Leiden University

ICT in Business and the Public Sector

Developing a Maturity Model for Computational Reproducibility in Health Data Science

Name:	Wessel van Putten
Student ID:	s2600889
Date:	24/09/2025
Supervisor 1:	Dr. A.E.J.L Lefebvre
Supervisor 2:	Prof.dr. M.R. Spruit

MASTER'S THESIS

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Einsteinweg 55
2333 CC Leiden
The Netherlands

Abstract

Computational reproducibility describes to what extent software that is used for data analysis can be used by other analysts while receiving the same output. Although computational reproducibility is seen as a very important element in both scientific research as in corporate environments, it often lacks in practice. This leads to analysis that can not be repeated and rising costs. This thesis mainly focuses on the health sector and especially on data analysis on health data. In this sector, bad reproducibility can cause a delay in inventing new medicine, which may harm the health of people.

The research follows a design science approach, combining both literature research and interviews. The interviews will follow the principle of a semi-structured interview. The respondents from these interviews come from two organizations that fit well in our scope, namely Zorg en Zekerheid and Leiden University Medical Center (LUMC). Zorg en Zekerheid is a health insurance company based in Leiden and has roughly 500 employees. Their respondents work as consultant, data analyst or data engineer. The other organization that our respondents come from is LUMC. This is an academic hospital that is linked with Leiden University. Most respondents come from the health campus of LUMC where they perform research on population health. After the respondents have been interviewed, the interviews were coded, and prioritized using the MoSCoW method. By doing this, we gained insight on which elements are important for computational reproducibility. Those elements could then be part of the maturity model.

Based on the literature research on maturity models and the insights on what influences computational reproducibility, a Focus Area Maturity Model (FAMM) turned out to match our desires. In contrast to other types of maturity models, a FAMM consists of multiple focus areas, which itself contain several capabilities. Those capabilities are spread across an axis of maturity level, where each new capability that is reached is placed higher on the scale of maturity. A FAMM can be displayed with focus areas on the vertical axis and maturity levels on the horizontal axis. The capabilities are then placed on the intersections of the axes. Important focus areas are documentation, version control, data manipulation and data sharing. After developing the maturity model, with the focus areas and capabilities placed on the model, it was evaluated during a focus group with respondents from Zorg en Zekerheid again. During this sessions, respondents used the model to evaluate their organizations maturity level and they were given the exercise to move capabilities. Furthermore, they could remove and add capabilities or focus areas. This lead to a slightly adapted maturity model, containing 14 focus areas and 57 capabilities.

The data analysis that was performed on the interviews showed that the importance of several focus areas and capabilities differ for the two different organizations. In general, the Zorg en Zekerheid employees were more focused on agreements and internal processes, whereas the LUMC researchers had their focus on how the sensitive data could be shared in order to stimulate computational reproducibility.

This thesis contributes to both companies and researchers that work with sensitive health data by delivering a maturity model on computational reproducibility. The model

proposes several steps that can be taken to become more mature on several focus areas. The model is made using input from respondents in the health sector, but can also be used on other fields where computational reproducibility is required. Further research on this topic could expand the number of organizations and perform a larger validation process. Another improvement that may be valuable is to explore if focus areas should be given a different weight, depending on the type of organization that is using the maturity model.

Acknowledgments

I would like to thank my supervisor Armel Lefebvre for guiding me through the final stages of my master studies. He introduced me to this research topic, helped with contacting respondents for the interviews and was always available to provide me with help and feedback on my work.

Secondly, I would like to thank Marco Spruit. After contacting him in December, he brought me in contact with Armel and offered to be my second supervisor, where he provided me useful feedback on both my research proposal and my final thesis.

In addition to thanking my supervisors, I would like to express that I am very grateful that Zorg en Zekerheid offered me the ability to work as a data analyst during my master studies. During my work at Zorg en Zekerheid, I have learned several skills that enabled me to perform the research that I have done for this thesis. After my studies I will continue working for Zorg en Zekerheid on full time basis, for which I am grateful as well.

Furthermore, I would like to thank all the respondents that have made time and effort to be available for the interviews that were held during my thesis. Their input was very valuable and made a big impact on the final deliverable.

Lastly, I would like to mention that I could not have finalized this journey without the support of my friends and family, who have always showed interest in my work and progress and supported me through my thesis.

Contents

1 Introduction

Computational reproducibility describes how software, which is used for research or data analysis, is reproducible for others. Computational reproducibility means that while someone is using the same code and data as originally used, this will lead to the same output (Lakens, 2016). Since a lot of research is built on software, computational reproducibility is beneficial to scientific progress. According to the results of a survey conducted by Piccolo and Frampton (2016), 92% of scientists in the United Kingdom use software for their research. Reproducibility itself is important for research to be reliable (McNutt, 2014). Scientists reproduce others work to confirm that their findings are reliable indeed (McNutt, 2014). An additional advantage of computational reproducibility is a cost reduction (Sandve et al., 2013). This is due to the fact that reproducible code requires less time to run successful as there will be no errors about missing data or libraries. This leads to a decrease of time required for data analysis, which may then lead to cost reduction. Furthermore, shorter data analysis times increase efficiency.

In addition to scientists, companies are increasingly using software code to perform data analysis as well. In a business context, reproducible code ensures that teams can work more efficiently together as they can share their code without having to adjust a lot of code, like changing paths to input files or having to install libraries manually. This is for instance important when multiple colleagues collaborate on a project or when a colleague is absent. This is not only important when other employees work on someone's code or in case of absence of a team member, but also for the programmer themselves. Reproducibility is also beneficial when creating new code as it can reduce the time programmers need to dive into their own code.

1.1 Problem statement

Despite the advantages and importance of computational reproducibility, research results are often not reproducible. Based on a survey from Baker (2016) published in Nature, 70% of researchers fail to reproduce other's work and 90% of scientists acknowledge that there is a reproducibility crisis. According to Freedman et al. (2015), approximately 114.8 billion dollars per year is spent on life sciences, from which $56.4 billion per year is spent preclinical research. The article mentions that the irreproducibility rate lies between 18% and 88.5%. Freedman et al. (2015) write that calculating with an irreproducibility rate of 50%, the outcome is that $28 billion dollars is annually spent on preclinical research that is not reproducible. However, it is important to note that this does not mean that there was no return on this invested money (Freedman et al., 2015) and not this whole amount of money has become worthless. Freedman et al. (2015) concludes that lifesaving treatments are delayed, research budgets face more pressure and the costs of treatments and drug development are increasing. They state that increasing reproducibility helps in solving these issues. Issues with reproducibility are not limited to the scientific field, as companies experience them as well. When focusing on the healthcare sector, companies that perform data analysis on healthcare data are health insurance companies. The Netherlands has several of these health insurance companies, which have data about medical treatments which their customers receive by health professionals. That data can be used to check whether patients received the right treatment and if that treatment was justified. Other applications of this data is to analyze where there could be spent more effort on preventing health issues. Zorg en Zekerheid is one of those insurance companies. They are based at Leiden and have roughly 500 employees. Their market cap in The Netherlands is around 3%, of which most customers live in the region around Leiden. Issues regarding computational reproducibility that Zorg en Zekerheid might experience are, for example, when a data analyst who has developed some code leaves the company or is absent as their code is not always computational reproducible. Investing effort in improving processes that influence computational reproducibility may help to avoid these problems, because when a colleague is absent of leaves, other colleagues are able to work with the software when it is reproducible. To be able to improve those processes, Zorg en Zekerheid should firstly identify which areas are important for computational reproducibility. Secondly, it is important to know at which areas improvement will be effective and lastly, they must know how they can improve at those areas. A maturity model that evaluates whether code or research is computational reproducible enough may aid in solving these problems. Such a model can be used to check if the piece of software meets the standards that are set by an organization or society.

