the health economic model and accompanying user-interface to be widened, without concern for confidentiality & data security. For example a shiny application could be developed for a model built under this framework without the programmer needing access to any sensitive data or information. However, if it is necessary to restrict access to model code, it is possible for the API to be passed 'private' source code. As keen proponents of open source modeling that was determined to be beyond the scope of this publication.

- The computational burden of the model is handled on a remote server. The power of these servers is typically considerably greater than that of a typical personal computer, speeding up model run time considerably. This is likely to be especially important for models that incorporate uncertainty through monte-carlo sampling algorithms which can be parallelized on machines with multiple cores[15], for example probabilistic one way sensitivity analysis[16] or partial expected value of perfect information[17].

- The use of APIs to perform distinct tasks can improve interoperability within the field of health economics. Different modules, or tasks within a modelling framework can be written in different languages (e.g. R, Python, Julia & C++) and linked using APIs. This is likely to improve collaboration between different sub-disciplines, which often use different languages (e.g. health economists in R and data-scientists in Python).

- API calls can be made at any time, and will always reflect the data held by the data owner. In many cases these datasets are updated regularly, allowing companies, and other stakeholders, to see the results of the decision model based on the most up to date data, without needing human intervention to: send new datasets, re-run analysis, write a report, and provide that report in a suitable format for the data-owner. Automating model updates at set schedules, or when data is updated, may be invaluable where data is updated regularly, as has been the case throughout the COVID-19 pandemic.

- Any model can be passed to the API, as long as the inputs and outputs to the model meet the requirements of the API. This means that multiple health economic models could be passed to the API, to be run using the data on the data-owner's server, and compared to account for structural uncertainty.

- In the (intentionally simple) example we give in this paper, it is assumed that the data owner provides readily estimated parameter values. In a real world use case, this does not need to be the case: the consultant may also send a survival model specification, for example, to estimate parameters from individual patient data, before the parameters are then passed to the health economic model.

- Specifying a model without access to and feedback from the actual data may come with its own challenges. However, if these hurdles are overcome, the benefit is that it enforces a thorough statistical analysis plan by default, in every use case. This helps to avoid biases introduced by stakeholder incentives.

However, the framework has a number of limitations:

- Firstly, the method is relatively complex, and requires a strong understanding of health economic modelling in R, API creation and hosting, RMarkdown or other automated reporting packages, and GitHub Actions. While we hope that this paper provides a useful resource to health economists seeking to utilise these methods, the bulk of the industry still operates in MS Excel[18]. Providing tuition to upskill health economists, or creating teams consisting of both health economists and data-scientists and software engineers may mediate this limitation somewhat. Groups like the R for HTA consortium has the potential to play a crucial role in upskilling the industry.

- There are still likely to be concerns about data security, even with the authentication procedures built in to the API functionality. Collaboration with experts in this field may mediate this significantly, since there is no fundamental reason why health data is any more sensitive, or vulnerable, than the plethora of other data (including banking data) that relies on APIs every day. It will be important to reassure companies that the use of APIs is likely to reduce, not increase the risk of data breaches, and that every interaction with the data can be logged.

- There is a risk that running the model remotely will result in the perception that the model is a 'black box'. The use of user-interfaces (such as those increasingly being created in *shiny*) to interrogate the model, as well as the increased transparency associated with being able to share code on sites such as GitHub, should reassure stakeholders that this framework is more transparent than the existing spreadsheet based solutions[19].

- R is single threaded, and therefore will only work on one task at a time. This can make it slow when lots of requests are made simultaneously, which may occur if a model takes a long time to run. Most hosting platforms (including RSConnect, where we have deployed this example) solve this problem by creating multiple R processes, which work in isolation. The PRISM solution outlined by Abidi *et al.*[2] uses OpenCPU which works in the same way. This is fine for the example model we provide, since our model doesn't store any information required by other users, who would be working on another R process, during the session. Further information can be found on the plumber guide here: https://www.rplumber.io/articles/hosting.html.

- Often, when building a model, it is helpful to have the underlying data to be able to investigate the data, often through the generation of descriptive statistics. The process of sharing pseudo-data enables modellers to ensure that the models they create conform to the structure of the data input. However, the modeller still needs to be able to write code that is versatile enough to cope with data with unknown distributions ranges and number of observations. This is easily solved, again by improved training and the use of standard packages such as *hesim* and *heemod*[20,21].

