



Towards sharing tools and artefacts for reusable simulations in healthcare

Thomas Monks, Alison Harper & Navonil Mustafee

To cite this article: Thomas Monks, Alison Harper & Navonil Mustafee (02 May 2024): Towards sharing tools and artefacts for reusable simulations in healthcare, *Journal of Simulation*, DOI: [10.1080/17477778.2024.2347882](https://doi.org/10.1080/17477778.2024.2347882)

To link to this article: <https://doi.org/10.1080/17477778.2024.2347882>



© 2024 The Author(s). Published by Informa UK Limited, trading as Taylor & Francis Group.



Published online: 02 May 2024.



Submit your article to this journal



Article views: 1595



View related articles



View Crossmark data



Citing articles: 8 View citing articles

Towards sharing tools and artefacts for reusable simulations in healthcare

Thomas Monks ^a, Alison Harper  ^{b,c} and Navonil Mustafee  ^b

^aUniversity of Exeter Medical School, Exeter, UK; ^bUniversity of Exeter Business School, Exeter, UK; ^cNIHR Applied Research Collaboration South West Peninsula, University of Exeter, Exeter, UK

ABSTRACT

Discrete-event simulation (DES) is a widely used computational method in health services and health economic studies. Despite increasing recognition of the advantages of open, reusable DES models for both healthcare practitioners and simulation researchers, in practice very few authors share their model code alongside a published paper. In the context of Free and Open Source Software (FOSS), this paper presents a pilot framework called STARS: Sharing Tools and Artefacts for Reusable Simulations to begin to address the challenges and leverage the opportunities of sharing DES models in healthcare. STARS aligns with existing guidelines and documentation, including reproducibility initiatives, and enables computer models to be shared with users of differing technical abilities. We demonstrate the feasibility and applicability of STARS with three applied DES examples using Python. Our framework supports the development of open, reusable DES models which can enable partner healthcare organisations to preview, validate, and use models. Academic research teams can benefit from knowledge exchange, enhanced recognition and scrutiny of their work, and long-term archiving of models.

ARTICLE HISTORY

Received 6 November 2023
Accepted 19 April 2024

KEYWORDS

Discrete-event simulation;
healthcare; reusable models;
open science

1. Introduction

The health and medical research community is increasingly recognising the value of open science, and transparent research practices for science and society at large (Celi et al., 2019; Goldacre et al., 2019; Zečević et al., 2021). Open science can be defined as a collection of methods, approaches, and a movement to make the content and process of producing research and evidence transparent and accessible to others (Munafò et al., 2017). In healthcare modelling and simulation (M&S), discrete-event simulation (DES) is the most widely used simulation method (Philip et al., 2022; Roy et al., 2021; Salleh et al., 2017; Salmon et al., 2018). Systematic scoping reviews of the simulation health literature demonstrate that DES has been applied widely to benefit patients and the public (Forbus & Berleant, 2022; Roy et al., 2021; Vázquez-Serrano et al., 2021). The majority of healthcare DES studies do not openly share their computer simulation models with their publication or adhere to open science best practice (Monks & Harper, 2023a).

Open sharing of healthcare DES computer models is associated with simulation model reuse (Robinson, 2005). The Open Modelling Foundation (OMF, 2023)¹ define reusability as “implicitly includ[ing] usability and focus[ing] on the ability of humans and machines to execute, inspect, and understand the software so that it can be modified, built upon, or incorporated

into other software”. Making models that are generic and reusable can, therefore, reduce duplication of effort, by making available a usable computer model that has been tested in one healthcare organisation, to a similar problem in a different organisation without the need to rebuild (Crowe et al., 2023; Fletcher & Worthington, 2009).

1.1. Study motivation

Our motivation is to support the sharing of DES models in health research to enable increased reuse by practitioners and researchers, opportunities for early career researchers to learn from models, and the transfer of reusable practical DES models into the hands of clinicians, analysts, and other professionals in health. This paper aims particularly to support early career researchers to navigate the work associated with open science and open models as other studies have found that there are potentially large career benefits, including increased citations, new collaborations, and reuse of outputs (C. Allen & Mehler, 2019; McKiernan et al., 2016).

Our previous work has shown that researchers are beginning to share healthcare DES computer models with their publications; although this is still a small minority of the overall literature (Monks & Harper, 2023a). We found that, in most cases, the approach to sharing computer models did not meet the simple best

practice criteria we specified based on the Turing Way (The Turing Way Community, 2022), high level open scholarship recommendations for M&S (Taylor et al., 2017), and OMF's open science checklists (OMF, 2023). For example, in the majority of cases authors inadvertently retained exclusive copyright of the models and did not grant others the rights to reuse them in their research. Although not essential to the practice of sharing models, very few authors paid attention to the *accessibility* of their computer models. Accessibility is an important facet of open science, and could be enhanced via contemporary web technology to enable healthcare professionals or other researchers to use models without the requirement to install software (Dagkakis & Heavey, 2016; Harper, Monks, & Manzi, 2023; R. Smith & Schneider, 2020; Wojciechowski et al., 2015).

Barriers to open science and open models are now well documented (C. Allen & Mehler, 2019; Gomes et al., 2022; Zečević et al., 2021), for example, time and knowledge, and the opportunity now exists to test if some of these challenges in DES can be tackled using contemporary technology including sharing computer models openly using web-based tools.

1.2. Study context: Free and Open Source Software

Our study focuses on DES computer models developed in Free and Open-Source Software (FOSS) languages, such as Python, R, or Julia. Within the FOSS literature *free* is defined as *freedom*, and not price, or cost of the software. FOSS grants the rights for users to adapt and distribute copies of software however they choose using an appropriate open licencing model. For example, a research team might develop a DES model of a hospital ward using the R DES package *simmer*. The team might choose to publish a licenced version of their model alongside a journal article. Another research team then has the freedom to inspect, run, reuse, or adapt the DES model in their own work, e.g., applied to another similar ward, or as part of a whole hospital model, and republish the adapted model if they desire.

In recent years FOSS simulation packages, particularly models coded in R or Python, have become more prevalent in the healthcare M&S literature and have been shown to be more likely to be shared alongside a journal article than models built in commercial software (Monks & Harper, 2023a). Although there is great potential for using FOSS software in healthcare DES, FOSS *does not guarantee a DES model will actually be accessible and reusable*. Indeed our review found that best practices for sharing these FOSS models are often not used. Consider, for example, a model coded in Python where the code is copy-pasted

into word processing software and published as an online journal article supplementary material. The model is indeed available, but there are many potential issues that such an approach could cause for reuse. The most likely is that this model will now no longer run without manual intervention. Formatting issues introduced via word processing software would require a user to manually edit the code. The user could never be fully confident that the model executes as the original authors intended as the code has been changed. Our framework aims to tackle these challenges, as well as others, by simplifying the sharing of healthcare DES models developed using FOSS software.

2. Objectives of the study

Our overarching research question asked *is it feasible to overcome existing barriers to open and reusable DES models in healthcare?* To answer this question we set the following four research objectives:

- (1) Review the model and code sharing literature to identify the opportunities and barriers to sharing DES models in health;
- (2) Design and propose a new framework for sharing FOSS DES models in health;
- (3) Test the feasibility of the framework for sharing Python models using existing FOSS technology;
- (4) Identify current limitations of the approach and future research direction.

We limit the feasibility test of objective (3) to Python computer models, as it is consistently ranked in the top four of Stack Overflow's most used programming languages (fourth in 2022) and top five most loved programming languages by developers. Python is also synonymous with modern data science and is used extensively in research and development of computational methods and applications.

The remainder of the paper is organised as follows. First, we review the aims of open science, and model and code sharing in simulation. The review examines rates of code sharing across healthcare computational research; in healthcare M&S; and the disincentives and challenges to code sharing. Second, we describe a pilot framework to address the challenges of sharing DES computer models in health. We have named the framework "STARS: Sharing Tools and Artefacts for Reusable Simulation". Following our description, we demonstrate the feasibility of STARS in three applied examples. Finally, we identify the strengths and current limitations of the pilot framework and additional work needed to increase uptake and utility.

3. Model and code sharing literature

3.1. Aims of open science

The aim of open science is to make scientific process and research outputs more widely accessible, promoting the transparency, credibility, and reuse of computational artefacts (Pouwels et al., 2022). Broader definitions emphasise a culture of openness, including the whole process of undertaking and sharing research (Munafò et al., 2017). This refers to a shift in how methods of research are conducted, alongside technology change which enables new workflows to increase efficiency, collaboration, and sharing of artefacts. The popularity of open science was initiated in part by a focus on scientific reproducibility, hence the term often encompasses principles of reproducibility (Foster & Deardorff, 2017; Munafò et al., 2017). The ability to reproduce or verify the results of a published study is enabled by open code and transparent methodologies, however both models developed using commercial software and those using FOSS can be reproducible, as can meticulously documented models in the absence of shared code. Hence, reproducibility is not the same as open science.

For M&S, Taylor et al. (2017) outlined some of the benefits of open science, including the potential to reduce the cost and effort of research duplication, increase the quality of research validation, enhance knowledge transfer, and improve trust in research and outputs. In healthcare, Goldacre et al. (2019) drew attention to the importance of code sharing for transparency, by providing an unambiguous record of analysis rather than the brief narrative description of complex analysis found in published papers. More widely, McKiernan et al. (2016) reviewed literature showing that researchers using open practices can gain more citations and visibility, potential collaborations, job opportunities and funding opportunities.

3.2. The practice of code sharing

As the principles and practice of open science are increasingly being adopted by researchers and institutions, a number of studies have found that code sharing practices, while gradually improving over time, remain suboptimal. A comprehensive review of 7500 agent-based models across 1500 journals found that only 11% of articles shared code, although the authors reported a slight upward trend (Janssen et al., 2020). The field of psychology fared better, with 44% of papers sharing computer code, although 70% of the scientific code was poorly annotated, and 52% was found to be incomplete (Kucharsky et al., 2020). In contrast, a meta-analysis of code and data sharing in medicine and health (examining a total of nearly 12,000 articles) estimated code sharing rates since 2016 to be less than 0.5%, with little change over

time (Hamilton et al., 2023). The practice of identifying code sharing from papers is laborious; across the biomedical literature, Sergiou et al. (2021) evaluated over 2.7 million open access articles using automated text extraction of transparency indicators, finding that 1% of papers had a code sharing statement. With few journals systematically capturing machine-readable meta-data, their algorithm has some weaknesses, but its automation significantly enhances analysis of the uptake of open science practice, with the goal of increasing the dissemination of these models to ultimately benefit academics and practitioners.

3.3. Code sharing in healthcare DES research

Given the potential benefits of open science, we conducted a systematic scoping review of model and code sharing practices in healthcare DES from 2019 to 2022 (Monks & Harper, 2023a). In this study we reviewed 564 papers, finding 47 (8.3%) which shared a model or code in some form; most of these (66%) used FOSS tools. The proportion of shared models increased from 4% in 2019 to 9% in 2022, indicating a rising interest in sharing.