1.2 Research Questions

In this thesis, we investigate the following research question:
"How can a maturity model aid in improving computational reproducibility in health data analysis?"
In order to answer this question, some sub-questions should be answered first:

RQ 1 *What are currently encountered issues when trying to reproduce other's work that is build to analyze health data?*

To know how to improve computational reproducibility, it is important to identify current challenges. Examples of those issues can be code that can not be understood or missing data. Those examples may lead to a longer time that is required to work with the code or the code giving a wrong or invalid output.

RQ 2 *What are the components and capabilities that strengthen computational reproducibility?*

Where the previous research question focuses on what current challenges, this research question focuses on what contributes to computational reproducibility. This means that the fundamentals of computational reproducibility will be researched and identified such that required processes and elements of the processes that improve computational reproducibility can be formed.

RQ 3 *What should a maturity model contain in order to measure computational reproducibility of software processing health data?*

A maturity model is used to measure maturity of an organization or processes that an organization performs. After measuring the maturity, the model can be used to improve on those processes. To develop such a model, firstly the capabilities and focus areas should be defined. That is done to answer the previous research questions. Those focus areas describe areas where effort should be invested to improve computational reproducibility, where the capabilities display steps towards more maturity. Furthermore, it should be clear how much maturity each of the capabilities adds to the focus area it belongs to. That is done to design the different steps that should be taken to gain more maturity and to identify how much maturity these steps add to certain processes.

1.3 Thesis outline

This thesis is divided in different sections. Section 2 describes scientific literature that will be used as the foundation of this thesis. Firstly, this section describes literature about computational reproducibility. This focuses on the importance of computational reproducibility and how can be achieved. This also includes handling sensitive data while making data analysis reproducible. Furthermore, Section 2 describes literature about how maturity models can be used to improve processes within an organization. Different types of maturity models will be discussed, followed by the process of creating a maturity model. After the literature review, Section 3 is about the methodology, introducing the design science, which is the method of this research. Furthermore, the methodology describes how data was gathered from the interviews and how that data was processed. This section also shows which steps were followed to design the maturity model, following the approach that has been learned from the literature review. Section 4 shows the results of performing the steps that are described in the methodology. This leads to this section containing the findings from the interviews and compares Zorg en Zekerheid with LUMC. Furthermore, this section presents the final maturity model. Section 5 sums up the most important findings from this research and compares our model with existing models and suggests possible improvements. Furthermore, our model is validated in this section. Finally, Section 6 concludes the thesis and summarizes the findings.

2 Literature

2.1 Literature review

This section explores various literature about our topic. Firstly relevant literature about computational reproducibility is discussed. This literature mainly describes the core components of computational reproducibility and shows the important factors that should be present to make software reproducible. This section scopes on how the software should be written and updated and shows how code can be shared such that others can work with it properly. Besides focusing on the code, data management is also a topic that is explained. Besides computational reproducibility, maturity models are explained here. The section about maturity models firstly discusses multiple types of maturity models and the specific characteristics of those models. It also discusses specific models related to our research topic. After discussing this, a methodology for creating a maturity model for computational reproducibility is proposed.
The literature that this research relies on comes from both Google Scholar and the online Leiden University Library. Both are using search engines to find literature based on the keywords that the user requests. Keywords that have been used to find literature that is discussed in this section are related to computational reproducibility, maturity model, FAMM, FLMM, data analytics reproducibility, documentation, version control and combinations of those keywords. As a search query returned multiple articles, the priority was mostly given to those with a higher amount of citations.

2.2 Computational Reproducibility

When research is reproducible, it means that other researchers get the same results as the original research when following the same methodology. 90% of scientists acknowledge that there is a reproducibility crisis (Baker, 2016). This leads to an increase in research costs and a delay in new inventions (Freedman et al., 2015). That is among other things due to the fact that poor reproducibility slows down other research as it is harder to reuse parts of previous research (Grüning et al., 2018). Depending on which type of research is being performed, the demands of reproducibility may differ. When research consists of a data analysis, for example, that analysis should be computational reproducible. That means that someone who is using the same data and code as in the original research, the output of the code will be the same (Lakens, 2016). Ziemann et al. (2023) have defined five pillars of computational reproducibility, as shown in Figure 1.