The recent working paper by Adibi *et al.*[2] has provided a similar call to action, extolling the virtues of the API for decision modelling, and showing how APIs can be used to shift much of the computational burden away from those querying models, making models more accessible. However, there are several limitations to this innovative paper. Firstly, while the authors outline a framework for making models more transparent and accessible, and describe how they have done this for a number of models using the PRISM server, they do not provide instruction on how to replicate this process. Additionally, while the authors state that "A practical model accessibility platform should be able to protect confidential information such as patient data and confidential pricing" (p6), the framework as described would require companies to give the owners of the model accessibility platform access to their confidential data, or else host the model accessibility platform themselves.

This paper has attempted to address some of these limitations, providing open source code for the creation and deployment of an API with an accompanying automated health economic evaluation update framework. It also provides open source code on two new pieces of additional functionality not previously described elsewhere; firstly it demonstrates how companies can host APIs themselves to negate the need to share data with subject experts, and secondly it demonstrates how model updates can be automated with scheduled workflows run on remote servers.

Conclusions

This example framework, with accompanying open source code base, demonstrates that it is possible, within a HEOR setting, to separate a health economic model from the data, and automate the main steps of the analysis pipeline. We believe this is the first application of this procedure in the HEOR context, and is certainly the first example to be made open source for the benefit of the wider community. We hope that this framework will improve the transparency of health economic models, reduce the cost and administrative burden of updating models, and increase the speed at which updates can occur.

Data availability

Underlying data

Zenodo: Parameter distributions. https://doi.org/10.5281/zenodo.6727629[14].

This project contains the following underlying data:

- parameter_distributions.csv (example dataset for modification. This dataset also sits within the server, with some of the rows marked as non-editable; these are characterised as 'sensitive' throughout the manuscript. This dataset is edited in 'Code Chunk 2' to test the API).

Software availability

Source code available from: https://github.com/RobertASmithBresMed/plumberHE.

Archived source code at time of publication: https://doi.org/10.5281/zenodo.6556888[6]

License: MIT

Acknowledgements

We would like to thank the participants at R-HTA Oxford, Richard Birnie (Lumanity) and Dawn Lee (Lumanity) for feedback on the original concept and manuscript. All errors are the fault of the authors.

References

1. Williamson EJ, Walker AJ, Bhaskaran K, *et al.*: **Factors associated with covid-19-related death using opensafely.** *Nature.* 2020; **584**(7821): 430–436.
PubMed Abstract | Publisher Full Text | Free Full Text
2. Adibi A, Harvard S, Sadatsafavi M: **Programmable interface for statistical & simulation models (prism): Towards greater accessibility of clinical and healthcare decision models.** *arXiv preprint arXiv: 2202.08358.* 2022.
Reference Source
3. Xie Y, Dervieux C, Riederer E: **R Markdown Cookbook.** Chapman and Hall/CRC, Boca Raton, Florida, 2020.
Reference Source
4. Chang W, Cheng J, Allaire JJ, *et al.*: **shiny: Web Application Framework for R.** 2021.
Reference Source
5. Smith R, Schneider P: **Making health economic models Shiny: A tutorial [version 2; peer review: 2 approved].** *Wellcome Open Res.* 2020; **5**: 69.
PubMed Abstract | Publisher Full Text | Free Full Text
6. Smith R, Schneider P, Mohammed W: **Robertasmithbresmed/plumberhe: R-hta june 2022.** software, 2022.
http://www.doi.org/10.5281/zenodo.6556888
7. Alarid-Escudero F, Krijkamp EM, Enns EA, *et al.*: **A tutorial on time-dependent cohort state-transition models in r using a cost-effectiveness analysis example.** *arXiv preprint arXiv: 2108.13552.* 2021.
Publisher Full Text
8. Alarid-Escudero F, Krijkamp E: **DARTHgit/cohort-modeling-tutorial-intro: Second Zenodo release.** 2022.
Publisher Full Text
9. Ooms J: **The jsonlite package: A practical and consistent mapping between json data and r objects.** *arXiv:1403.2805 [stat.CO].* 2014.
Reference Source
10. Schloerke B, Allen J: **plumber: An API Generator for R package**