The state-of-the-art practice for sharing model artefacts is an evolving field. Based on recent open science community-driven guidelines and standards, we identified a common set of practices in our previous work which can support the goals of open science for healthcare M&S studies. Contemporary sharing of computer model artefacts is best done through a digital open science repository that guarantees persistence using a Digital Object Identifier (DOI) (Lin et al., 2020). This can be used to cite the exact version of the model which generated published results. A related concept is the Open Researcher and Contributor Identifier (ORCID), a unique identifier for an individual researcher. A trusted archive will accommodate ORCIDs within the meta-data of a deposited artefact, providing an unambiguous permanent link back to the authors. Published models should also be accompanied by an open licence, detailing how others may use or adapt the artefact(s), as well as re-share modifications and credit authors. To maximise the chances that another user can execute a computer artefact, a model's dependencies and the software environment must be specified. Formal methods to manage dependencies range from package managers, such as "conda" or "renv", to containerisation (where a model, parameters, an operating system and dependencies are deployed via a container and software such as Docker) (Moreau et al., 2023), to Open Science Gateways that allow remote execution (Taylor et al., 2017). Additionally, execution of a computer model artefact should be guided by a clear set of instructions. At a minimum, this should include a README file that provides an overview of

the scope of the model, how to execute it, and how to vary parameters. Documentation can be enhanced with notebooks that combine code and explanation (Ayllón et al., 2021; The Turing Way Community, 2022), and reporting guidelines to improve explanation of models and data (Monks et al., 2019). Finally, to enhance the accessibility and maintainability of models, contemporary web technology can enable online execution, interaction, and experimentation (Dagkakis & Heavey, 2016; Harper, Monks, & Manzi, 2023; Monks & Harper, 2023b).

Table 1 summarises these best practices for model and code sharing. We refer readers to Monks and Harper (2023a) for full details. Many of the shared models we identified were not reusable by researchers or healthcare data analysts, for example less than half were licenced for reuse or provided simple instructions for running the model. Other researchers have found that reuse of code is primarily motivated by the existence and extent of good quality documentation describing the code (den Eynden et al., 2016).

3.4. Disincentives and challenges for researchers

Many researchers acknowledge the beneficial principles of open science and code sharing. For example, in a survey of health economists (Pouwels et al., 2022) ($n = 230$), most believed that open source models increase transparency (92%), efficiency (76%), reuse (86%), and credibility (75%). A survey of computational bioscientists ($n = 251$) found that 53% of researchers believe that code should be well-documented and openly shared (Samota & Davey, 2021). However, given the low rates of sharing across computational disciplines, a number of barriers have been explored in the open research literature. Evidence from surveys by the Wellcome Trust ($n = 583$) (den Eynden et al., 2016) and PLOS Computational Biology journal (Hrynaszkiewicz et al., 2021) ($n = 214$) found that lack of time was the most commonly cited reason for not sharing. Other common concerns are code quality, software and system dependencies, and protecting intellectual property.

Through a working group of the Society for Open, Reproducible, and Transparent Ecology and Evolution (SORTEE), Gomes et al. (2022) distilled a discussion

of the barriers to code sharing, categorised into *disincentives*, *reuse concerns*, and *knowledge barriers*. *Disincentives* include “scooping” research ideas, lack of time, and lack of career incentives. *Reuse concerns* include inappropriate use, rights and intellectual property ownership, sensitive content, and concerns about the transience of storage. *Knowledge barriers* include uncertainty about the process, manual or complex workflows, insecurity about code quality, large datasets, and not seeing any value in sharing. The authors emphasise that in many cases these perceived barriers can be easily overcome, and in other cases, the extra effort may be outweighed by the significant advantages.

3.4.1. Disincentives

The short- and long-term time commitments required to share models are frequently reported to be a disincentive for researchers (den Eynden et al., 2016; Gomes et al., 2022; Hrynaszkiewicz et al., 2021). Time costs include annotating and quality control of code, data and documentation. C. Allen and Mehler (2019) considered time to be the greatest challenge for conducting open science for early career researchers, requiring careful consideration to research and workflow design early in a project.

What is clear is that more complex open science methods require more effort than simpler efforts (Harper & Monks, 2023a; Harper, Monks, & Manzi, 2023; Monks & Harper, 2023a). For example, in a small trial, the journal *Nature Biotechnology* reported that it took a median of nine days for authors to set up an online executable version of their computational artefacts using the containerisation and compute services provided by Code Ocean (Nature Biotechnology, 2019). In contrast, simply depositing code or a model in a trusted digital archive such as Zenodo requires only the time to upload the data and effort to add meta-data such as ORCIDs.

3.4.2. Reuse concerns

A survey of members of the Professional Society for Health Economics and Outcomes Research (ISPOR) Open Source Models Special Interest Group (OSM SIG) by Pouwels et al. (2022) found concerns about regulating the use and preventing the misuse of open models. Concerns about inappropriate reuse of

Table 1. Best practice model sharing criteria, adapted from Monks and Harper (2023a).

Item	Description
Digital Object Identifier (DOI)	Does the model have a DOI and promise of persistence? Can it be cited?
Open Researcher and Contributor ID (ORCID)	Is the model linked to one or more of the authors via an ORCID?
License	Does the model have a recognised open license to control the use of code, liability and credit?
Dependency management	Has a formal tool been used to manage software dependencies for the model?
Documentation	Are potential users of the model supported with instructions to run and explanations of the models workings?
Support for running the model	Is the model accessible to users with different skills sets and backgrounds? For example, inclusion of a user friendly interface?

a model, intellectual property, and transient storage can be addressed relatively easily via documentation describing the purpose and scope of the model, instructions for use, and terms of reuse according to the open licence selected (Gomes et al., 2022; Hrynaszkiewicz et al., 2021). Additionally, an open science archive will enable researchers to set the permissions and rights of access and reuse. Associating the repository with researcher ORCIDs ensures the work is linked to the authors for continuity of contact. Our review found that these simple steps were rarely followed (Monks & Harper, 2023a).

3.4.3. Knowledge barriers

An acknowledged barrier to open science is inadequate education and training. Early career researchers may feel that they do not have the requisite knowledge or are not fully supported in the practice (Gomes et al., 2022; Zečević et al., 2021). Despite this, Hrynaszkiewicz et al. (2021) found that early and mid-career researchers were more accepting of a mandatory code-sharing policy than senior researchers, indicating a willingness to engage with good research practice. Insecurity about publishing code is commonly reported (den Eynden et al., 2016; Gomes et al., 2022; Hrynaszkiewicz et al., 2021), and the Wellcome Trust (den Eynden et al., 2016) emphasised the need for training on code development standards, documentation practices, and licencing. Published examples and tutorials can additionally support learning, and increasing specialist guidance is available in the published literature for M&S researchers using FOSS tools such as R or Python (Harper & Monks, 2023b; Knight & Palmer, 2022; Monks & Harper, 2023c; Palmer & Tian, 2021; R. Smith & Schneider, 2020; Wojciechowski et al., 2015). Nonetheless, several authors have emphasised that a significant change in norms and culture is needed to embed openness into research practice (Janssen et al., 2020; Zečević et al., 2021).

4. The STARS framework

To address these challenges, we propose a framework to allow health researchers to Share Tools, and Artefacts for Reusable Simulation (STARS) models developed in a FOSS language. Before we describe the framework in detail we provide an overview of the aims, outputs, beneficiaries and alignment with existing initiatives.

4.1. Aims, outputs and beneficiaries

The primary aim of the STARS framework is to support simulation researchers to share a version of their FOSS computer simulation model that is accessible and reusable by others. In addition to researchers the

ultimate beneficiaries of models made open using the STARS framework are health services (e.g., the UK National Health Service (NHS)), other simulation researchers, early career researchers and students studying M&S.

Potential users of the framework can think of STARS as a set of open practices, tools and learning materials to produce enhanced versions of their research artefacts that are intended to increase the accessibility for others to (re)use, adapt, and build upon their work. Enhanced artefacts might include simulation models and novel algorithms/methods, study documentation, code and model repositories, journal articles, and lastly their own research portfolio.

As an example, consider a simulation model of an emergency department (ED) written as a simple Python script. A researcher using the STARS framework would publish a set of research artefacts designed to support others to reuse their work. This might include a web app interface version to the ED model, and specialist interactive documentation on the model design, code and procedural support in its use. The web app would allow ED managers and clinicians working within the health service to test the model and feedback on design. The interactive documentation would allow other researchers or students to explore the code workings, check model validation and testing, and run the model either by installing on their own laptop or by using remote open science infrastructure. All of these artefacts would have clear licencing defining how they can be reused and adapted; be properly archived with guarantees of persistence; and could be cited independently in the same manner as a journal article.

Figure 1 illustrates the aims, outputs (research artefacts) and beneficiaries of the framework.

4.2. Alignment and compatibility with existing initiatives

A requirement of STARS is that the framework must align and be compatible with existing initiatives. We considered the Association of Computing Machinery's (ACM) Replicating Computational Results (RCR) initiative, relevant general guides and tutorials to open science, and approaches to model documentation.

4.2.1. ACM replicating computational results (RCR)

The most well-known reproducibility initiative within simulation is the ACM RCR Initiative (ACM, 2023). An RCR is an optional peer review of computational artefacts (e.g., a model) that is offered by ACM journals such as *Transactions on Modelling and Computer Simulation*. Several literature reviews have demonstrated that healthcare simulation authors may publish in a wide variety of journals outside of ACM's

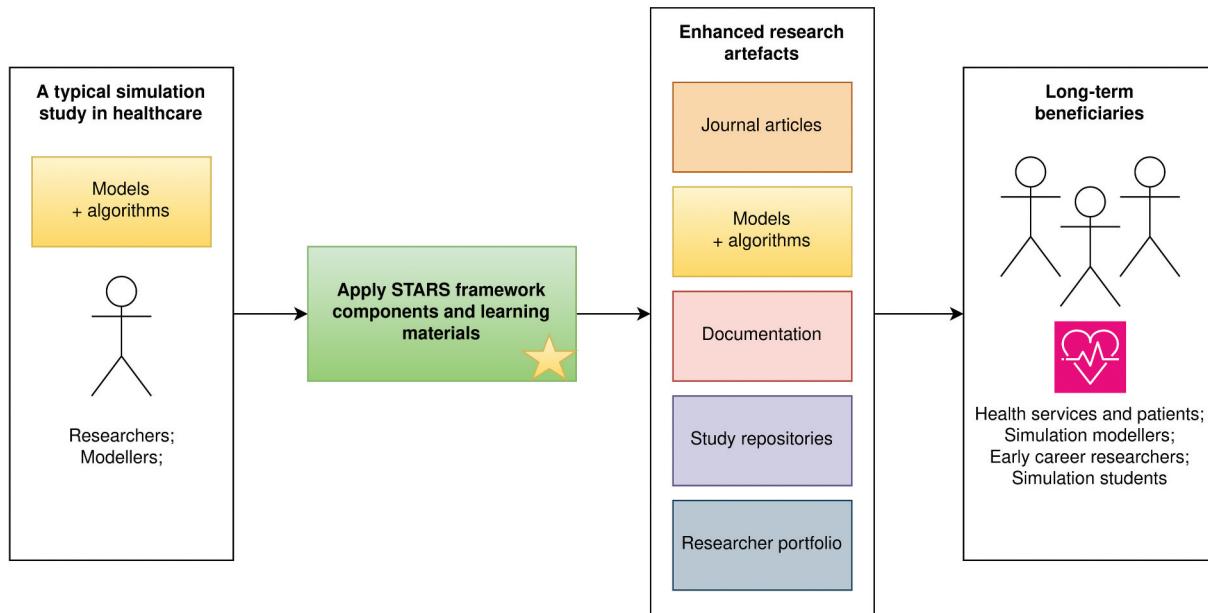


Figure 1. Outputs and beneficiaries of the STARS framework.

portfolio (Monks & Harper, 2023a; Vázquez-Serrano et al., 2021; Zhang, 2018) that do not offer an RCR. Nevertheless an RCR review is a gold standard for reproducibility; as such, we designed STARS to support authors to produce materials that meet RCR standards regardless of journal choice. We therefore opted to make STARS complement an RCR review by enhancing artefacts so that they are able to achieve three of the five ACM badges if reviewed: “artefacts available”, “artefacts evaluated – functional”, and “artefacts evaluated – reusable”. We note that the main aims of the pilot version of STARS are model reuse, longevity and accessibility. These aims are complementary to reproducibility of results published in a journal article, but do not guarantee it. We therefore do not guarantee that a study employing STARS would obtain a “results reproduced” badge with ACM’s RCR initiative. The final badge, “results replicated”, is not applicable, as it involves replication in the absence of author-supplied artefacts. We highlight which components support badges in the next section.