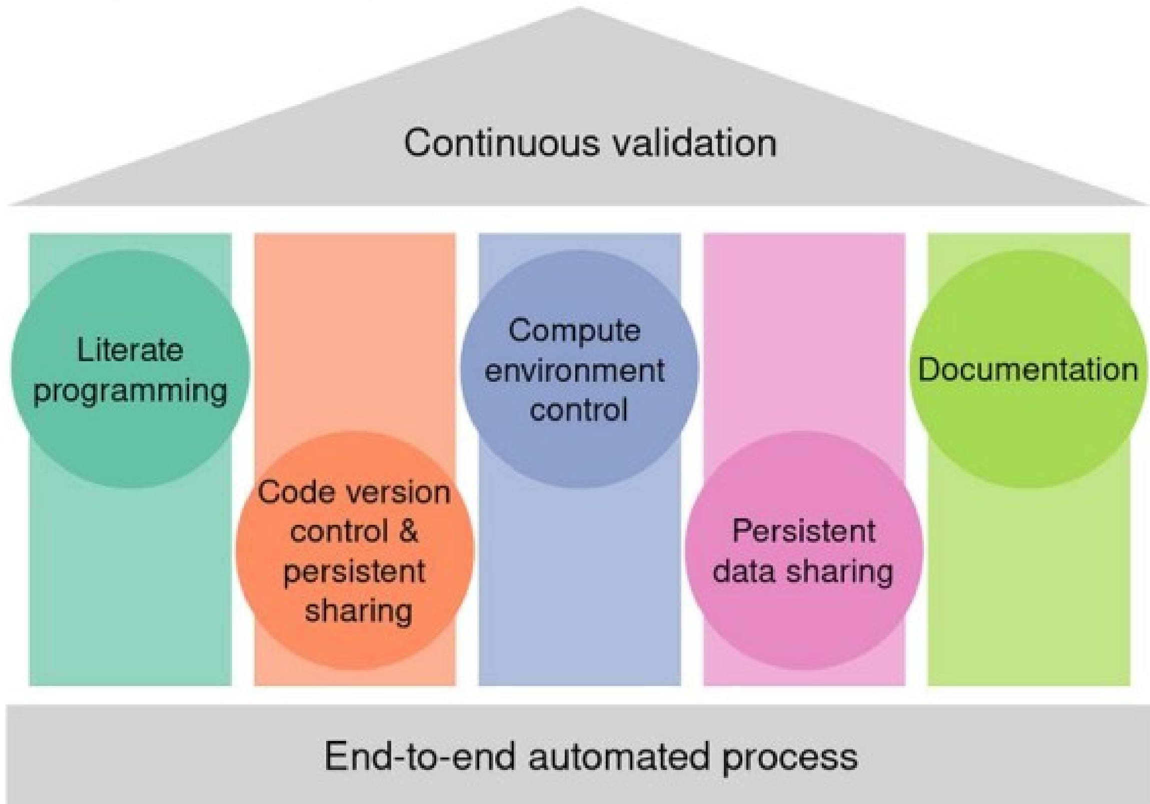

Figure 1: The five pillars of computational reproducibility (Ziemann et al., 2023)

The base consists of end-to-end automated process. This means that workflows should be automated, because this removes manual steps which are prone to errors and time consuming (Ziemann et al., 2023). Furthermore, automated processes improve auditing and makes reproduction easier. Ziemann et al. (2023) mentions that even in automated processes, the programmer should be cautious as randomness should be avoided and replaced by a fixed seed. Office automation was seen as the holy grail in the 1970's (Stohr & Zhao, 2001). As workflow automation removes the urge to perform time-consuming tasks manually, efficiency will receive a boost. According to Zayas-Cabán et al. (2022), the health care domains that can be automated are administration, operation, treatment and care delivery, population health, reporting, surveillance and analytics. The first pillar that is shown in Figure 1 is Literate programming. This concept combines pieces of codes with "human-readable text" (Ziemann et al., 2023). Knuth (1984) writes that programs are more than instructions for computers and that a program should explain to people what the computer does. Literate programming is introduced as a moral statement by Knuth (1984), as he mentions that nobody should be willing to write an illiterate program. Furthermore, Knuth (1984) mentions that this way of programming leads to better programming, as programmers have to be more careful when they write their program and that it leads to more efficiency. He states that writing a literate program may be more time-consuming, but understanding the program will be done more quickly. Compiling this leads to a document that may contain computer output like calculations, tables and figures in combination with textual explanations (Ziemann et al., 2023). According to Ziemann et al. (2023), literate programming leads to quicker report creation as the output is directly in a report format, more flexibility and code error free reports. Furthermore, Kery et al. (2018) state that through literate programming, analyses are more understandable as code, comments and output are all in one file. When an analysis is literate programmed, the whole process is visible, which stimulate trust and reproducibility (Kery et al., 2018). The second pillar describes code version control and persistent sharing which will be described respectively in Section 2.2.2 and Section 2.2.3. Pillar 3, compute environment control, focuses on reporting the version and environment where the analysis was made in. Ziemann et al. (2023) advises this because software updates may change the way that code is being executed. To save time in the future, Hurley et al. (2015) even goes a step further, by proposing to use virtual environments where the original environment is simulated. The article states that reproducibility is hindered because software is dependent on tools, libraries or operating systems and small differences with the original environment can change the output of return an error. To avoid this, Hurley et al. (2015) advises to publish output together with a virtual environment, such that all dependencies that were present in the original environment are available for other researchers as well. However, due to licenses and hardware requirements, this is not always as easy (Hurley et al., 2015). Ziemann et al. (2023) also refers to virtual environments. They present it as an alternative to just documenting the used versions and can be considered as more advanced and optional. The persistent data sharing pillar is all about the importance of sharing data for open science and how that data can be shared. Section 2.2.3 elaborates on this further. Pillar five, which is about documentation is covered in Section 2.2.1. Above the pillars in Figure 1, we see continuous validation. Ziemann et al. (2023) use this to describe that the principles in the pillars should be tested regularly alongside with the code itself. If this is not done frequently enough, errors with computational reproducibility may come up. Moreover, Sandve et al. (2013) have set up multiple rules to ensure computational reproducibility in research. Those rules can be categorized in documentation, version control, data management and standardization. Combining the articles of Sandve et al. (2013) and Ziemann et al. (2023) may set a good foundation for understanding how computational reproducibility can be achieved. To dive deeper in measures that stimulate computational reproducibility, the next sections focus on the rules that Sandve et al. (2013) has written about and some of the pillars that are reported in the article of Ziemann et al. (2023), but are not yet discussed.

2.2.1 Documentation

Documentation is a very important concept and can be seen as one of the main pillars of computational reproducibility (Ziemann et al., 2023). Computational reproducibility is improved if a code repository has a manual. According to Ziemann et al. (2023), this manual, or so called README, should contain the reason or purpose of the code and why it is relevant for the performed research. Besides that, the same article states that the README should contain requirements, instructions and all other things that are required to run the code and thus reproduce the outcomes. Furthermore, the documentation should contain several hard- and software requirements like the minimum RAM, CPU and GPU, but also the used operating system and dependencies should be present in the README. Lastly, Ziemann et al. (2023) mention that the documentation should contain clear instructions which describe how to contribute to the repository and how to reproduce the outcomes. It should also be clear which research article the code belongs to and how to contact developers for reporting issues. Sandve et al. (2013) write that for each result, it should be documented how that result was achieved as that makes clear which steps were performed. If a result was achieved by a computer program, it is important to document which program it was and which parameters have been used. Sandve et al. (2013) state that it is important to watch out with manually noting the taken steps as the documentation can easily become outdated when comparing it with the code. Sometimes an analysis include randomness, for example when making predictions. Those algorithms can give slightly different results for each execution. To solve that issue, it is often possible to use a random seed while calling the function. To make sure that the code is reproducible, this seed should also be part of the documentation (Sandve et al., 2013). According to Cook and Visconti (1994), bad documentation makes it harder for other developers to work with the code and running, and thus reproducing, the code delivers problems. Cook and Visconti (1994) have created a maturity model concerning software documentation and their model showed that organizations that have a higher maturity on documentation are able to correct mistakes in the code earlier and thus have less issues when the code is published. This leads to a better computational reproducibility.