version 1.1.0, 2021.
Reference Source

11. Wickham H, Danenberg P, Csárdi G, *et al.*: **roxygen2: In-line documentation for r.** *R package version.* 2020; **7**(1).
12. Wickham H: **httr: Tools for Working with URLs and HTTP.** R package version 1.4.2. 2020.
Reference Source
13. Chandrasekara C, Herath P: **Introduction to github actions.** In: *Hands-on GitHub Actions.* Springer, 2021; 1–8.
Publisher Full Text
14. Smith R: **Parameter distributions.** dataset. 2022.
http://www.doi.org/10.5281/zenodo.6727629
15. R Core Team: **R: A Language and Environment for Statistical Computing.** R Foundation for Statistical Computing, Vienna, Austria, 2020.
Reference Source
16. McCabe C, Paulden M, Awotwe I, *et al.*: **One-Way Sensitivity Analysis for Probabilistic Cost-Effectiveness Analysis: Conditional Expected Incremental Net Benefit.** *Pharmacoeconomics.* 2020; **38**(2): 135–141.
PubMed Abstract | Publisher Full Text | Free Full Text
17. Brennan A, Kharroubi S, O'hagan A, *et al.*: **Calculating partial expected value of perfect information via monte carlo sampling algorithms.** *Med Decis Making.* 2007; **27**(4): 448–470.
PubMed Abstract | Publisher Full Text
18. 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**(5): 575–579.
PubMed Abstract | Publisher Full Text
19. Pouwels XGLV, Sampson CJ, Arnold RJG, *et al.*: **Opportunities and Barriers to the Development and Use of Open Source Health Economic Models: A Survey.** *Value Health.* 2022; **25**(4): 473–479.
PubMed Abstract | Publisher Full Text
20. Incerti D, Jansen JP: **hesim: Health economic simulation modeling and decision analysis.** *arXiv preprint arXiv:2102.09437.* 2021.
Publisher Full Text
21. Filipovic-Pierucci A, Zarca K, Durand-Zaleski I: **Markov models for health economic evaluation modelling in r with the heemod package.** *Value Health.* 2016; **19**(7): A369.
Publisher Full Text

Open Peer Review

Current Peer Review Status: ✔ ✔

Version 2

Reviewer Report 24 October 2022

https://doi.org/10.21956/wellcomeopenres.20423.r52750

✔ **Mohsen Sadatsafavi**
Faculty of Pharmaceutical Sciences, University of British Columbia, Vancouver, BC, Canada

Thanks for revising the paper so diligently. I enjoyed reading this manuscript and again find the contribution very relevant to the health economics modeling community and related stakeholders. I have very minor/discretionary suggestions:

- The new Figure 1 is very informative (albeit, I suggest using consistent terms [e.g., data-owner under 'Company'] with the text).
- Under 'Deploying API': Perhaps make it clear that if desired the data-owner can host the API on their own company servers (so really keeping the data local) and do not have to use remote cloud-based solutions.
- The paper can benefit from discussing (somewhere in the Discussion section) the security risk the data-owner is exposed to when letting a consultant upload the code on their server. There are indeed solutions to that (e.g., sandboxing strategies specifically provided by Linux servers, virtualization, etc).

Competing Interests: I am the lead of the PRISM project and the senior author of its paper (https://arxiv.org/abs/2202.08358) which is somewhat related to this technology. The authors appropriately cite this paper and make it clear that their proposed platform is significantly different in major ways.

Reviewer Expertise: Economic Evaluation; Epidemiology; Medical Decision Making

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Version 1

Reviewer Report 12 September 2022

https://doi.org/10.21956/wellcomeopenres.19871.r51659

Devin Incerti
Head of Data Science, EntityRisk Inc., Princeton, NJ, USA

This is a useful paper that has the potential to enhance the quality of the data used by health economic models.

I have several comments related to the discussion section:

1. The authors should clarify how they envision data being used. In this example, parameter estimates are shared by the data-owner. However, this assumes that the data-owner is capable of estimating the parameters.

2. A more useful workflow might be one in which data-owners share the raw underlying data (e.g., clinical trial data) and the consultant parameterizes the health economic model using that data. The authors rightfully note that "Often, when building a model, it is helpful to have the underlying data to be able to investigate the data, often through the generation of descriptive statistics." However, they understate the significance of this limitation:

    - Creating a fully automated workflow to parameterize a model without access to the underlying data is difficult. For instance: it is often helpful to have access to the actual data to (i) identify outliers and possible coding errors, (ii) recode predictors (e.g., by combining categories with small numbers of observations), and (iii) assess the feasibility of a model specification (e.g., ordered logistic regression vs. multinomial logistic regression).

    - The authors argue that "pseudo data" can help overcome this problem. While mostly true, creating synthetic data is no trivial task and should be noted as a challenge. I would argue that parameterization without access to the actual data will be easier if the pseudo data is more realistic.