4.2.2. General open science guides

The STARS framework is designed to begin to tackle the deficiencies we identified in our open science best practice review of the healthcare DES literature (Monks & Harper, 2023a). We aligned the review with *The Turing Way* (The Turing Way Community, 2022), a guide to reproducible research published by the UK’s National Institute for Data Science; the OMF minimum reusability standards (OMF, 2023); and a Winter Simulation Conference advanced tutorial on open science (Taylor et al., 2017). It follows that the STARS framework complements these general guides specifically in the context of healthcare DES.

4.2.3. Documentation methods

We aimed to make the STARS framework components sufficiently flexible to incorporate existing approaches to describing, and documenting models, and simulation studies. For example, STARS could incorporate notebooks that might document the evolution of a simulation model or study. This for example, would allow applying methods such as TRAnsparent and Comprehensive model Evaluation (TRACE; Ayllón et al. (2021)). Reporting guidelines for computer simulation (Monks et al., 2019; Zhang et al., 2020) can also be incorporated via the framework.

5. Description of framework components

We recognise that open science takes additional work. Therefore, we designed the STARS framework (Figure 2) to divide work into *essential* and *optional* components. The essential components provide the minimum work needed to share DES models to allow them to be available long-term, citable, functional, appropriately licenced, and likely to be at a standard that is reusable in further work (shaded green in Figure 2). The *optional* components of the STARS framework focus on enhancing the accessibility, understanding, and maintainability of FOSS simulation models (shaded blue in Figure 2). In this section, we provide a general description of the components. The following section includes three applied examples and to support understanding we translate Figure 2 into a simple list of components (see Table 2).

5.1. Essential components

We designed the *essential* components to support authors to meet the requirements of two ACM badges

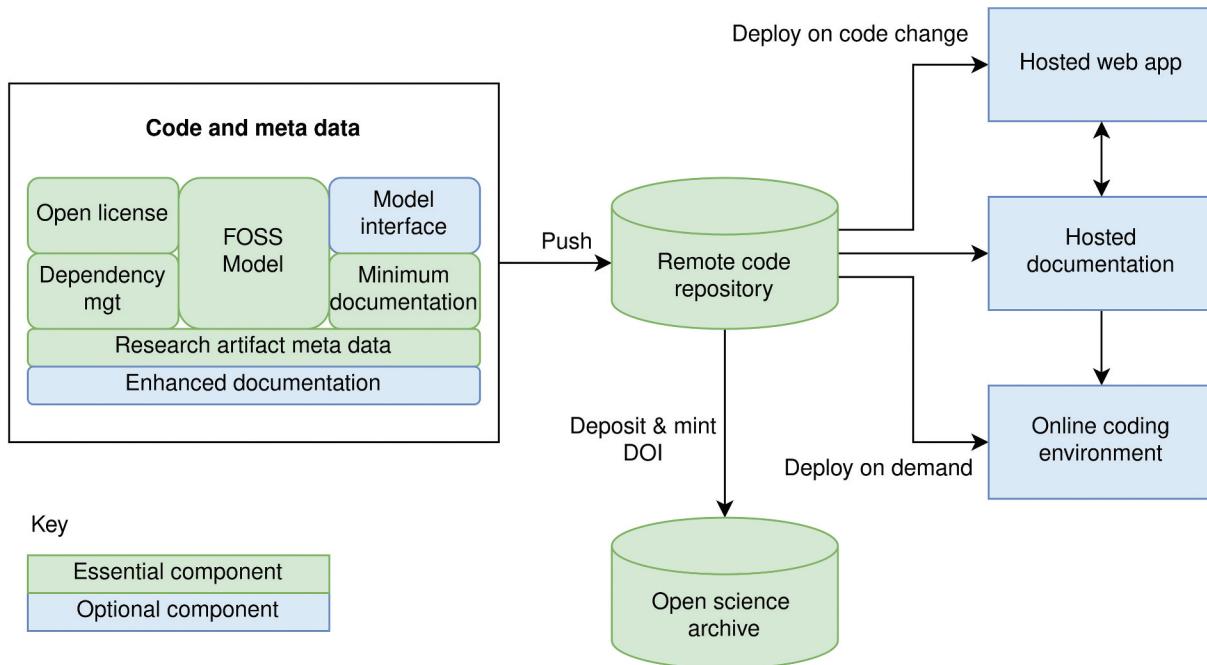


Figure 2. An overview of the STARS framework. Components shaded blue are optional: used depending on project requirements, skills, time available, and experience of team.

Table 2. Implementation of the framework across three examples.

STARS Components		Applied example 1	Applied example 2	Applied example 3
Essential	1. Open Licence	MIT	MIT	GNU Public Licence 3
	2. Dependency mgt	conda	conda	conda
	3. DES software	simpy	simpy	ciw
	4. Code repository	GitHub	GitHub	GitHub
	5. Meta-data	citation.cff + ORCID	citation.cff + ORCID	citation.cff + ORCID
	6. Minimum documentation	README.md	README.md	README.md
	7. Open science repository	Zenodo	Zenodo	Zenodo
	8. Online coding environment	Binder	Binder	Binder
	9. Enhanced documentation	Electronic Notebook	Jupyter Book + STRESS	Quarto + STRESS
	10. Documentation hosting		GitHub pages	GitHub pages
Optional	11. Model interface		streamlit	Shiny for python
	12. Web app hosting		streamlit community cloud	shinyapps.io

if tested in an RCR: “artefact available”; and “artefacts evaluated – functional”. A study implementing the essential components would meet the majority of the best practices evaluated by Monks and Harper (2023a).

5.1.1. Open license

Before sharing (or making any code public), it is necessary for authors to select an appropriate FOSS licence for the code and other research artefacts. This ensures appropriate use of the model, and if needed removes legal liability of the authors. A common approach in data science and other computational fields is to adopt a permissive licence such as the MIT, but other options exist. For example, a model could be made available under a GNU Public Licence (GPL). The GPL is an example of a copyleft licence: where the rights granted must be preserved in all derivative works (typically by publishing under the same GPL licence). Without a licence, the model is

available, but not (re)usable. Authors retain exclusive copyright and any derivative works are at risk of a take down request from the original authors.

Adding a licence to a simulation project is a minor task. For support selecting a licence and for licence text we recommend consulting the online resources provided by “Choose an Open Source License”.² Popular online code repositories such as GitHub and GitLab also provide templates and simple workflows to automatically add licences to a project.

5.1.2. Dependency management

For any type of software or code deployment problem, it is necessary to adopt a language that allows model developers to specify the software libraries, version numbers and sources that are required in order for the model to run. Without this information the likelihood of others being successful in reusing models is

greatly reduced as models will generate missing dependency error that researchers will be required to solve. Monks and Harper (2023a) found that the vast majority of healthcare authors that coded their DES models did not make use of formal dependency management.

If we take a popular FOSS language such as Python for example, three well-known options for managing dependencies exist: pip virtual environments, conda environments, and poetry. Each of these approaches allow a user to accurately record the key Python software used in the project. In turn the dependency management tools then reconcile the dependencies required for this software. For example, conda virtual environment document dependencies in the *Yet Another Mark-up Language (.yml)* format.

Once learnt, using a dependency management tool is a simple task and has low time overhead to setup. Our experience is that dependencies may change during a project and that some rework of environment files is needed during the project (as dependencies are added or removed) along with a final check at the end.

5.1.3. DES model

The STARS framework is designed to be general to any DES model coded in a FOSS simulation package; for example, using R, Julia or Python. In our application exemplars, we demonstrate different ways to use the framework using two popular simulation packages in Python: *Simpy* and *Ciw*. There are no formal requirements within STARS for how a model is structured or implemented. We do recommend that modellers aim to keep model code independent of any *front-end* and is executable standalone. Guidance and support for how setup could be achieved in Python is available elsewhere (Monks & Harper, 2023b, 2023c).

5.1.4. Minimum documentation

Within the STARS framework, DES models are shared with a set of minimal instructions that provide an overview of what a model does, how to install and run the model to obtain results, and how to vary parameters to run new experiments. A simple way to achieve this is a README file. In data science projects, this information is often contained in a markdown file in the root directory for the modelling project (i.e., README.md). An advantage of this approach is that online code repositories (GitHub/GitLab/BitBucket) show the contents of this file as part of the project's webpage, that is, users do not need to download the documentation, they can read it online.

5.1.5. Research artefact meta-data

A model shared using STARS is stored with data describing the researchers and the artefact. STARS requires two basic items of meta-data. The first are the Open Researcher and Contributor IDs (ORCID)

for each study author. An ORCID is a unique identifier for a researcher; it provides a permanent link back to an authors publication (including related publications) and funding history. For example, the lead author of this papers ORCID is 0000-0003-2631-4481 and their details can be accessed via <https://orcid.org/0000-0003-2631-4481>. A minimal implementation to store this information is to include researcher ORCIDs in the README.md file.

While it is easy for a researcher to cite an academic paper, it may appear less obvious how to cite a model archive (e.g., model code). The second requirement is therefore citation information for the research artefact i.e., a way to provide credit to the author, and direct readers to original code, if the work or derivatives are used in any future study. A solution is to create CITATION.cff file that contains all the citation information for the model and code artefacts (e.g., name, authors, version). This approach is support by remote code repositories (e.g GitHub), Open Science Archives (we shall describe these shortly) and some reference management software tools. There are online tools to support the creation of CITATION.cff files (Druskat et al., 2021). The next section provides three applied examples implementing CITATION.cff and links to authors via ORCIDs.

5.1.6. Remote code repository

Central to Figure 2 is a remote code repository. The most popular solutions are GitHub,³ GitLab,⁴ and BitBucket.⁵ GitHub and other online code repository's primary purpose is version control of code. That is, modellers incrementally commit updates to their code and models during development. These tools allow for easy *rollback*, inspection, or use of older versions of the code. Another major feature of version control is *branching*. The main (sometimes called master) branch of a repository contains the production model i.e., a simple, but functional simulation model that can be deployed to users. Incremental developments are typically made on a development or feature branch, tested and then merged into the main branch when a modeller is confident it will not break the production model. GitHub is an online tool. In traditional workflows, all of the model coding is done off-line in a local code repository that is managed by a local version control system such as Git. A modeller codes their model on their own machine, and *pushes* their commits to GitHub (or a similar repository). A new user of the code would *clone* the GitHub repository to their own local machine. A user who is already using the code might *pull* updates from the remote repository and *merge* changes into their own branch.