2.2.2 Version Control

Another important concept that is used to improve computational reproducibility is Version Control. This is because small adjustments in a script can cause a total different output (Sandve et al., 2013). A different script may generate a different output even if the input data and parameters stay the same. To make sure that previous results can be retrieved, Sandve et al. (2013) emphasizes the importance of archiving different versions of the code. Version control systems that are used a lot are Subversion, Git and Mercurial of which Git is used the most among programmers (Kappagantula & Fds, 2025).This is mainly due to its powerful features and efficiency (Ziemann et al., 2023) If these tools are not used, the minimum version control that should be used is to save copies of a script every now and then (Sandve et al., 2013). According to Zolkifli et al. (2018), a version control system like Git keeps track of all changes, such that managing and editing code becomes easier with multiple developers. Zolkifli et al. (2018) mentions the difference between a centralized version control system, which contains one central repository, and a distributed version control system, where every user has an own repository. In the latter case, working offline is possible, enabling developers to collaborate on location.

2.2.3 Persistent code and data sharing

To make research computational reproducible, it is important to share both code and data , because it shows quality and trustworthiness (Kowalczyk & Shankar, 2011). After sharing the data and code, the publisher should be able to respond on requests about the data and methodology. According to Sandve et al. (2013), this increases the chances of being cited by other researchers. Ziemann et al. (2023) also admits that sharing code improves the amount of citations. Sharing code can easily be done with the help of a Version Control system like Git as, depending on the repository settings, repository is publicly available. Kowalczyk and Shankar (2011) states that data is too often not published. This can be due to privacy issues, but Kowalczyk and Shankar (2011) state that a reason for researcher to not share data as they fear that people that are in the sample will not appreciate it if others use the data they have contributed to. Furthermore, institutions are afraid of others using their data without having made any effort in collecting data. Besides sharing code, Ziemann et al. (2023) also urges for the importance of sharing data as without it, research is not computational reproducible. To make sure that the data is shared effectively, Kowalczyk and Shankar (2011) state that the data should be persistent available, which means that it should not be removed and stay accessible for a longer period. In addition to that the data has to be usable and reliable (Kowalczyk & Shankar, 2011). Other principles that shared data should adhere to are the FAIR principles (Jati et al., 2022). FAIR data means that data should be Findable, Accessible, Interoperable and Reusable. Accessibility in terms of FAIR data means that data should be accessible for those who should be able to have access, whereas (Jati et al., 2022) describes open data as data that is available for anyone. Therefore FAIR data is a better option to store privacy prone data, like health data. Furthermore, FAIR data supports that data is reusable which is very important to improve computational reproducibility, as mentioned by both Jati et al. (2022). To ensure reproducibility of health data, a good practice is to store it according to the FAIR guidelines. (Jati et al., 2022) explains the differences between open data and FAIR data. Open data means that the data is accessible for everyone. As health data is very personal data, privacy is an issue. That means that storing health data as open data is not a possibility. There are several ways to share this sensitive data without harming the privacy of people in the dataset. First of all, sharing the code that is used for the analysis should always be shared (Shepherd et al., 2017). This code should not contain any personal information and sharing it will not harm privacy. According to Shepherd et al. (2017), all manipulations should be made in this code, such that it is clear how the input data structure should be. In addition to that, simulated or synthetic data should be provided with the code. This synthetic data should have the same structure as the original data, such that the original code works flawless with it. Thirdly Shepherd et al. (2017) write that it is required to share the output of the original code with the simulated data, such that the research can be reviewed. By doing this, research on health data becomes ”quasi-reproducible” (Shepherd et al., 2017). In order to make this quasi-reproducible research even more reproducible, the synthetic data should have the same statistical properties as the original data (Gonzales et al., 2023).

2.2.4 Data Manipulation

In addition to the pillars, the way of which data is manipulated impacts computational reproducibility. When data has to be manipulated to perform an analysis, the preferred way to do this should be inside the program that is written instead of manually or in an external program. (Sandve et al., 2013). This both reduces to chance of errors and makes the data manipulation reproducible. That means that an analysis should start by reading in a raw dataset (Cooper et al., 2017). After that, Cooper et al. (2017) write that ”Any cleaning, merging, transforming, etc. of data should be done in scripts, not manually.”. However, Sandve et al. (2013) mentions that if it is not possible to avoid manual steps, there should at least be documentation of which data was modified and why it was done.

In the end-to-end automated process principle, Ziemann et al. (2023) emphasizes that spreadsheets are sensitive to data manipulation errors. Even using a hybrid form of data manipulation between spreadsheets and scripts is not recommended due to inefficiency and errors (Ziemann et al., 2023).

2.2.5 Code style

Another important factor that affects how reproducible software is they that the code is built. It is important to use a recognizable and logic coding style (Cooper et al., 2017). An essential part of this coding style is using meaningful file names. Cooper et al. (2017) give the following example where `abc.r` is a bad file name, while `01-download-data.R` is a good file name. The same goes for object names which should clearly refer to what the object is or does, without using a name of existing variables or functions. "Variable names should usually be nouns and function names should be verbs." (Cooper et al., 2017). Furthermore, it is important for the computational reproducibility of code that the code is clean, such that the understandability is increased (Filazzola & Lortie, 2022). Accessibility of code is also increased when the code is cleaned, because this enables other developers to reuse scripts. Filazzola and Lortie (2022) explain the following basics that are fundamental of clean code:

- Communication - Code should be clear from itself, without the need of a lot of comments. Using logic object names stimulate this
- Formatting - Using a consistent format makes code more readable and therefore easier for others to understand and reproduce.
- Abstraction - This is done such that code does not look too complicated. Applying abstraction is done by wrapping complex code chunks in functions which are loaded somewhere else in the code. This makes that it is easier to understand what the code does.

Besides these basics, Filazzola and Lortie (2022) describe some principles like the DRY principle. This means that avoiding duplication and repeating improves the readability of code. Other principles that Filazzola and Lortie (2022) introduce are the Single-Responsibility principle and Refactoring. Single responsibility means that each function should only perform one task and refactoring is improving code structure without changing how the code works (Filazzola & Lortie, 2022).
Before diving deeper in the theory of maturity models, it is good to know how the principles

above can be used in a model. The principles above can be seen as areas where improvement is possible. Documentation for example, is such an area and there are multiple ways of how documentation can be implemented. This can be multiple steps that build upon each other or it can be done by just doing it in a better way. Each of these steps add a different amount of maturity to such an area. In maturity models, those areas are called Focus Areas and the steps or different implementations are capabilities, thus a focus area consists of one or multiple capabilities.