3. While the points raised in #2 above are a limitation of the framework, they are also a potential strength. As noted by the first reviewer, the use of pseudo data may reduce perceived bias; e.g., by requiring statistical analyses to be pre-specified and fully automated. Health economic models (and the statistical models that parameterize them) can be set up as "pipelines" (like in machine learning) capable of being run on any dataset and blind to the outcomes.

4. Data-owners may be concerned that the transfer of sensitive data may not be secure. The authors should comment on any potential security concerns.

5. The current code is open-source. The authors may want to comment on closed-source approaches since many consultants will be hesitant to make code publicly available.

I also have some minor comments about the code and user interface:

- The body of a for loop should be indented (e.g., starting on line 40 in code chunk 1) to improve readability.

- *renv* should be used to manage package dependencies and enhance the reproducibility of the paper. I needed to install *readr* before I could successfully run code chunk 2.

- The user interface does not currently provide error messages if invalid model inputs are used (e.g., if a negative shape parameter is used for utility). This is issue #2 in the GitHub repository, but it has not yet been addressed.

Is the rationale for developing the new method (or application) clearly explained?

Yes

Is the description of the method technically sound?

Yes

Are sufficient details provided to allow replication of the method development and its use by others?

Yes

If any results are presented, are all the source data underlying the results available to ensure full reproducibility?

Yes

Are the conclusions about the method and its performance adequately supported by the findings presented in the article?

Yes

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: Health economics, Biostatistics

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Author Response 26 Sep 2022

Robert Smith, University of Sheffield, Sheffield, UK

This is a useful paper that has the potential to enhance the quality of the data used by health economic models.

We are glad you think it is useful and thanks for the helpful comments.

The authors should clarify how they envision data being used. In this example, parameter estimates are shared by the data-owner. However, this assumes that the data-owner is capable of estimating the parameters.

We agree, this was kept intentionally vague because different use-cases may require sharing at different stages in the process (potentially companies may wish to do some early analysis 'in-house'). However, we have added a section in the discussion stating that:
"In the (intentionally simple) example we give in this paper, it is assumed that the data owner provides readily estimated parameter values. In a real world use case, this does not need to be the case: the consultant may also send a survival model specification, for example, to estimate parameters from individual patient data, before the parameters are then passed to the health economic model."

A more useful workflow might be one in which data-owners share the raw underlying data (e.g., clinical trial data) and the consultant parameterizes the health economic model using that data. The authors rightfully note that "Often, when building a model, it is helpful to have the underlying data to be able to investigate the data, often through the generation of descriptive statistics." However, they understate the significance of this limitation:

1. **Creating a fully automated workflow to parameterize a model without access to the underlying data is difficult. For instance: it is often helpful to have access to the actual data to (i) identify outliers and possible coding errors, (ii) recode predictors (e.g., by combining categories with small numbers of observations), and (iii) assess the feasibility of a model specification (e.g., ordered logistic regression vs. multinomial logistic regression).**
2. **The authors argue that "pseudo data" can help overcome this problem. While mostly true, creating synthetic data is no trivial task and should be noted as a challenge. I would argue that parameterization without access to the actual data will be easier if the pseudo data is more realistic.;**

We agree that building the model without access to the underlying data is very difficult and may not be feasible in many cases - especially where the model is more complex (for example deviating from a typical oncology model). However, lots of use cases remain, including model updates for new data cuts and, increasingly, updates to RWE from databases. It's still really early days, but we could gradually approach 'livingHTA' using hospital data to inform commissioning in a continuous feedback process.

While the points raised in #2 above are a limitation of the framework, they are also a potential strength. As noted by the first reviewer, the use of pseudo data may reduce perceived bias; e.g., by requiring statistical analyses to be pre-specified and fully automated. Health economic models (and the statistical models that parameterize them) can be set up as "pipelines" (like in machine learning) capable of being run on any dataset and blind to the outcomes.

Completely agree, we have added a section in the text in the discussion which reads:
"Specifying a model without access to and feedback from the actual data may come with its own challenges. However, if these hurdles are overcome, the benefit is that it enforces a thorough statistical analysis plan by default, in every use case. This helps to avoid biases

introduced by stakeholder incentives."

Data-owners may be concerned that the transfer of sensitive data may not be secure. The authors should comment on any potential security concerns.