As a simple example, consider a model of an emergency department (ED) developed in the Python DES package SimPy. The model logic is fully coded and it

can be run from the command line. The working SimPy model is currently committed to the main branch. A researcher can *clone* the main branch of the repository and execute the model to analyse the ED in their study. Simultaneously, a simulation modeller can develop a graphical interface for the model on a separate feature branch (e.g., called *interface*). When the interface is complete the modeller can issue a pull request to the main branch. A pull request is effectively a request to merge code from one branch into another e.g., from *interface* to main. It provides a safety net for the changes to be reviewed before merging (and potentially breaking a production model). Once merged the researcher can simply pull the new code from the main branch, although there are no guarantees the new model interface will install easily on the researcher's machine.

A secondary use of online code repositories is for collaboration on scientific studies and model development. These solutions provide many mechanisms for its users to share, review, and control code. One simple example is *Issues*. This is a discussion log that might report bugs, potential improvement, new ideas, and general queries. Another example is the concept of a *release*. A release tags a snapshot of the code. In our ED model example, the SimPy command line model might be tagged as version 1.0.0. While the ED model with a graphical user interface might be tagged as version 2.0.0. A release is an easy way for modellers to talk to their users about the correct version of the model: it supports a shared language between modellers and users.

5.1.7. Open science archive

Our last essential component is for long-term archiving (and preservation) of simulation models. Our framework links code in the remote repository to an open science archive that has FORCE11 compliant citation (A. M. Smith et al., 2016) and guarantees on the persistence of digital artefacts by adhering to TRUST principles (Lin et al., 2020). Examples, include Figshare,⁶ Zenodo,⁷ the Open Science Framework,⁸ and the Computational Modeling in the Social and Ecological Sciences Network (CoMSES Net).⁹ Each of these archives has guarantees on storage and mints Digital Object Identifiers (DOIs) to enable citation and, includes meta-data to improve discoverability. In the long term, open science archives hold significant advantages over remote code repositories. For example, code in a GitHub repository could change over time, be made private, or be deleted.

As an example consider the orthopaedic planning simulation model code published by Harper and Monks (2023b). The model has been deposited in Zenodo and has been allocated the DOI [10.5281/zenodo.7900852](https://doi.org/10.5281/zenodo.7900852) (accessed via <https://zenodo.org/record/8011462>). This artefact is citable in the same

way as a journal article and uses ORCID meta-data to link the work to the authors research portfolio.

Depositing models in open science archives fits seamlessly into the journal article publication lifecycle using versioning. Before a manuscript is submitted to a journal, the model code is deposited, a DOI is minted, and the artefact is cited in the article. If changes are required to the model code after the initial peer review these are then re-deposited, a new DOI is minted for the artefact, and the new deposit is cited in the revised article. The deposit is automatically linked back to the previous version using the original DOI.

5.2. Optional components

The *optional* components of the STARS framework focus on enhancing the accessibility, understanding, and maintainability of FOSS simulation models (shaded blue in Figure 2). Users may include one or more of the optional components to improve the likelihood of their simulation model being reused. We designed the optional components to support healthcare simulation authors to meet the requirements of one ACM RCR badge: "artefacts evaluated – reusable".

5.2.1. Enhanced documentation

The STARS framework advocates that any simulation used for health decision making should provide open and high quality documentation on how the model is implemented and works. We encourage users to adopt contemporary FOSS tools to support the creation of these materials.

A simple solution is to adopt Jupyter notebooks for documenting a model. Project Jupyter is a FOSS project that supports scientific coding in Python, R and Julia in a classical notebook format. A simple way to conceptualise Jupyter notebooks are as a browser-based interface and an execution kernel. The interface provides a set of markdown and code cells that a user can control. Markdown is simply a way to present formatted text, mathematics, and code together. Notebooks provide a way to explain, document and execute simulation experiments. For example, an analyst may have created a set of Jupyter notebooks for model validation, determining the number of replications, warm-up analysis, defined experiments, or interactive experimentation.

Jupyter's kernel executes all of the code. Typically, the kernel is located on the users local machine, but given the server nature of Jupyter, the kernel could be located on a remote (possibly powerful) machine. In these remote cases the user need not have the FOSS simulation package or Jupyter installed on their own machine.

Notebook and markdown files containing simulation model code and documentation can be brought together using contemporary publishing software.

Two examples are Quarto¹⁰ and Jupyter Book.¹¹ These are high level tools that offer different ways to collate and publish documentation. For example, both can be used to create a structured website. This provides researchers the opportunity to construct a detailed book that could be hosted online for their simulation study with the aim of supporting simulation model understanding and appropriate reuse. While the STARS framework is not prescriptive about the content included in enhanced documentation, authors are encouraged to consider some or all of the following areas:

- A plain English summary of the project context, and model;
- Clarifying simulation model open licensing terms;
- Clear citation instructions for the model and documentation;
- Instructions for how others can contribute (e.g., code, bug fixes, modifications, improved documentation), and report issues to the project;
- Instructions for installing and using the model;
- A structured code walk through of the model;
- Documenting the modelling cycle using TRACE (Ayllón et al., 2021);
- Annotated simulation reporting guidelines e.g., the STRESS-DES.
- A clear description of model validation including its intended purpose.

5.2.2. Documentation hosting

STARS users could, as an option, choose to host their enhanced documentation online using a free web hosting service. At the time of writing, there are multiple secure stable services for serving static content; for example, GitHub Pages (owned by Microsoft),¹² GitLab Pages,¹³ BitBucket Cloud,¹⁴ or Quarto Pub (for Quarto created content).¹⁵ Although all of these services have limits on their usage, they would be suitable for a healthcare simulation project; for example, at the time of writing GitHub pages has a limit of 1GB of content and a *soft* bandwidth of 100GB per month. In our applied examples the largest website hosting enhanced documentation was < 5MB in size. This would allow for more than 20,000 visits per month, assuming all site pages were visited by each user, before soft limits were reached.

These services offer a way to publish companion enhanced documentation with a research article. However, STARS recognises the transient nature of technology and all enhanced documentation should be deposited in an Open Science archive to guarantee its longevity and maintain a snapshot of the documentation at the time of article publication.

5.2.3. Online coding environment

Jupyter's use of the browser and ability to run code on a remote kernel allows for easy deployment of simulation notebooks to a technical user-base. In the UK for example, these may be well trained NHS analysts from the NHS-Python/NHS-R community or other researchers familiar with reading and running code. In summary, an optional component of the STARS framework is to provide an online environment where users can run and change code.

A FOSS solution for deployment is to make use of a BinderHub such as mybinder.org. A BinderHub provides a Jupyter environment in the cloud that others can use to run your simulation model. One simple requirement is that the FOSS simulation model code, and Jupyter notebooks are available in a public code repository hosted in the cloud; e.g., on GitHub or GitLab. Binder works by on-demand containerisation of a code repository and code and deployment. In computing, containers refer to lightweight, portable, self-contained software units that package together an application, such as a simulation model, and all its software dependencies, including libraries, frameworks, data, and system tools. Within STARS, for example, this would make use of the essential dependency management approached used (e.g., conda) to install the necessary libraries to run the model on a linux operating system and then deploy to the binder web server.

One advantage of a BinderHub approach is that it can integrate with enhanced documentation components of STARS. For example, a notebook that is part of a Jupyter Book can become interactive at the click of a button.

In addition, to Binder's FOSS solution, proprietary online notebook alternatives also exist. For example, Google Colaboratory¹⁶ or Deepnote.¹⁷ These are closed source, but offer both free and paid tiers of service. Unlike BinderHub the software environment is not specified before launch. For example, a user would need to attempt to install a simulation package after a notebook has been opened.

5.2.4. Model interface

A web application (web app) interface to the model is accessible to less technical simulation users such as healthcare analysts, healthcare managers, researchers not familiar with Python or R, or the general public. A web app will provide simple ways to setup and execute the model. This might include parameter fields, logic diagrams, basic animation, and buttons.

For example, in Python there are multiple solutions available to create an interface for a model e.g., Streamlit (streamlit.io, 2022), Shiny for Python (shinyapps.io, 2023), or plotly dash (Hossain, 2019). An important point to note here is that there is no

requirement that model interfaces are hosted on a web server. They are browser-based technology and, just as enhanced documentation, can equally be installed and run on a laptop or desktop. This opens STARS up to different types of project including supporting private code – for ethical or governance reasons – but shared between a project team.

Each solution provides its own API for screen widgets (e.g., sliders, results tables, buttons) as well as unique benefits and downsides for simulation. If deployed to a web app hosting service then the simulation models become much more accessible; for instance they are designed to display and be usable via phones, and other mobile devices. This brings us to the final optional component for deploying web apps online.

5.2.5. Web app hosting

In healthcare simulation, online deployment of computer models, opens a new channel for the FOSS community to work with busy clinicians and health professionals. For example, a simulation researcher working with a paediatric consultant, could prototype a web app containing a simulation model of a children's emergency department, using Python and streamlit; deploy to an online host, and message the consultant the URL. The busy consultant can then review when they are free, perhaps on their phone, and message back with feedback.

Web app software, such as streamlit, or Shiny, may have its own free tier of hosting. For example, StreamLit community cloud is a free hosting service for Streamlit apps. When used together they provide a robust and easy way to share simulation models to healthcare users. As it is the aim of these services, deployment of apps is often trivial. For example, Streamlit Community Cloud only needs to be given access to a GitHub repository and it copies the code, builds the app and deploys to a user customised URL. As with BinderHub, the free tier of any web app hosting service has limitations; typically in the number of processors available, memory and/or monthly bandwidth. If required, services like ShinyApps offer monthly paid plans to increase computational resources. If apps are not used for an extended period then service providers often suspend the app. The next user that accesses the URL then faces a short wait while the app wakes up.

6. Applied examples in Python

To demonstrate the feasibility of our framework and support researchers in its use, we provide three applied examples. In each example, we vary the technology, approach, and degree of effort (implemented

components), to illustrate the flexibility available to researchers. For example, varying the open licence, Python simulation software, web application, documentation tools and hosting. **Table 2** lists all components from **Figure 2** and summarises the tools used in each example to share research artefacts against essential and optional enhanced framework components. For simplicity we also summarised the artefacts archived in Zenodo in **Table 3**

All artefacts are archived in Zenodo. Follow the links in the Table to find the exact version of the code at time of submission to the Journal of Simulation.

6.1. Python DES packages

The advantages of Python for simulation modelling, including its simplicity and readability, have been described in Daggakis and Heavey (2016). The Python data science ecosystem contains a large number of modules, libraries and frameworks for supporting analysis, flexible modelling, and integration of simulation with other methods such as optimisation, machine learning, and visualisations.

In our applied examples, we employ two Python DES FOSS packages: *SimPy* (SimPy, 2020) and *Ciw* (Palmer et al., 2019). We chose these packages as they are both maintained, have recent publications using or developing them, and adopt permissive open licencing.

Ciw was developed with a focus on reproducibility, and best practice in terms of documentation, readability, transparency, version control, and testing. It allows users to very quickly build complex multi-class queuing networks. It is modular and modifiable, and by integrating with the Python ecosystem, several examples of open hybrid simulations have been implemented (Palmer & Tian, 2021, 2023). *Ciw* has been used for applied healthcare simulation modelling, for example for planning large-scale outpatient services (Zou et al., 2022).

Table 3. Simulation artefacts and archives from the applied examples.