2.3 Maturity Models

The first step in improving the computational reproducibility, is analyzing the current state of computational reproducibility. Maturity models help measuring the performance of organizations and can therefore be used to analyze the current computational reproducibility state. Maturity models are commonly used in research related with system information and are a way of capturing how mature an organization or process is (Lasrado et al., 2015). They evaluate the capabilities of an organization and show a roadmap of improvements (Becker et al., 2010). From the literature, we can distinguish multiple types of maturity models, namely fixed-level maturity models (FLMM) and Focus Area Maturity Models (FAMM) (Steenbergen et al., 2010). A famous fixed-level maturity model is the Capability Maturity Model (CMM) (Paulk et al., 1993). This model was created to describe the level of software within an organization. Paulk et al. (1993) describes a FLMM as a ”structured collection of elements that describe characteristics of effective processes at different levels of capability or maturity. The fixed-level approach assumes a linear progression through predefined stages, each requiring the completion of specific process areas.”. This directly shows a shortcoming of a FLMM as an organization does not always have a linear growth. FLMM's are therefore a solution for benchmarking and standardized growth, but they are not usable to analyze processes that improve incremental, because FLMM's are not able to express interdependencies between the processes that are linked to a maturity level (Steenbergen et al., 2010). To solve this issue, Steenbergen et al. (2010) came up with the FAMM. This type of maturity model has different focus areas and capabilities, which can be developed in a sequence that is decided by the organization itself. That means that a FAMM is more flexible and can be adjusted to the priorities of an organization. This feature comes out good as organizations coping with health data might prioritize security higher than companies that handle data that is already publicly available. Where in a fixed-level maturity model all stages have to be completed to reach a higher level of maturity, each capability of a focus area leads to a higher maturity level when using a Focus Area Maturity Model (Steenbergen et al., 2010).

FAMMs are applicable to a research context, as shown by Deekshitha et al. (2024), who have designed the Research Software Focus Area Maturity Model (RSMM). The RSSM has been designed since software often lacks quality and sustainability in order to be used properly for scientific research. This model focuses on areas like quality, security, sustainability and documentation. Deekshitha et al. (2024) has developed the RSMM in such a way that it serves as a checklist which can be used by organizations to rate their project management. By using the maturity model, the user gains insights on which elements improvements are possible and required. Another way that the maturity model can be meaningful is for funding agencies. They may use the model to evaluate which research projects are worth investing in. The maturity model that we will develop focuses on the computational reproducibility of software, therefore documentation might be the most important RSMM aspect for our model. As the used data consists of health data, security is an important component as well.

As more and more health data comes available, hospitals start investing in data analytics to make sure that their services are efficient and reliable (J. V. Carvalho et al., 2019). To identify the data analytics within hospitals, (J. V. Carvalho et al., 2016) have developed a 6-dimension maturity model, named Hospital Informations System Maturity Model (HISMM). This model was later validated and the data analytics dimension was transformed in a standalone Maturity Model (J. V. Carvalho et al., 2019) named HISMM-DA. Both models where constructed using literature review and design science. During the design science, multiple relevant maturity models were evaluated, which delivered six factors. Those factors are Electronic medial records, people, systems and IT infrastructure, data analysis, information security, Strategy, Cooperation, Usability, Telemedicine and mHealth. Those factors were then prioritized based on the input of health expert. Those experts received a survey where was asked to introduce their working career. The next part of the survey was were J. V. Carvalho et al. (2019) asked the respondents to rate the importance of each factor on a scale of 1 to 5. The respondents could also propose new factors. Furthermore, the experts had to rank the factors relative to each other. (J. V. Carvalho et al., 2019) have developed the Maturity Model using the methodology described by (Mettler, 2010). (Mettler, 2010) describes how to create a maturity model in five steps. Firstly, the need or opportunity of a new maturity model should be identified. If the need is there, the scope of the model has to be defined. This makes sure that the model won't be to broad, vague or complex. The third step in creating a maturity model is designing the model. The next step is evaluating the design. This could for example be done with the respondents of the interviews that were previously held. After these steps have been completed, Mettler (2010) states that the effects of applying the maturity model should be reflected. This is to make adaptions to the model if some effects of applying it turn out other than expected. The reflection step might lead to having to return to the first of the second step. This is schematically shown in Figure 2.

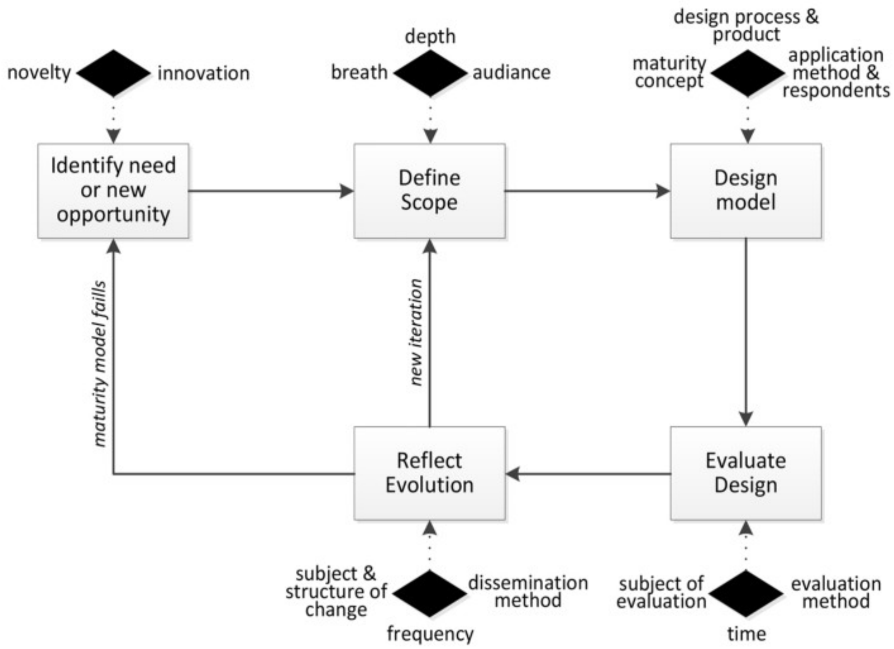

Figure 2: The five steps of creating a maturity model (Mettler, 2010)

The five steps that Mettler (2010) describes are primarily designed to create a fixed level maturity model, but since we will develop a focus area maturity model, these steps might not completely fit the approach that is required. Since Steenbergen et al. (2010) encountered the same while developing a FAMM, they analyzed the approach in Figure 2 and other approaches to develop a method that fits creating a FAMM. Steenbergen et al. (2010) made a UML visual of their design approach, which is shown in Figure 3. The development process consists of four phases, which are divided in a total of 10 steps:

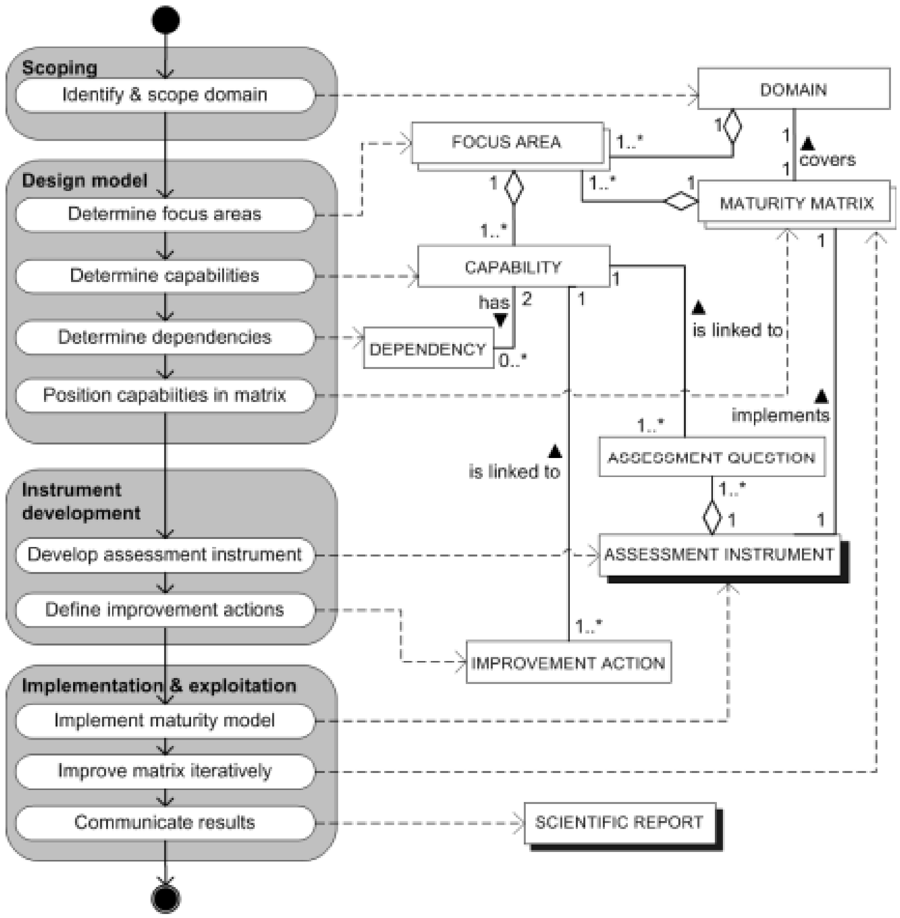

Figure 3: Designing a Focus Area Maturity Model (Steenbergen et al., 2010)

1. Scoping

- Identify & Scope domain: Before a FAMM can be developed, the domain must be scoped correctly. Steenbergen et al. (2010) describe that the developer of the maturity model has to know what should be included and excluded in the scope and which of the already existing maturity models in the same domain could be used as a starting point.

2. Design Model

- Determine focus areas: Focus areas can be identified using a literature review in combination with expert groups or caste studies (Steenbergen et al., 2010).

- Determine capabilities: All the focus areas have one or multiple capabilities linked to it. Those capabilities represent progressive maturity levels (Steenbergen et al., 2010). For each focus area, capabilities are placed in a path that adds more maturity for each step.

- Determine dependencies: For some capabilities, the order is fixed and for other capabilities, the order can differ based on preferences Steenbergen et al. (2010) gives the following example: ”it is not possible to use measurements for improvement if there is no measurement mechanism in place. In other cases the order is a preferred one, for instance it is advisable to set clear goals before putting a measurement mechanism in place, but it is not impossible, though unwise, to skip the step of goal-setting.” Before positioning the capabilities, is must be clear how they relate to each other.

- Position capabilities in matrix: The dependencies that are determined can now be used to position the capabilities. If a capability is dependent on another capability, it is placed to the right and therefore adds more maturity. After using field experience, the order can be further determined and the amount of scales become visible.

3. Instrument development

- Develop assessment instrument: To be able to measure the current maturity with a FAMM, Steenbergen et al. (2010) states that there should be control questions for the capabilities. These questions can be posed in a questionnaire. Those questions can be yes/no questions and if all questions that belong to a capability are answered with yes, the capability is achieved (Spruit & Roeling, 2014).

- Define improvement actions: During this step, it is possible to suggest the steps that can be taken to actually achieve a specific capability.

4. Implementing & Exploitation

- Implement maturity model: According to Steenbergen et al. (2010), there are multiple ways in which the FAMM can be implemented. They suggest to share a questionnaire with the assessment questions or to discuss the assessment questions in a workshop or interview.

- Improve matrix iteratively: Based on the responses to the assessment questions, the model can be evaluated. The responses should be stored such that the model can be evaluated on the long term.
- Communicate results: The last step that Steenbergen et al. (2010) describe is that the results of designing the maturity model should be communicated to both practitioners and to the scientific community.

According to Becker et al. (2010), maturity models often lack scientific foundations as they are mostly built upon experiences from practice. That means that during the design science process, using scientific research in addition to interviews makes a good foundation for a maturity model.

Based on this literature review, we may conclude that the concepts in Table 1 are deemed important for computational reproducibility.

Concept	Description
End-to-End Automation	By automating steps that were previously done manually, human errors are limited
Literate Programming	This makes the whole process of the analysis visible, enhancing reliability of analyses. It also enables other programmers to understand the code better
Compute Environment Control	When an operating system, programming language or package is updated, some functionalities may change. To prevent that this results in reproducibility issues, the versions that were used in the original environment should be documented, such that future developers can use the same versions as well. A more advanced option is to share the original environment as a virtual environment.
Documentation	Documentation is seen as one of the most important concepts in computational reproducibility as it enables new programmers to understand what is happening and why specific steps are taken. With good documentation, reproducing the results from an analysis should be possible by following the steps in the documentation
Version Control	When changes to code are made, they can cause the output to drastically change. By applying proper version control, older scripts can be brought back such that old results can be retrieved as well.
Persistent Code and Data Sharing	This means that data and code should be shared in order to make data analysis reproducible. Otherwise, the results can not be reproduced an therefore not be verified. When sensitive data is being used, as is often the case with health data, measures should be taken such that the data can still be shared without harming privacy of others.
Data Manipulation	Most of data analysis include data manipulation, where the original data set is changed such that it is usable for data analysis. The steps should also be part of the code, as other programmers also need to perform these manipulation steps to reproduce.
Code Style	A logic and recognizable code style enhances computational reproducibilty as well. This also includes the use of filenames and object names. Removing unused code parts also improves readability and therefore contributes to improved computational reproducibility.
Continuous Validation	The measures that are taken to improve computational reproducibility should be tested regularly to identify any room for improvement and to avoid issues with computational reproducibility.

Table 1: An overview of important concepts for computational reproducibility that can be derived from the literature review

3 Methodology

3.1 Scoping

When deciding the research topic, the scope has been set on data analysis on health data. Both data analysis performed by companies and researchers will be in the scope. The focus will lay on how to make the software that is used in both environments computational reproducible without harming privacy of individuals whose data are being used. There are two organizations within our scope, namely Zorg en Zekerheid and the Leiden University Medical Center (LUMC). One respondent was interviewed on behalf of her work for the University Medical Center Groningen (UMCG), but is also affiliated with Leiden University. For the rest of this thesis, this respondent is labeled under the name of LUMC. This section will describe the organizations of both Zorg en Zekerheid and LUMC.