We agree, and expect this to be a significant hurdle. We have a section in the text that states that:
"There are still likely to be concerns about data security, even with the authentication procedures built in to the API functionality. Collaboration with experts in this field may mediate this significantly, since there is no fundamental reason why health data is any more sensitive, or vulnerable, than the plethora of other data (including banking data) that relies on APIs every day. It will be important to reassure companies that the use of APIs is likely to reduce, not increase the risk of data breaches, and that every interaction with the data can be logged."

The current code is open-source. The authors may want to comment on closed-source approaches since many consultants will be hesitant to make code publicly available.

Yes, I suspect many people will want to keep their code private. This is kind of contrary to our intentions (which is to create a situation where it is easier to share the model) so we have added a section in the text that states that:
"However, if it is necessary to restrict access to model code, it is possible for the API to be passed 'private' source code. As keen proponents of open source modeling that was determined to be beyond the scope of this publication."

The body of a for loop should be indented (e.g., starting on line 40 in code chunk 1) to improve readability.

Thanks, we have asked the editorial team to address this in the paper.
This has also been updated in the code:
https://github.com/RobertASmithBresMed/plumberHE/commit/9f567d5c6b79b3d20865a4860f33e2de4151b

renv should be used to manage package dependencies and enhance the reproducibility of the paper. I needed to install readr before I could successfully run code chunk 2.

Thanks, we should have done this before - the lock file can now be found here:
https://github.com/RobertASmithBresMed/plumberHE/blob/main/renv.lock

The user interface does not currently provide error messages if invalid model inputs are used (e.g., if a negative shape parameter is used for utility). This is issue #2 in the GitHub repository, but it has not yet been addressed.

We have created an issue: https://github.com/RobertASmithBresMed/plumberHE/issues/29 and are in the process of reviewing a solution to this problem. Many thanks for this suggestion.

Competing Interests: No competing interests were disclosed.

Reviewer Report 08 August 2022

https://doi.org/10.21956/wellcomeopenres.19871.r51661

?

Mohsen Sadatsafavi
Faculty of Pharmaceutical Sciences, University of British Columbia, Vancouver, BC, Canada

Smith and colleagues introduce an API framework for developing and implementing health economics models without necessarily sharing the underlying data with model developers. This is facilitated via an Application Programming Interface. The main utility of this framework is separating data from the model. The setup of the platform is that the client (say a pharma company) is commissioning the creation of a health economics model via a consultancy firm. The company is in possession of sensitive and not-sharable data that are used to develop the model. To make things work, the company creates a Web server that securely hosts the data. It optionally provides synthetic data to the consultant to help build the model. Once the model is built, it is sent to the Web server where it sits next to the sensitive data to complete what is needed to run the economic evaluation. It then sends the output back to the consultant and a report is generated. This process is further automated such that the entire process is activated at set intervals or after a change in the model or the underlying data. This pipeline uses several widely adopted standards or services such as JSON for data transfer, Github for haring the model code and automation, RMarkdown, and so on. I really enjoyed reading this paper and found it an important step in the right direction.

I have several suggestions for improving the exposition.

Major comments:

1. The authors should be clearer with their terminology throughout as well as which code is running where. It took this reader several reads to finally (apparently) understand. Suggest making it clear that a client has sensitive data and tasks a consultant for creating a model. Make it clear that the first code chunk runs on the client's server (next to data) while the second is run on the consultant's side (if I understood things correctly).

2. "*When a 'client' application wants to access data, it initiates an API call (request) via a web-server, to retrieve the data. If this request is deemed valid, the API makes a call to an external program/server, the server sends a response to the API with the data, and the API transfers the data to the 'client' application*." I found these sentences on top of Page 5 quite confusing. Is not sharing sensitive data whilst being able to run the model not the main point of this platform?

3. The only input that the run_model accepts seems to be the PA dataset. Are there flexibilities around other sets of parameters? What if someone wants to change the time horizon of the model? Similarly, what the investigators consider as the output of the model is restrictive (only costs and QALYs). A health economic model can have other payoffs (mortality, disease incidence) and to this reviewer, the presence of clinical or other payoffs is more norm than an exception. Can they make comments on to what extent these parts can be made more flexible?

4. Suggest making it clear that by sharing the model and keeping the data private, the requirement now is for the model to be open-source. This in itself is sharing IP, and the authors can make it clear that their innovation is that the model is shared not the data, and perhaps claim that oftentimes sharing data is more restrictive.