Example 1	URL
Code repositories	https://github.com/pythonhealthdatascience/stars-treat-sim
Archive and DOIs	https://doi.org/10.5281/zenodo.10026327
Example 2	URL
Code repositories	https://github.com/pythonhealthdatascience/stars-streamlit-example
Web app	https://stars-simpy-example.streamlit.app
Documentation	https://pythonhealthdatascience.github.io/stars-simpy-example-docs
Archive and DOIs	https://doi.org/10.5281/zenodo.10055169
	https://doi.org/10.5281/zenodo.10054063
Example 3	URL
Code repositories	https://github.com/pythonhealthdatascience/stars-ciw-example
web app	https://pythonhealthdatascience.shinyapps.io/stars-ciw-exemplar
Documentation	https://pythonhealthdatascience.github.io/stars-ciw-example
Archive and DOIs	https://doi.org/10.5281/zenodo.10051495

Simpy is the most commonly used Python DES package. It is easily mixed with standard Python and runs using Python generator syntax. *Simpy* has been used in several open-source OR-relevant publications (M. Allen et al., 2020; Chalk et al., 2021; Cubukcuoglu et al., 2020; Harper et al., 2023; Lim et al., 2020).

6.2. Applied example 1: A minimal implementation

Our first example is a minimal implementation of the STARS framework. We use all essential components to enhance a *Simpy* simulation model. We then further enhance the model by creating a Jupyter notebook containing an annotated explanation of each section of code in the model and instructions to run the model. We made the notebook executable online using Binder.

6.2.1. Case study model

We adapt a textbook example from Nelson (2013, p. 170): a terminating DES model of a U.S-based treatment centre. In the model, patients arrive to the health centre between 6 am and midnight following a non-stationary Poisson process. On arrival, all patients sign-in and are triaged into two classes: trauma and non-trauma. Trauma patients include impact injuries, broken bones, strains or cuts etc. Non-trauma include acute sickness, pain, and general feelings of being unwell etc. Trauma patients must first be stabilised in a trauma room. These patients then undergo treatment in a cubicle before being discharged. Non-trauma patients go through registration and examination activities. A proportion of non-trauma patients require treatment in a cubicle before being discharged. The model predicts waiting time and resource utilisation statistics for the treatment centre. The model allows managers to ask question about the physical design and layout of the treatment centre, the order in which patients are seen, the diagnostic equipment needed by patients, and the speed of treatments. For example: “what if we converted a doctors examination room into a room where nurses assess the urgency of the patients needs”; or “what if the number of patients we treat in the afternoon doubled”

6.2.2. Software environment

We used Python 3.8. Data manipulation, simulation, and general mathematical modelling were done using Simpy v4.0.1 (SimPy, 2020), NumPy v1.19.2 (van der Walt et al., 2011) and Pandas v1.2.3 (McKinney, 2011). Charts were produced with Matplotlib v3.3.4 (Hunter, 2007). Enhanced documentation was created using Jupyter-Lab v2.4.3 (JupyterLab, 2022).

6.2.3. Implementation of essential framework components

We used a GitHub repository to host and version control our production code (<https://github.com/python-healthdatascience/stars-treat-sim>). All core computational materials (Figure 2) have been made available under an MIT permissive licence. Any users of the research artefacts and DES model are free to re-share the model under any terms they choose including commercial, but they must cite the authors and accept that the authors do not have any liabilities. We made the Python software environment explicit and reproducible using the conda package manager. Basic documentation is provided using a README file i.e., basic instructions for installing and running the model are contained in the README file that is seen when navigating to the GitHub repository URL. Our README file also contains markdown badges displaying the authors ORCIDs and are hyperlinked to the authors' individual ORCID records. We then archived all essential components for this publication (all contained with the GitHub repo) in Zenodo. The materials are citable using DOI: [10.5281/zenodo.10026327](https://doi.org/10.5281/zenodo.10026327) (Monks & Harper, 2023d).

6.2.4. Optional components: Documentation and online interactive code

In our minimal example, we first enhanced model documentation using a Jupyter Notebook. The notebook breaks down the code for model inputs, logic and usage into sections and provides a detailed explanation of each.

Given that we had created a notebook, we used Binder to provide an online instance of Jupyter Lab for expert users to run the model's Python code without the need to instal any software on their local machine. Binder expects that a conda virtual environment.yml file is placed into a sub-directory binder/. Once this has been committed it is a simple case of navigating to <https://mybinder.org> and copy and pasting the URL into the build and launch text box. Binder will then create a remote instance that contains both the code and dependencies specified in the binder/environment.yml file. The first time the instance is built it will likely take several minutes. This process may need to be repeated if the instance is not used regularly or code is changed.

A model instance can be launched on Binder from the Github repository. A user need only click on the badge in the README file labelled “Launch Binder”. The model notebook can be found src/full_model.ipynb. You will be presented with a notebook; an example is in Figure 3.

6.3. Applied example 2

Applied example 2 is again based on essential STARS framework components from example 1 (using the treatment centre case study model) and fully implements the optional components including enhanced

```

1: To run all code in this notebook go to menu item Run -> Run All Cells .
2: simpy uses process based model worldview. Given its simplicity it is a highly flexible discrete-event simulation package.
3: One of the benefits of a package like simpy is that it is written in standard python and is free and open for others to use.
4: • For research this is highly beneficial:
   • models and methods tested against them can be shared without concerns for commercial licensing.
   • experimental results (either from model or method) can be recreated by other research teams.
5: • The version of simpy in use can also be controlled. This avoids backwards compatibility problems if models are returned to after several years.
6: Here we will take a look at code that implements a full simpy model including, time dependent arrivals, results collection, control of random numbers and multiple replications.
7: The full scope of what is possible in simpy is out of scope. Detailed documentation for simpy and additional models can be found here: https://simpy.readthedocs.io/en/latest/

```

Imports

It is recommended that you use the provided conda virtual environment `treat-sim`.

If you are running this code in Google Colab then `simpy` can be pip installed.

```

[1]: # install simpy if running in Google Colab
import sys
if 'google.colab' in sys.modules:
    !pip install simpy==4.0.1

[2]: import simpy
simpy.__version__
[2]: '4.0.1'

```

Figure 3. Simulation model and Jupyter lab running in Binder.

documentation that is hosted online, and a web app to support usability (Figure 2). We created a new GitHub repository to host our essential components and model interface (<https://github.com/pythonhealthdatascience/stars-streamlit-example>).

6.3.1. Web app: Streamlit

Our web app is implemented in *streamlit* and hosted on streamlit's free community cloud (<https://stars-simpy-example.streamlit.app>). We have provided

a multi-page application allowing users to switch between an interactive simulation, a fixed set of reproducible experiments, custom batches of experiments, and an “about” page providing information and links to enhanced documentation. Deployment of the web app is controlled by linking a streamlit community cloud account to our GitHub repository. Figure 4 illustrates the interactive page from the web app: users can manipulate model parameters using sliders and are presented with results.

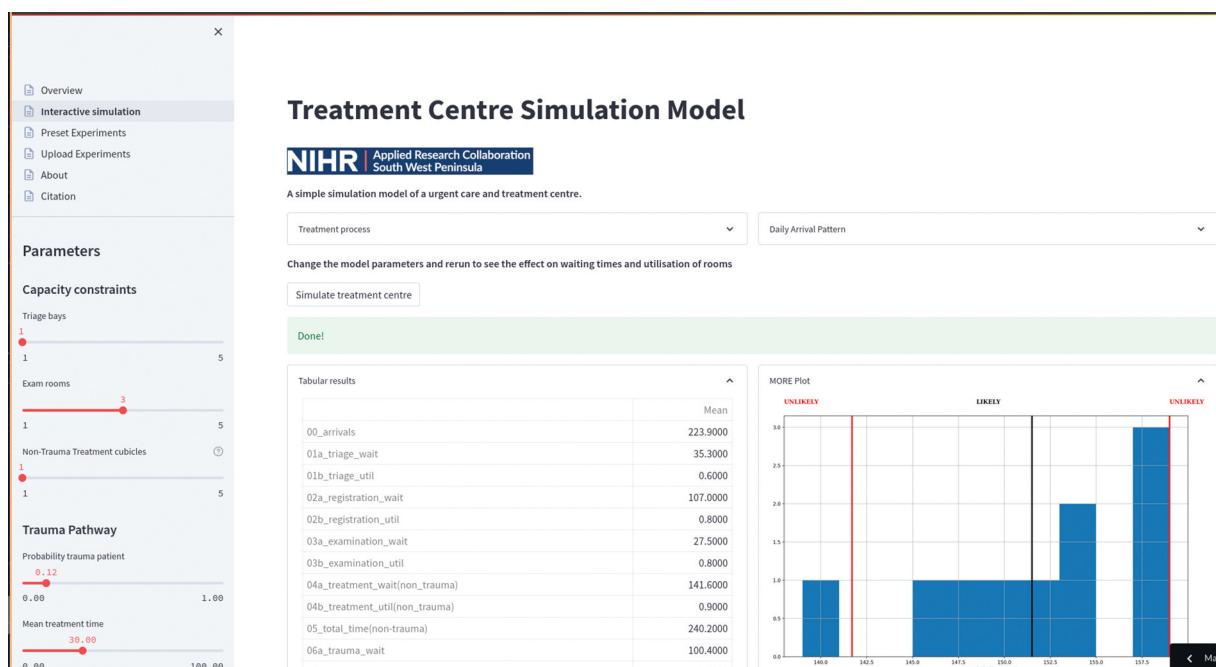


Figure 4. Treatment centre user interactive simulation.

Live code for the streamlit interface to the model is available under the MIT licence in GitHub. Code is also archived in Zenodo and citable using DOI [10.5281/zenodo.10055169](https://doi.org/10.5281/zenodo.10055169) (Monks & Harper, 2023e)

6.3.2. Enhanced documentation: Jupyter Book

For our documentation we created a second Github repository <https://github.com/pythonhealthdatascience/stars-simpy-example-docs> again made open with an MIT licence.

We incorporated our computational notebooks into an interactive Jupyter Book and hosted our enhanced documentation for the model on GitHub pages <https://pythonhealthdatascience.github.io/stars-simpy-example-docs>. The book combines plain English explanations of the model, with code explaining the inner works of the model. In our example, we used this to include more detailed sections on the STRESS-DES reporting guidelines (Monks et al., 2019). All material was again archived using Zenodo under DOI [10.5281/zenodo.10054063](https://doi.org/10.5281/zenodo.10054063) (Monks & Harper, 2023f).

6.4. Applied example 3

Our third example again fully implements all components of the STARS framework. To demonstrate how different tools can be used, we vary the open licence, enhanced documentation publishing software, web app framework and web app hosting. In this example we enhance the sharing of a simulation model coded using *Ciw*.

6.4.1. Case study model

We reuse a stylised urgent care call centre model that we have previously published Monks and Harper (2023b). In the model, a telephone call from a patient with urgent care needs arrives at random to a call centre. The centre is staffed by call operators who answer calls from a first-in-first-out queue. Patients are triaged, and provided a call designation; for example, whether the patient should be allocated an appointment in primary care with a General Practitioner (family doctor) within 48 hours, or if a call back from a nurse is needed. Callers that are designated as needing a nurse callback enter a first in first out queue until a nurse is available.

6.4.2. Software environment

We used Python 3.9. Data manipulation, simulation, and general mathematical modelling were done using *ciw* v2.3.1 (Palmer et al., 2019), NumPy v1.25.0 (van der Walt et al., 2011) and Pandas v2.0.2 (McKinney, 2011). Charts were produced with *plotly* v5.15.0 (JupyterLab, 2022). The web app was produced using Shiny for Python v0.4.0 (shinyapps.io, 2023). Enhanced documentation was created using Quarto v1.3 (quarto.org, 2023).