3.1.1 Zorg en Zekerheid

Zorg en Zekerheid is a Dutch health insurer, with a main focus on the region around Leiden. Zorg en Zekerheid has approximately 500 employees. The employees that will be interviewed work for the knowledge center and the data center. These employees work as consultant, data analyst and data engineer. The data they work with are all related with health, but the purpose of data tables varies. Knowledge center employees mostly work with declaration data that comes from healthcare providers and insured persons who make declarations themselves. This data contains data about the insured person, the healthcare provider and the treatment that has been given. Other datasets give more information about the insured persons. It goes deeper on who a person is and contains information like address, age and name. It depends on the employee's function whether they are allowed to see this information. Furthermore, they work with external datasets like CBS data which is mostly used to do analysis on their population. The data center also works with those data sources. They prepare this data such that it can be used by others.

3.1.2 LUMC

Leiden University Medical Center is a university medical center affiliated with Leiden University. The LUMC is a hospital, where they cure patients, but they are also part of the university where research is done. One of the researching parts of LUMC is the health campus. That is where the scope of the LUMC is for this thesis. At the health campus researchers focus on population health in the region of The Hague. The main goal of the research at the health campus is shifting how the Dutch health system deals with illness. Instead of focusing on curing a disease, the health campus strives to prevent diseases. They do this by collaborating with important partners in the region like health care professionals, health insurance companies, local authorities and patients. At the moment, there are around 117 people working at the health campus in roles as student, student assistant, PhD candidate, researcher and management. They use data that is provided by their partners and from the CBS. Data managers make sure that researchers can safely work with the data that they are provided with. The health campus respondents work as either, data manager, director, Post-doc researcher and data scientist. As stated earlier, one of the respondents is interviewed on behalf of her work for the UMCG. Her role is reproducibility coordinator.

3.2 Design Model

3.2.1 Research Method

We will follow a design science approach to develop our maturity model (Hevner, 2007). A design science project consists of three parts, as shown in Figure 4.

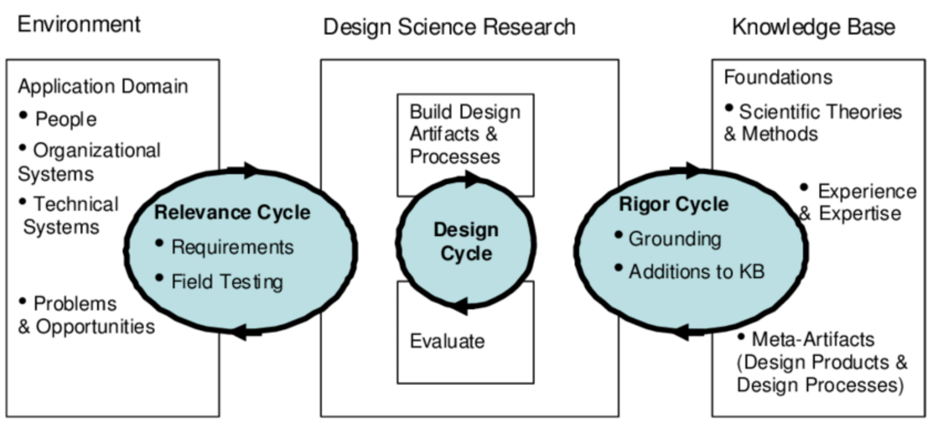

Figure 4: Design Science (Hevner, 2007)

The first cycle is called the relevance cycle and this is where experts will be interviewed. For this research, the experts who will be interviewed are people who work in the field with health data and have to deal with software that must be computational reproducible. The goal of the interviews is collecting information of best practices of reproducible work. This helps identifying the strengths and weaknesses of making reproducible software in practice. Other important data points that must be collected are the current state of reproducibility and the issues that are experienced if software is not reproducible enough. This information can best be acquired by performing interviews with professionals who work in the field. The respondents that will be selected for the interviews will be both researchers at LUMC and employees of Zorg en Zekerheid. Both organizations are described in Section 3.1. Besides the information that is gathered from the interviews, scientific literature will be used to collect relevant information for the maturity model, like capabilities or focus areas that are important for computational reproducibility. The interview will follow the concepts of a semi-structured interview. This means that the interview is not limited to the prepared questions and follow-up questions may be asked based on the responses of the respondents (Adams, 2015). During the design cycle, the maturity model will be created. After creating the model, evaluation is performed. The last cycle that Figure 4 displays is called the rigor cycle. This is where the findings from the first cycle will be compared with what already has been discovered during research in this field. Hevner (2007) states that this is done such that the new research will add value instead of repeating what is already known. Furthermore, this cycle makes sure that the performed research is reproducible. The evaluation will be done with experts with field experience. Those people will be from the same sample as the interviews that have been done during the relevance cycle. Figure 5 shows how the Design Science process will look for this thesis.

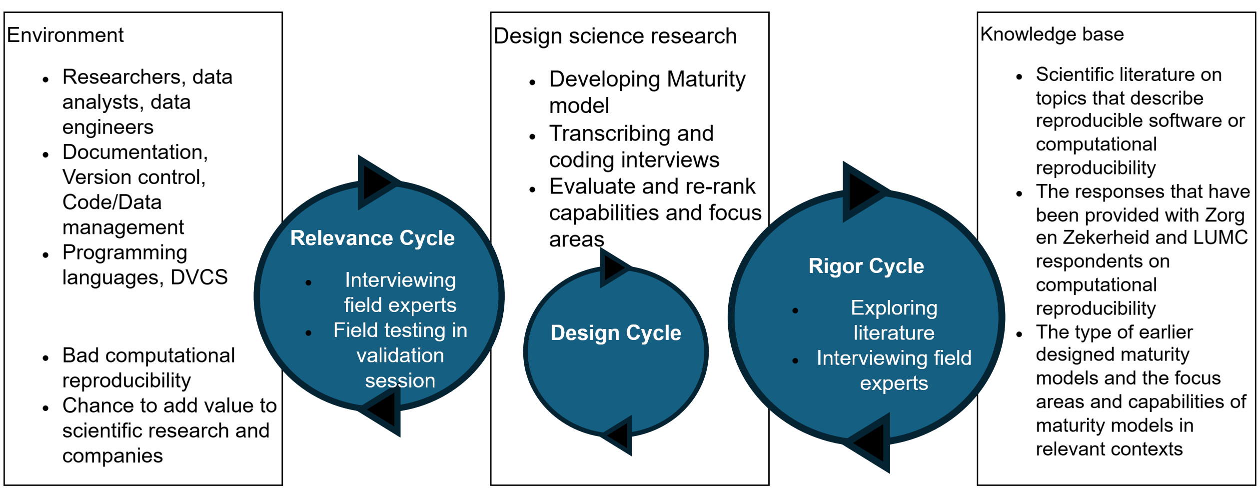

Figure 5: Design Science diagram of this research adapted from (Hevner, 2007)

The model design will follow the design science approach of Steenbergen et al. (2010) which is visualized in Figure 3.