5. Please make it clear that plumber is a state-based environment and as such, it connects all the consultants to the same R session. The advantage of this approach is that it is 'live' such that subsequent function calls can be made to interact with the model. The drawback is that multiple consultants might overwrite each other's sessions and also keeping R alive, especially with complex models can be problematic for the server. The authors can make a distinction with Adibi *et al.*'s PRISM which uses OpenCPU.

Minor comments:

1. Is adhering to the DARTH naming convention absolutely necessary? While using such standards is generally good, is this platform not more generic in nature (e.g., the company and the consultant agreeing on a function call)?

2. In the case study for which code chunks are provided, what data are considered sensitive for illustrative purposes?

3. Why the dataframe representing the probabilistic input should have distributions with two parameters? This will preclude the use of some distributions like generalized gamma that have three or more parameters.

4. One advantage of this platform can be that by making the modeler team use synthetic data, the potential for stakeholder bias is also minimized.

5. Introduction: "*The development of economic models generally involves the transfer of sensitive data (e.g. individual patient or price data) between parties*". I think '*generally*' is too strong here. Perhaps use 'at times'? Many health economics modeling efforts are based on the literature or publicly available data. Perhaps the authors can restrict this statement to models developed by the industry for their new health technologies for which there are often sensitive data.

6. I could not understand the point of overwrite_parameter_value() function. Can the authors clarify? Similarly, in the code chunk #2, line 52: "*here we expect a single dataframe with 6 columns and 1000 rows*". It is not obvious to me how we expect this to be 1000 exactly. Should this not be decided by the dimension of the input data frame?

7. Page 7: "*for this, but once you have one it is possible*"; '*you*' is a bit colloquial for a science paper. Similarly, referring to Adibi's framework as '*brilliant*' is a bit colloquial.

Is the rationale for developing the new method (or application) clearly explained?
Yes

Is the description of the method technically sound?
Yes

Are sufficient details provided to allow replication of the method development and its use by others?
No

If any results are presented, are all the source data underlying the results available to ensure full reproducibility?
Yes

Are the conclusions about the method and its performance adequately supported by the findings presented in the article?
Yes

Competing Interests: I am the lead of the PRISM project and the senior author of its paper (https://arxiv.org/abs/2202.08358) which is somewhat related to this technology. The authors appropriately cite this paper and make it clear that their proposed platform is significantly different in major ways.

Reviewer Expertise: Economic Evaluation; Epidemiology; Medical Decision Making

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard, however I have significant reservations, as outlined above.

Author Response 26 Sep 2022

Robert Smith, University of Sheffield, Sheffield, UK

Smith and colleagues introduce an API framework for developing and implementing health economics models without necessarily sharing the underlying data with model developers. This is facilitated via an Application Programming Interface. The main utility of this framework is separating data from the model. The setup of the platform is that the client (say a pharma company) is commissioning the creation of a health economics model via a consultancy firm. The company is in possession of sensitive and not-sharable data that are used to develop the model. To make things work, the company creates a Web server that securely hosts the data. It optionally provides synthetic data to the consultant to help build the model. Once the model is built, it is sent to the Web server where it sits next to the sensitive data to complete what is needed to run the economic evaluation. It then sends the output back to the

consultant and a report is generated. This process is further automated such that the entire process is activated at set intervals or after a change in the model or the underlying data. This pipeline uses several widely adopted standards or services such as JSON for data transfer, Github for having the model code and automation, RMarkdown, and so on. I really enjoyed reading this paper and found it an important step in the right direction.

Thank you for taking the time to review this paper and for the very constructive comments which we think have improved the quality of the paper. We are glad that you enjoyed reading it and think that it is taking an important step in the right direction.

The authors should be clearer with their terminology throughout as well as which code is running where. It took this reader several reads to finally (apparently) understand. Suggest making it clear that a client has sensitive data and tasks a consultant for creating a model. Make it clear that the first code chunk runs on the client's server (next to data) while the second is run on the consultant's side (if I understood things correctly).

We agree. We have done several things to try and address this issue - we believe this makes the paper clearer:

1. We have changed references to 'client' in the main body of the text to 'data-owner' to reduce confusion with the term client in the context of APIs.

2. We have added a section in the methods section of the manuscript which reads as below. This hopefully gives more context. making it clear that the data-owner has the data and is tasking the consultant to build the model without access to the data.

3. Third, we have added additional information to the code chunk headings. For "Code chunk 1 - Generating the API" we have added, "this code is run on the data-owner's server". For "Code chunk 2 - Query the API, retrieve model results and generate report" we have added, "this code is run by the consultant".