6.4.3. Implementation of essential framework components

We varied two components from applied example 2. Firstly we made all code and the model available under a GNU Public License (GPL) version 3. This is a copyleft licence and means that any code derived from our own must be released under the same licensing terms i.e., a GPL v3.

The second change was to vary the FOSS simulation software used. We implemented the call centre model in *ciw* as opposed to *simpy*. Production code is available on Github <https://github.com/pythonhealthdatascience/stars-ciw-example> and archived in Zenodo (Monks & Harper, 2023g).

6.4.4. Web app: Shiny for Python

Our web app is implemented in Shiny for Python and hosted on shinyapps.io (<https://pythonhealthdatascience.shinyapps.io/stars-ciw-examplar>). The model is simpler than applied example 2; we therefore provide a simple two page app that allows users to vary a small number of parameters and run multiple replications. Results are visualised in an interactive histogram using the *plotly* library. A second “About” page provides some basic details about the project and a link to the enhanced documentation.

6.4.5. Enhanced documentation and coding environment

Our enhanced documentation code is stored in the same repository as the simulation model. We created a simple companion website using Quarto and hosted online using GitHub pages (<https://pythonhealthdatascience.github.io/stars-ciw-example>). In addition to usage instructions and model code walk through we used the website to provide an annotated version of the STRESS-DES reporting guidelines for the model. As all code is in the same repository the artefacts would be cited using the same DOI as the essential components (Monks & Harper, 2023g).

Quarto websites do not integrate as seamlessly with BinderHub as Jupyter Books. We therefore created a custom link to launch a Binderhub instance within the website.

7. Discussion

In this study, we have demonstrated that it is feasible to share open and reusable DES models for healthcare research using existing FOSS and Open Science technology. In support of our claim that our proposed framework is usable we have provided three applied examples of varying complexity using Python and DES FOSS tools. Other researchers could, if desired, adopt our approach to share their own DES models and research artefacts. While our framework approach works towards overcoming the well-known barriers to

open science that our review identified, we note that there are still many challenges for the M&S community going forward. We now describe the strengths and contributions of the STARS framework, its current limitations, challenges for the M&S healthcare community, and further work.

7.1. Strengths and contributions

7.1.1. Enabling new forms of collaboration

The FOSS tools within our framework enable potential model users such as healthcare staff to test it out without going through any installation, source, or dependency management. The model interfaces we have demonstrated via web apps are contemporary tools that are often optimised for display on devices such as phones or tablets. This can support researchers to collaborate with busy healthcare clinicians and enable fast feedback and revision of a model to fix mistakes or better meet user requirements. The STARS optional components allow users to experiment and generate results from a simulation model during and beyond a funded project period.

7.1.2. Model reuse

A researcher or healthcare analyst/modeller may be interested in testing or modifying a simulation model developed and tested in a given application area to a similar problem in a different health service organisation. Use of the essential components within the STARS framework can enable this by making the model accessible and reusable. For researchers, it also generates an archive of M&S code under open licence for reuse by others, which avoids duplication of research effort and can advance research progress.

7.1.3. Tackling the barriers to open modelling

The barriers and challenges to sharing models and code are well documented in the open science literature (Celi et al., 2019; den Eynden et al., 2016; Goldacre et al., 2019; Gomes et al., 2022; Hrynaszkiewicz et al., 2021). One significant barrier is the time needed for open science activities. To be clear, to adopt the STARS framework, as with any open science practice, does involve time and effort. A strength of STARS is that we have split the framework into essential and optional components to support researchers to manage their time in a project. The essential components included in STARS require minimal time for a researcher and include some automation, for example, selecting and adding an open licence to a code repository, creating citation meta-data and depositing in an open science archive. This overhead is likely to be larger when the framework is initially adopted, but will become less over time as researchers become familiar with the components. We also highlight that several of the

essential components can be front loaded in a project. An example, workflow in an open modelling project might begin by creating an online (public or private) code repository, automatically adding a licence, automating the creation of a citation file, creating a README documentation file, and then specifying the list of software dependencies to be used. We again emphasise that successful reuse of models will rely on good quality documentation (den Eynden et al., 2016).

The optional components offer substantial support to health care for open modelling and its accessibility, but by definition also add substantial work to a project. One strength of the technology we have used in our applied examples is that researchers can plan this work into their project from the beginning. For example, with enhanced model documentation, the initial tasks in a DES study could include creating a basic Quarto website (either locally, or published online). This could be comprised of the project README file initially, and as the project progresses expanded to include more usage support from the options we outline in section 5.2.1

7.1.4. Benefits for early career researchers

There are considerable benefits of open science for early career researchers and the time needed for open science should be seen as an “investment” (C. Allen & Mehler, 2019; McKiernan et al., 2016). A key career benefit of open science appears to be increased citations, for example, studies that publish open data alongside an article receive more citations (from other authors) over the long term (McKiernan et al., 2016; Piwowar & Vision, 2013).

We also argue that shared models benefit early career researchers in terms of learning. Others have called this sharing for benefaction (Collberg & Proebsting, 2016). Open models that are accessible, for example, via enhanced documentation or a web app interface, offer the community a way to learn how simulations are *implemented*, and designed at a very detailed level, from a diverse and wider group of modellers than their usual collaborators. This is equally important for the learning and retention of new students of simulation, keeping the discipline of simulation competitive with other areas of data science, such as machine learning, where resources such as *papers with code*¹⁸ make it easy for students to discover and [re]use research artefacts.

7.2. Challenges for the healthcare M&S community

The final objective of the study was to identify limitations of the framework and remaining open science challenges for the M&S community. We detail these in this section.

7.2.1. The switch to FOSS technology

One downside of code-based FOSS models at the heart of STARS is that the switch from commercial software can be challenging. Commercial software is convenient and powerful. For those researchers unfamiliar with R or Python, there are abundant learning materials and guidance available for general applications. For specialist applications, there are increasing resources available to support the transition (Harper & Monks, 2023a, 2023b; Knight & Palmer, 2022; Monks & Harper, 2023c; R. Smith & Schneider, 2020; Wojciechowski et al., 2015).

An obvious direction for STARS is therefore to provide the healthcare community with additional, specialised learning resources for building and deploying shareable DES models using FOSS. We acknowledge that this is substantial task and will require additional input from the M&S community. Towards this aim, we have provided three pilot applications of STARS within this publication and have published extensive materials elsewhere to support web-based simulation using Python (Monks & Harper, 2023b, 2023c). The Health Technology Assessment community have also been working to support researchers using R in their modelling (R. Smith & Schneider, 2020; R. Smith et al., 2022, 2023).

7.2.2. Feasibility and pilot studies in other FOSS languages

While a strength of the STARS framework is that it is not prescriptive in terms of the tools or technology used, we have only demonstrated its feasibility with Python DES models. An obvious next step for STARS is to test the feasibility and applicability to models developed in other relevant FOSS languages.

There is some evidence to support the generalisability and utility of STARS to the healthcare M&S community. Specifically, our review found that R was another common FOSS used for healthcare simulations. We have previously argued that several of the technologies used in STARS are agnostic to the programming language used (Harper & Monks, 2023a). For example, R Shiny can provide a simulation front end to a model built using R Simmer or base R, the *renv* library (RStudio, 2023) can provide dependency management, and publishing systems such as Quarto are agnostic to languages used.

7.2.3. Limitations of remote computing

The majority of healthcare DES studies use standard simulation tools and have standard hardware requirements (Monks & Harper, 2023a). At the time of writing a minority of healthcare simulation studies incorporating DES models such as FACS-CHARM (Anagnostou et al., 2022) require powerful computing infrastructure. The free tiers of the computing infrastructure we have described in the optional

components of STARS for an online coding environment and a web app will not be sufficient to fully support these projects. In such cases highly specialised infrastructure may be needed such as a high performance computing cluster in an academic institution or general paid cloud compute resources, e.g., Google Cloud. For the latter STARS can still be employed, but the artefacts do not have the same longevity i.e., when a study funding ends so does paid web app hosting. The essential components of STARS are still relevant to projects with high compute requirements (for example, there is always a need for dependency management, open licencing, and archiving of code) and would still support model reuse.

Although we provide no formal guidance, a variation of STARS could also be considered where modellers deploy their simulations via containerisation and continuous integration technology (Harper, Monks, & Manzi, 2023; Moreau et al., 2023). For example, when a model is updated a Docker image is automatically built and deployed to DockerHub (a container registry). The container (which includes an operating system, software dependencies, and a model) could be pulled by a user and run within their own environment on hardware with the correct specification. This mode of deployment can also support projects where there are real or perceived data confidentiality and security issues. Containerisation allows easy installation of models on users' machines or within secure research environments for use with private data. A caveat of this approach is that the user's machine must have containerisation software, such as Docker, installed.

7.2.4. Reproducibility of simulation studies in health

Finally, our framework is focused on supporting researchers to develop reusable models. Model reuse is related to the broader issue of reproducibility of model results and analyses. There has been a longstanding effort to support M&S researchers to make their computational work reproducible, that is, by fully reproducing published results using the same code and data (Monks et al., 2019; Taylor & Shao, 2018; Taylor et al., 2018), including the Association of Computing Machinery's (ACM) Reproducible Computational Results (RCR) initiative (ACM, 2023). Other efforts in this area include work on provenance patterns in simulation for ensuring the reproducibility and transparency of simulation studies, and for facilitating the reuse, adaptation, and execution of simulation experiments (Uhrmacher et al., 2023; Wilsdorf et al., 2022).

The STARS framework does not specifically guide, or provide a pipeline for, researchers to meet the ACM's RCR badge for "results – reproduced". However, STARS does support the production of

reusable models that are able to achieve “artefacts available”, “artefacts evaluated – functional” (essential components), and “artefacts evaluated – reusable” (optional components). These support reproducible healthcare results by enabling researchers and practitioners to publish a high quality version of their computer model alongside their journal articles or papers. The enhanced documentation of STARS is also easily integrated with the STRESS reporting guidelines (Monks et al., 2019) that are designed to improve transparency and reproducibility of simulation models. We have not made this an essential or optional component, but we include examples of its implementation in our applied case studies as part of the enhanced documentation.

We note that a healthcare simulation study with reproducible results does not automatically imply that a model artefact has been made publicly available. Such studies could receive an ACM badge “results – reproduced” by providing the model to a reviewer privately, e.g., by adding a reviewer to a private GitHub repository. Many of the essential and optional components of STARS support such studies to organise their simulation tools and artefacts so that they can easily be shared and run by reviewers; for example, with enhanced documentation and dependency management.

8. Conclusion

In healthcare, there are compelling reasons for developing open, reusable simulation models. These include providing opportunities for students and early career researchers to learn from models; to enhance visibility and potential collaborations; and to make models available to healthcare users such as analysts, managers and clinicians to preview, validate, and use.

Barriers and disincentives to open modelling include the additional time and effort required for the work; technical challenges associated with FOSS software and sharing models; and concerns about how models might be reused. Our STARS framework aims to support researchers to share FOSS models by providing a structured approach that enables a model to be accessible and reusable by others. Our worked examples demonstrate the generalisability and feasibility of the framework for reusable simulations.

STARS aligns with existing guidelines and reproducibility initiatives, including ACM RCR (ACM, 2023), and complements an RCR review by enabling artefacts to achieve three of the five ACM badges. We do not provide a pipeline for reproducibility (ACM “results – reproduced”), however STARS enables researchers and practitioners to adopt open science principles, which support efforts to improve the reproducibility of their work. Further work will

investigate how the framework can be used for M&S reproducibility, and will additionally test STARS with other FOSS languages such as R.

Notes

1. <https://www.openmodelingfoundation.org/stanards/reusability/>.
2. <https://choosealicense.com/>.
3. <https://docs.github.com/en>.
4. <https://about.gitlab.com/>.
5. <https://bitbucket.org/>.
6. [\(https://figshare.com/\)](https://figshare.com/).
7. [\(https://zenodo.org/\)](https://zenodo.org/).
8. [\(https://osf.io/\)](https://osf.io/).
9. [\(https://www.comses.net/\)](https://www.comses.net/).
10. [\(https://quarto.org/\)](https://quarto.org/).
11. [\(https://jupyterbook.org/\)](https://jupyterbook.org/).
12. <https://support.atlassian.com/bitbucket-cloud/docs/get-started-with-bitbucket-cloud/>.
13. <https://docs.gitlab.com/ee/user/project/pages/>.
14. <https://support.atlassian.com/bitbucket-cloud/docs/get-started-with-bitbucket-cloud/>.
15. <https://quartopub.com/>.
16. <https://colab.google/>.
17. <https://deepnote.com/>.
18. <https://paperswithcode.com>.

Acknowledgments

This work was supported by the National Institute for Health Research Applied Research Collaboration South West Peninsula. The views expressed in this publication are those of the author(s) and not necessarily those of the National Institute for Health Research or the Department of Health and Social Care.

Disclosure statement

No potential conflict of interest was reported by the author(s).

ORCID

Thomas Monks  <http://orcid.org/0000-0003-2631-4481>
 Alison Harper  <http://orcid.org/0000-0001-5274-5037>
 Navonil Mustafee  <http://orcid.org/0000-0002-2204-8924>

References

- ACM. (2023). *ACM artifact review and badging*. <https://www.acm.org/publications/policies/artifact-review-and-badging-current>. (Accessed: 2023-10-18).
- Allen, M., Bhanji, A., Willemsen, J., Monks, S., Logan, T., & Monks, T. (2020). A simulation modelling toolkit for organising outpatient dialysis services during the COVID-19 pandemic. *Public Library of Science ONE*, 15(8), e0237628. <https://doi.org/10.1371/journal.pone.0237628>
- Allen, C., & Mehler, D. M. (2019). Open science challenges, benefits and tips in early career and beyond. *PLoS biology*, 17(5), e3000246. <https://doi.org/10.1371/journal.pbio.3000246>
- Anagnostou, A., Groen, D., Taylor, S. J., Suleimenova, D., Abubakar, N., Saha, A., Mintram, K., Ghorbani, M.,

- Daroge, H., Islam, T., Xue, Y., Okine, E., & Anokye, N. (2022). FACS-CHARM: A hybrid agent-based and discrete-event simulation approach for covid-19 management at regional level. *Proceedings of the 2022 Winter Simulation Conference (WSC)*, Singapore (pp. 1223–1234). IEE.
- Ayllón, D., Railsback, S. F., Gallagher, C., Augusiak, J., Baveco, H., Berger, U., Charles, S., Martin, R., Focks, A., Galic, N., Liu, C., van Loon, E. E., Nabe-Nielsen, J., Piou, C., Polhill, J. G., Preuss, T. G., Radchuk, V., Schmolke, A., Stadnicka-Michalak, J., Thorbek, P., & Grimm, V. (2021). Keeping modelling notebooks with TRACE: Good for you and good for environmental research and management support. *Environmental Modelling & Software*, 136, 104932. <https://doi.org/10.1016/j.envsoft.2020.104932>
- Celi, L. A., Citi, L., Ghassemi, M., Pollard, T. J., & Mueck, L. A. (2019). The plos one collection on machine learning in health and biomedicine: Towards open code and open data. *Public Library of Science ONE*, 14(1), e0210232. <https://doi.org/10.1371/journal.pone.0210232>
- Chalk, D., Robbins, S., Kandasamy, R., Rush, K., Aggarwal, A., Sullivan, R., & Chamberlain, C. (2021). Modelling palliative and end-of-life resource requirements during covid-19: Implications for quality care. *BMJ Open*, 11(5), e043795. <https://doi.org/10.1136/bmjopen-2020-043795>
- Collberg, C., & Proebsting, T. A. (2016). Repeatability in computer systems research. *Communications of the ACM*, 59(3), 62–69. <https://doi.org/10.1145/2812803>
- Crowe, S., Grieco, L., Monks, T., Keogh, B., Penn, M., Clancy, M., Elkhodair, S., Vindrola-Padros, C., Fulop, N. J., & Utley, M. (2023). Here's something we prepared earlier: Development, use and reuse of a configurable, inter-disciplinary approach for tackling overcrowding in nhs hospitals. *Journal of the Operational Research Society*, 75(4), 689–704. <https://doi.org/10.1080/01605682.2023.2199094>
- Cubukcuoglu, C., Nourian, P., Sarıyıldız, I. S., & Tagetiren, M. F. (2020). A discrete event simulation procedure for validating programs of requirements: The case of hospital space planning. *SoftwareX*, 12, 100539. <https://doi.org/10.1016/j.softx.2020.100539>
- Dagkakis, G., & Heavey, C. (2016). A review of open source discrete event simulation software for operations research. *Journal of Simulation*, 10(3), 193–206. <https://doi.org/10.1057/jos.2015.9>
- den Eynden, V. V., Knight, G., Vlad, A., Radler, B., Tenopir, C., Leon, D., Manista, F., Whitworth, J., & Corti, L. (2016). Survey of wellcome researchers and their attitudes to open research. Project report No. 10.6084/m9. Figshare.4055448.v1. Wellcome Trust. Retrieved from <https://repository.essex.ac.uk/24866/>.
- Druskat, S., Spaaks, J. H., Chue Hong, N., Haines, R., Baker, J., Bliven, S., Willighagen, E., Pérez-Suárez, D., & Konovalov, A. (2021). Citation file format. Zenodo. Retrieved from <https://doi.org/10.5281/zenodo.5171937>
- Fletcher, A., & Worthington, D. (2009). What is a ‘generic’ hospital model?—a comparison of ‘generic’ and ‘specific’ hospital models of emergency patient flows. *Health Care Management Science*, 12(4), 374–391. <https://doi.org/10.1007/s10729-009-9108-9>
- Forbus, J. J., & Berleant, D. (2022). Discrete-event simulation in healthcare settings: A review. *Modelling*, 3(4), 417–433. <https://doi.org/10.3390/modelling3040027>
- Foster, E. D., & Deardorff, A. (2017). Open science framework (osf). *Journal of the Medical Library Association: JMLA*, 105(2), 203. <https://doi.org/10.5195/jmla.2017.88>
- Goldacre, B., Morton, C. E., & DeVito, N. J. (2019). Why researchers should share their analytic code. *BMJ: British Medical Journal (Online)*, 367, l6365. <https://doi.org/10.1136/bmj.l6365>
- Gomes, D. G., Pottier, P., Crystal-Ornelas, R., Hudgins, E. J., Foroughirad, V., Sánchez-Reyes, L. L., Turba, R., Martinez, P. A., Moreau, D., Bertram, M. G., Smout, C. A., & Gaynor, K. M. (2022). Why don't we share data and code? perceived barriers and benefits to public archiving practices. *Proceedings of the Royal Society B*, 289(1987), 20221113. <https://doi.org/10.1098/rspb.2022.1113>
- Hamilton, D. G., Hong, K., Fraser, H., Rowhani-Farid, A., Fidler, F., & Page, M. J. (2023). Prevalence and predictors of data and code sharing in the medical and health sciences: Systematic review with meta-analysis of individual participant data. *BMJ: British Medical Journal*, (382), <https://doi.org/10.1136/bmj-2023-075767>
- Harper, A., & Monks, T. (2023a). A framework to share healthcare simulations on the Web Using Free and Open Source Tools and Python. *Proceedings of SW23 The OR Society Simulation Workshop* (pp. 250–260). Retrieved from <https://www.theorsociety.com/media/7313/doior1036819sw23030.pdf>
- Harper, A., & Monks, T. (2023b). *Supplementary material: Code for the deployment of a simpy and streamlit model to dockerhub*. Zenodo. Retrieved from <https://doi.org/10.5281/zenodo.8011462>
- Harper, A., Monks, T., & Manzi, S. (2023). *Deploying healthcare simulation models using containerization and continuous integration*. OSF Preprints. Retrieved from osf.io/qez45
- Harper, A., Monks, T., Wilson, R., Redaniel, M. T., Eyles, E., & Jones, T. (2023). Development and application of simulation modelling for orthopaedic elective resource planning in england. *BMJ Open*, 13(12), e076221. <https://doi.org/10.1136/bmjopen-2023-076221>
- Hossain, S. (2019). Visualization of bioinformatics data with Dash Bio. *Proceedings of the 18th Python in Science Conference*, Austin, Texas (pp. 126–133). In C. Calloway, D. Lippa, D. Niederhut, & D. Shupe (Eds.).
- Hrynaszkiewicz, I., Harney, J., & Cadwallader, L. (2021). *A survey of code sharing practice and policy in computational biology*. OSF Preprints. Retrieved from <https://osf.io/f73a6>
- Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3), 90–95. <https://doi.org/10.1109/MCSE.2007.55>
- Janssen, M. A., Pritchard, C., & Lee, A. (2020). On code sharing and model documentation of published individual and agent-based models. *Environmental Modelling & Software*, 134, 104873. <https://doi.org/10.1016/j.envsoft.2020.104873>
- JupyterLab. (2022). JupyterLab. Author. Retrieved from <https://github.com/jupyterlab/jupyterlab> (Accessed: 2023-10-18).
- Knight, V., & Palmer, G. (2022). *Applied mathematics with open-source software*. Chapman & Hall/CRC Series in Operations Research.
- Kucharský, Š., Houtkoop, B.-L., & Visser, I. (2020). *Code sharing in psychological methods and statistics: An overview and associations with conventional and alternative research metrics*. OSF preprints. Retrieved from <https://osf.io/daews/download>
- Lim, C. Y., Bohn, M. K., Lippi, G., Ferrari, M., Loh, T. P., Yuen, K.-Y., Adeli, K., & Horvath, A. R. (2020). Staff rostering, split team arrangement, social distancing