In the methods section:

"A common problem in health technology assessment is a situation in which a data-owner (e.g. a company in the pharmaceutical industry) holds sensitive data, but requires the services of a consultant to conduct health economic modelling. For example the data-owner may have interim clinical trial data at the patient level, and may need an external health economist (the consultant) to build a state transition model to determine the cost-effectiveness of the treatment. Currently the data-owner may be required to send the consultant cuts of data as the trial progresses (e.g. 12 month, 18 month, 24 month). This is burdensome and results in multiple iterations of sharing sensitive data. We propose a solution that does not require the sharing of data between parties and allows for automated updates to the analysis as data is updated. This automated analysis and reporting pipeline for health economic modelling consists of three parts:"

"When a 'client' application wants to access data, it initiates an API call (request) via a web-server, to retrieve the data. If this request is deemed valid, the API makes a call

to an external program/server, the server sends a response to the API with the data, and the API transfers the data to the 'client' application." I found these sentences on top of Page 5 quite confusing. Is not sharing sensitive data whilst being able to run the model not the main point of this platform?

Thank you, we agree that the description of the process for an API is too general and not specific to this use case. We have changed this paragraph to read as below, with the hope that this is clearer:

"An application programming interface is a set of rules, in the form of code, that allow different computers to interact with one another in real time. Whereas user-interfaces such as those generated by the R package shiny allow humans to interact with data, APIs are designed to enable computers to interact with each other in a consistent way (for example to exchange data or access algorithms)4.

When a request is sent by one application (an API call) via a web server to another application, it is evaluated and if appropriate, an action is triggered and/or a response is sent. An API call can contain anything, ranging from data to a model, or just a single GET request (for example a web browser sends a request when trying to access a web page). If necessary, the API transfers the response of the API request to the client application (the application making the request). In a sense, the API is the broker (or middle-man) between two systems.

In the example described in this paper, the consultant is going to send an API request to the data owner's computer. The request will contain some data (non-sensitive model inputs) and the health economic model, which is to be run on the data owner's computer using the sensitive data available there. The API runs the health economic model and responds to the request with the output of the model."

The only input that the run_model accepts seems to be the PA dataset. Are there flexibilities around other sets of parameters? What if someone wants to change the time horizon of the model? Similarly, what the investigators consider as the output of the model is restrictive (only costs and QALYs). A health economic model can have other payoffs (mortality, disease incidence) and to this reviewer, the presence of clinical or other payoffs is more norm than an exception. Can they make comments on to what extent these parts can be made more flexible?

We have added the following paragraph to the paper.
Also, see response to minor comment #3:
"The function described is designed as a simple reproducible example. The proposed method is in principle flexible to any inputs, model structure, and outputs."

Suggest making it clear that by sharing the model and keeping the data private, the requirement now is for the model to be open-source. This in itself is sharing IP, and the authors can make it clear that their innovation is that the model is shared not the data, and perhaps claim that oftentimes sharing data is more restrictive.

We have adapted the introduction section to read:

"We therefore see the primary contribution of this paper as being the development of a system in which the health economic model and the data are two separate entities, and the health economic model is sent to the data rather than the other way around. By working in this way, it is theoretically easier to share the model without the sensitive data on which it is run, although making the model open source is not a requirement".

Please make it clear that plumber is a state-based environment and as such, it connects all the consultants to the same R session. The advantage of this approach is that it is 'live' such that subsequent function calls can be made to interact with the model. The drawback is that multiple consultants might overwrite each other's sessions and also keeping R alive, especially with complex models can be problematic for the server. The authors can make a distinction with Adibi et al.'s PRISM which uses OpenCPU.

We agree with your comment, but think it is unlikely to be a problem given that most platforms create multiple R processes which I believe gives a similar result to PRISM. For example, a plumber-powered API could be containerised and deployed on Google Cloud Run, which scales the resources (number of API containers) based on demand (https://w-mohammed.github.io/posts/deploying-a-plumber-API-on-Google-Cloud/). It is also maybe a very specific technical issue that can be overcome in isolation to the wider method and concept.

However, the comment is really interesting, and actually, I wonder if the bigger limitation is actually that this means that different R processes are working in isolation, so model runs will not interact, which would be important for things like expert elicitation but can easily be overcome using external databases.