- (physical distancing) and use of personal protective equipment to minimize risk of workplace transmission during the COVID-19 pandemic: A simulation study. *Clinical Biochemistry*, 86, 15–22. <https://doi.org/10.1016/j.clinbiochem.2020.09.003>
- Lin, D., Crabtree, J., Dillo, I., Downs, R. R., Edmunds, R., Giaretta, D., De Giusti, M., L'Hours, H., Hugo, W., Jenkyns, R., Khodiyar, V., Martone, M. E., Mokrane, M., Navale, V., Petters, J., Sierman, B., Sokolova, D. V., Stockhouse, M., & Westbrook, J. (2020). The TRUST principles for digital repositories. *Scientific Data*, 7(1), 144. <https://doi.org/10.1038/s41597-020-0486-7>
- McKiernan, E. C., Bourne, P. E., Brown, C. T., Buck, S., Kenall, A., & Lin, J. (2016). How open science helps researchers succeed. *eLife*, 5, e16800. <https://doi.org/10.7554/eLife.16800>
- McKinney, W. (2011). Pandas: A foundational python library for data analysis and statistics. *Python for High Performance and Scientific Computing*, 14(9), 1–9.
- Monks, T., Currie, C. S., Onggo, B. S., Robinson, S., Kunc, M., & Taylor, S. J. (2019). Strengthening the reporting of empirical simulation studies: Introducing the STRESS guide-lines. *Journal of Simulation*, 13(1), 55–67. <https://doi.org/10.1080/1747778.2018.1442155>
- Monks, T., & Harper, A. (2023a). Computer model and code sharing practices in healthcare discrete-event simulation: A systematic scoping review. *Journal of Simulation*. Retrieved from 1–16. <https://doi.org/10.1080/1747778.2023.2260772>
- Monks, T., & Harper, A. (2023b). Improving the usability of open health service delivery simulation models using python and web apps. *NIHR Open Research*, 3, 48. Retrieved from <https://doi.org/10.3310/nihropres.13467.1>
- Monks, T., & Harper, A. (2023c). SimPy and StreamLit tutorial materials for healthcare discrete-event simulation. *Zenodo*. Retrieved from <https://doi.org/10.5281/zenodo>
- Monks, T., & Harper, A. (2023d). Towards sharing tools, and artifacts, for reusable simulation: A minimal model example. *Zenodo*. Retrieved from <https://zenodo.org/records/10026327>
- Monks, T., & Harper, A. (2023e, October). Towards sharing tools and artifacts for reusable simulation: Deploying a ‘simpy’ model as a web app. *Zenodo*. Retrieved from <https://doi.org/10.5281/zenodo.10055169>
- Monks, T., & Harper, A. (2023f). Towards sharing tools and artifacts for reusable simulation: Example enhanced documentation for a ‘simpy’ model. *Zenodo*. Retrieved from <https://doi.org/10.5281/zenodo.10054063>
- Monks, T., & Harper, A. (2023g). Towards sharing tools, artifacts, and reproducible simulation: A ciw model exemplar. *Zenodo*. Retrieved from <https://doi.org/10.5281/zenodo.10051495>
- Moreau, D., Wiebels, K., & Boettiger, C. (2023). Containers for computational reproducibility. *Nature Reviews Methods Primers*, 3(1), 50. <https://doi.org/10.1038/s43586-023-00236-9>
- Munafò, M. R., Nosek, B. A., Bishop, D. V. M., Button, K. S., Chambers, C. D., Percie du Sert, N., Simonsohn, U., Wagenmakers, E. J., Ware, J. J., & Ioannidis, J. P. A. (2017). A manifesto for reproducible science. *Nature Human Behaviour*, 1(1), 0021. Retrieved from <https://doi.org/10.1038/s41562-016-0021>
- Nature Biotechnology. (2019). Changing coding culture. *Nature Biotechnology*, 37(5), 485–485. <https://doi.org/10.1038/s41587-019-0136-9>
- Nelson, B. (2013). *Foundations and methods of stochastic simulation: A first course*. Springer.
- OMF. (2023). *Omfl: The modeling foundation*. Retrieved from <https://www.openmodelingfoundation.org/stanards/reusability/> (Accessed: 2023-10-18).
- Palmer, G., Knight, V. A., Harper, P. R., & Hawa, A. L. (2019). Ciw: An open-source discrete event simulation library. *Journal of Simulation*, 13(1), 68–82. <https://doi.org/10.1080/1747778.2018.1473909>
- Palmer, G., & Tian, Y. (2021). *Source code for ciw hybrid simulations*. *Zenodo*. Retrieved from <https://doi.org/10.5281/zenodo.4601529>
- Palmer, G., & Tian, Y. (2023). Implementing hybrid simulations that integrate DES + SD in python. *Journal of Simulation*, 17(3), 240–256. <https://doi.org/10.1080/1747778.2021.1992312>
- Philip, A. M., Prasannavenkatesan, S., & Mustafee, N. (2022). Simulation modelling of hospital outpatient department: A review of the literature and bibliometric analysis. *Simulation*, 0(), 00375497221139282. Retrieved from <https://doi.org/10.1177/00375497221139282>
- Piwowar, H. A., & Vision, T. J. (2013). Data reuse and the open data citation advantage. *PeerJ Life & Environment*, 1, e175. Retrieved from <https://doi.org/10.7717/peerj.175>
- Pouwels, X. G., Sampson, C. J., Arnold, R. J., Janodia, M. D., Henderson, R., & Lamotte, M., ... Nuijten, M. (2022). Opportunities and barriers to the development and use of open source health economic models: A survey. *Value in Health*, 25(4), 473–479.
- quarto.org. (2023). *Welcome to quarto® an open-source scientific and technical publishing system* Retrieved from <https://quarto.org/> (Accessed: 2023-10-18.)
- Robinson, S. (2005). Discrete-event simulation: From the pioneers to the present, what next? *Journal of the Operational Research Society*, 56(6), 619–629. <https://doi.org/10.1057/palgrave.jors.2601864>
- Roy, S., Venkatesan, S. P., & Goh, M. (2021). Healthcare services: A systematic review of patient-centric logistics issues using simulation. *Journal of the Operational Research Society*, 72(10), 2342–2364. <https://doi.org/10.1080/01605682.2020.1790306>
- RStudio. (2023). *Introduction to Renv*. <https://rstudio.github.io/renv/articles/renv.html>. (Accessed: 2024-08-01.)
- Salleh, S., Thokala, P., Brennan, A., Hughes, R., & Booth, A. (2017). Simulation modelling in healthcare: An umbrella review of systematic literature reviews. *PharmacoEconomics*, 35(9), 937–949. <https://doi.org/10.1007/s40273-017-0523-3>
- Salmon, A., Rachuba, S., Briscoe, S., & Pitt, M. (2018). A structured literature review of simulation modelling applied to emergency departments: Current patterns and emerging trends. *Operations Research for Health Care*, 19, 1–13. <https://doi.org/10.1016/j.orhc.2018.01.001>
- Samota, E. K., & Davey, R. P. (2021). Knowledge and attitudes among life scientists toward reproducibility within journal articles: A research survey. *Frontiers in Research Metrics and Analytics*, 6, 678554. <https://doi.org/10.3389/frma.2021.678554>
- Serghiou, S., Contopoulos-Ioannidis, D. G., Boyack, K. W., Riedel, N., Wallach, J. D., Ioannidis, J. P., & Bero, L. (2021). Assessment of transparency indicators across the biomedical literature: How open is open? *PLoS Biology*, 19(3), e3001107. <https://doi.org/10.1371/journal.pbio.3001107>
- shinyapps.io. (2023). *Build interactive web applications easily with the power of python’s data and scientific stack*. Retrieved from <https://shiny.posit.co/py/> (Accessed: 2023-10-18.)

- SimPy, T. (2020). *Simpy*. <https://simpy.readthedocs.io/en/latest/index.html>. (Accessed: 2023-10-18).
- Smith, A. M., Katz, D. S., Niemeyer, K. E., & FORCE11 Software Citation Working Group. (2016). Software citation principles. *PeerJ Computer Science*, 2, e86. Retrieved 2023-04-25, from <https://doi.org/10.7717/peerj-cs.86> <https://peerj.com/articles/cs-86>
- Smith, R., Mohammed, W., & Schneider, P. (2023). Packaging cost-effectiveness models in r: A tutorial. *Wellcome Open Research*, 8(419), 419. <https://doi.org/10.12688/wellcomeopenres.19656.1>
- Smith, R., & Schneider, P. (2020). Making health economic models shiny: A tutorial. *Wellcome Open Research*, 5(69), 69. <https://doi.org/10.12688/wellcomeopenres.15807.2>
- Smith, R., Schneider, P., & Mohammed, W. (2022). Living HTA: Automating health economic evaluation with R. *Wellcome Open Research*, 7(194), 194. <https://doi.org/10.12688/wellcomeopenres.17933.2>
- streamlit.io. (2022). *Streamlit. The fastest way to build and share data apps*. <https://streamlit.io/>. (Accessed: 2023-10-18).
- Taylor, S. J., Anagnostou, A., Fabiyi, A., Currie, C., Monks, T., Barbera, R., & Becker, B. (2017). Open science: Approaches and benefits for modeling & simulation. *Proceedings of 2017 Winter Simulation Conference (WSC)*, Las Vegas, NV, USA (pp. 535–549). IEEE.
- Taylor, S. J., Eldabi, T., Monks, T., Rabe, M., & Uhrmacher, A. M. (2018). Crisis, what crisis—does reproducibility in modeling & simulation really matter? *Proceedings of 2018 winter simulation conference (WSC)*, Gothenburg, Sweden (pp. 749–762). IEEE.
- Taylor, S. J., & Shao, G. (2018). Simulation standards and reproducibility: Issues in simulation standards. *Proceedings of the 2018 winter simulation conference*, Gothenburg, Sweden, IEEE.
- The Turing Way Community. (2022). *The turing way: A handbook for reproducible, ethical and collaborative research*. Zenodo. Retrieved from <https://doi.org/10.5281/zenodo.7470333> (Accessed: 2023-10-18).
- Uhrmacher, A., Frazier, P., Hahnle, R., Klugl, F., Lorig, F., Ludascher, B., Bertram, N., Laura, R. M., Cristina, R., Bernhard, S., Gabriel, A., & Wilsdorf, P. (2023). Context, composition, automation, and communication – the c2ac roadmap for modeling and simulation. *arXiv Preprint arXiv: 1503.02531* 2. Retrieved from <https://doi.org/10.48550/arXiv.2310.05649>
- van der Walt, S., Colbert, S. C., & Varoquaux, G. (2011). The numpy array: A structure for efficient numerical computation. *Computing in Science & Engineering*, 13 (2), 22–30. <https://doi.org/10.1109/MCSE.2011.37>
- Vázquez-Serrano, J. I., Peimbert-García, R. E., & Cárdenas-Barrón, L. E. (2021). Discrete- event simulation modeling in healthcare: A comprehensive review. *International Journal of Environmental Research and Public Health*, 18 (22), 12262. <https://doi.org/10.3390/ijerph182212262>
- Wilsdorf, P., Heller, J., Budde, K., Zimmermann, J., Warnke, T., Haubelt, C., Timmermann, D., van Rien, U., & Uhrmacher, A. M. (2022). A model-driven approach for conducting simulation experiments. *Applied Sciences*, 12(16), 7977. <https://doi.org/10.3390/app12167977>
- Wojciechowski, J., Hopkins, A. M., & Upton, R. N. (2015). Interactive pharmacometric applications using r and the shiny package. *CPT: Pharmacometrics & Systems Pharmacology*, 4(3), 146–159. <https://doi.org/10.1002/psp4.21>
- Zečević, K., Houghton, C., Noone, C., Lee, H., Matvienko-Sikar, K., & Toomey, E. (2021). Exploring factors that influence the practice of open science by early career health researchers: A mixed methods study. *HRB Open Research*, 3, 56. <https://doi.org/10.12688/hrbopenres.13119.2>
- Zhang, X. (2018). Application of discrete event simulation in health care: A systematic review. *BMC Health Services Research*, 18(1), 687. <https://doi.org/10.1186/s12913-018-3456-4>
- Zhang, X., Lhachimi, S. K., & Rogowski, W. H. (2020). Reporting quality of discrete event simulations in healthcare—results from a generic reporting checklist. *Value in Health*, 23(4), 506–514. <https://doi.org/10.1016/j.jval.2020.01.005>
- Zou, C., Wang, J., & Cheng, Y. (2022). The impact of the variability of patient flow and service time on the efficiency of large-scale outpatient systems. *IEEE Transactions on Computational Social Systems*, 10(3), 1230–1240. <https://doi.org/10.1109/TCSS.2021.3137930>