I hope that the addition of the paragraph below, with reference to PRISM, to the discussion helps to clear some of this up:

"R is single threaded, and therefore will only work on one task at a time. This can make it slow when lots of requests are made simultaneously, which may occur if a model takes a long time to run. Most hosting platforms (including RSConnect, where we have deployed this example) solve this problem by creating multiple R processes, which work in isolation. The PRISM solution outlined by Abidi et al. [2] uses OpenCPU which works in much the same way. This is fine for the example model we provide, since our model doesn't store any information required by other users, who would be working on another R process, during the session. Further information can be found on the plumber guide here: https://www.rplumber.io/articles/hosting.html."

Minor

Is adhering to the DARTH naming convention absolutely necessary? While using such standards is generally good, is this platform not more generic in nature (e.g., the company and the consultant agreeing on a function call)?

The DARTH sick-sicker model has been used as an example because it is a commonly used teaching tool in Health Economics and there are published papers describing the code, so if interested readers want to dig into the code further they can. However, you are completely correct that this process would work for any model. We actually see one of the benefits of this technology being that several consultancies could submit their models to the data - and the results could be compared to better understand the structural uncertainty associated with the models - all without any of the consultancies accessing the underlying data.

In the case study for which code chunks are provided, what data are considered sensitive for illustrative purposes?

The underlying data can be found here: plumberHE/parameter_distributions.csv at main · RobertASmithBresMed/plumberHE (github.com)

The sensitive rows of the dataframe are flagged by the column 'editable' being FALSE. To give more context around this we have added to the section 'The economic model', by adding to the end of the paragraph reading "psa_inputs, which is a data-frame containing Probabilistic Sensitivity Analysis parameter inputs for the model variables that are allowed to vary." the sentence "Additional, sensitive parameters (including treatment costs and hazard ratio for treatment B) are not allowed to be varied by the API request and will be informed by the values held by the data-owner."

Why the dataframe representing the probabilistic input should have distributions with two parameters? This will preclude the use of some distributions like generalized gamma that have three or more parameters.

We agree. However, we sought to make this example as simple as possible. Models constructed using this framework are completely flexible to adapt to any data format or model structure. To reflect this we have added to the bottom of 'The Economic Model' section the paragraph:
"The function described is designed as a simple reproducible example. The proposed method is in principle flexible to any inputs, model structure, and outputs."

One advantage of this platform can be that by making the modeler team use synthetic data, the potential for stakeholder bias is also minimized.

We agree. The following paragraph was added to the paper:
"Specifying a model without access to and feedback from the actual data may come with its own challenges. However, if these hurdles are overcome, the benefit is that it enforces a thorough statistical analysis plan by default, in every use case. This helps to avoid biases introduced by stakeholder incentives."

Introduction: "The development of economic models generally involves the transfer of sensitive data (e.g. individual patient or price data) between parties". I think 'generally' is too strong here. Perhaps use 'at times'? Many health economics modeling efforts are based on the literature or publicly available data. Perhaps the authors can

restrict this statement to models developed by the industry for their new health technologies for which there are often sensitive data.

We have changed the sentence to read:
"The development of economic models sometimes involves the transfer of sensitive data (e.g. individual patient or price data) between parties".

I could not understand the point of overwrite_parameter_value() function. Can the authors clarify? Similarly, in the code chunk #2, line 52: "here we expect a single dataframe with 6 columns and 1000 rows". It is not obvious to me how we expect this to be 1000 exactly. Should this not be decided by the dimension of the input data frame?

The 'overwrite_parameter_value' function takes the data-frame passed by the consultant to the data-owner in code Chunk 2-Line-22, and combines it with the default parameters stored on the company server (it doesn't overwrite those values, instead creating a new data-frame). This allows the consultant to change any parameter for which they have permission (see point 2 above with the 'editable' column).

In code chunk #1 line 52 the expectation is for a data-frame with 6 columns and 1000 rows. In this simple example, this would have had to be pre-agreed between the consultant and the data-owner since it is hard-coded into the model code, however, the number of PSA runs could be provided as an input to the API if deemed necessary.

Page 7: "for this, but once you have one it is possible"; 'you' is a bit colloquial for a science paper. Similarly, referring to Adibi's framework as 'brilliant' is a bit colloquial.

We agree, we have changed page 7 to read:
"An account is required for this, but provides the benefit of being able to deploy the API directly from the Rstudio integrated development environment."

We have also changed the description of the Adibi framework from "brilliant" to innovative".

Thank you again for taking the time to review this paper. We think that these responses and the edits made have improved the clarity of the paper for others.

Kind regards,

Rob

Competing Interests: No competing interests were disclosed.