3.2.2 Interview protocol

To prepare for the interview, a list of 27 interview questions has been created. The full list is shown in Appendix F. As the interview will follow the concept of a semi-structured interview, the order of the questions may differ per interview, some questions may not be asked and some questions which are not on the list may be asked. This all depends on the flow of the interview and how the respondent reacts to the questions. Before the start of the interview, a short introduction is given. This introduction depends on whether the respondent works at Zorg en Zekerheid or is a LUMC researcher. This introduction is described in Appendix F as well. As the respondents can be categorized in two different groups, they are selected in different ways. Respondents from Zorg en Zekerheid were selected based on experience and function and were personally approached. They received an email with the invite shown in Appendix E. On the other hand, the researchers where firstly invited in a more open email which was sent to multiple researchers. Due to a lack of respondents, researchers were personally approached as well by sending the invite shown in Appendix D. This resulted in a total of six Zorg en Zekerheid respondents and five LUMC respondents.
A total of 27 questions were prepared among 7 categories. As some questions contain some overlap and may already be answered before the question was asked, the total number of posed questions varies per interview. Based on the respondents input, some unprepared questions have been asked as well. Each interview had a duration of roughly 30-45 minutes.

3.2.3 Processing Interviews

This section describes how the interviews have been processed. This includes the steps that were taken after interviewing a respondent until having prioritized the capabilities that will be used in the maturity model.

3.2.4 Coding Interviews

Coding interviews is an important step in research as this generates quantitative data from the interviews. To do this, it is important to already understand which information can be relevant for the final product. Therefore, the literature review has to be studied such that important concepts will be recognized and can be labeled. Per interview, this information was labeled with the help of Atlas.TI. Atlas.ti is a software product that aids in coding textual data (Vila-Henninger, 2019). In this case it was used to label the data that was gathered in the interviews in order to prioritize which information was relevant for the maturity model. As it is important to know whether someone works at LUMC or Zorg en Zekerheid as this may influences the experiences, preferences and opinions someone has about a specific subject, each code contained a format namely [organization]_[n]:[label]. Where [organization] stand for either ZZ (Zorg en Zekerheid) or LUMC, [n] stands for the interview number and label stands for the actual label that describes what was said. For example the code starting with "LUMC_3" describes a statement of the LUMC interviewer with ID 3. After each interview had been coded, several code groups could be created. These code groups contain statements that were more of less the same. After having coded every interview, a list which contains every code and how often each code was used is generated and exported to Excel. By doing that, it was possible to see how often a statement was made which helped prioritizing future capabilities. This also created insights in differing preferences for both organizations.

3.2.5 MoSCoW

To measure the importance, each factor will be prioritized by using the MoSCoW framework (Kravchenko et al., 2022). This frame work describes four levels of importance, namely Must, Should, Could and Would. Must requirements have to be "satisfied in the final solution for the solution to be considered a success" (Kravchenko et al., 2022). Should requirements have a high priority and should be integrated if that is possible. Could requirements are wished to be in the final product but are not necessarily required. Finally, Kravchenko et al. (2022) states would requirements as requirements that will not be integrated in the final product, but may be considered for further development. A good way to prioritize requirements is by creating a MoSCoW matrix, where each requirement can be marked with a M, S, C or W (Kravchenko et al., 2022). A way to assign the MoSCoW labels to each statement is by measuring how much respondents have named a specific requirement. Each code group that was created in Section 3.2.4 was prioritized using the MoSCoW framework. The prioritization has been done by taking the number of Grounded in account for each group. After prioritizing the groups, the individual codes were prioritized. This was done by analyzing the prioritization of the code group it belonged to and by analyzing what the exact statement was. Since each individual code started with the name of the organization, it enabled us to compare prioritization of Zorg en Zekerheid with LUMC to see whether these two organizations have different ideas of what is important for computational reproducibility. The MoSCoW matrices on both code and code group level are shown in respectively Appendix B and Appendix C.

3.2.6 Model development

After applying the MoSCoW prioritization framework, we investigate which code groups are suitable to use as focus area for the maturity model. The prioritization that we have performed tells us which topics are found important by the respondents and are therefore important to be used as a focus area to measure maturity. This is all part of the Design phase in Figure 3. Groups which are prioritized with a "W" in the MoSCoW matrix in Appendix C will not be used as focus areas, while groups with a higher prioritization will be included in the maturity model. The way that each focus group contributes in the maturity of an organization may differ based on how its code group has been prioritized. Some code groups may not directly be used as a focus area, but can aid in being more mature on a focus area. That means that such a code group will be used as a capability of that specific focus area.

After having identified all focus areas, it is important to identify which capabilities will be required, following the FAMM development approach (Steenbergen et al., 2010). This capabilities can be seen as processes/steps towards maturity. Each focus area has a total of 13 maturity levels, where level 0 has the lowest maturity and level 12 has the highest maturity. At most five capabilities per focus area are spread on this maturity scale, where the amount of capabilities depends on how much steps can be taken to become mature on a single focus area. The distance between each capability, or the way that they are spread across the maturity scale, depends on how much effort is required to implement a capability and how much maturity it will deliver compared to maturity of the previous capability. Before the capabilities can be positioned on the matrix, the dependencies between the capabilities have been determined. The following example gives more insight into the process of creating focus areas and capabilities from the code groups: The code group "Organization should set guidelines for reproducibility" has a Could priority. That means that it will be used in the maturity model, where it will come back as a focus area as, depending on which guidelines are set, the maturity can differ. The code group "Programmer is responsible for reproducibility" is also prioritized as could. Since this code group refers to a type of guideline, it will be used as a capability for this focus area. The maturity level of this capability will be determined based on literature, field work experience and information from the interviews. This also counts for capabilities that can not directly be gathered from the interviews.

Another example is the code group "Uses SQL", which has a W priority. This means that this will not come back in the final model as using SQL as a programming language does not directly affect computational reproducibility.

To know if a capability is satisfied, assessment questions should be set. However, due to how the capabilities are formed, each capability should get just one yes/no question attached. This means that the questions are not formulated, but the description of the capability is clear enough to determine if the capability is achieved or not.

3.3 Instrument Development

To assess whether a capability is reached, there should be some assessment questions for each of those capabilities. This are yes/no questions. If the questions that belong to a capability can be answered with yes, that capability is satisfied. However, each of the capability in our model comes with a description, which describes how this capability can be satisfied. Based on these descriptions, an organization can determine if this capability is reached and there are no separate assessment questions developed.

3.4 Implementing & Exploitation

After filling in the model with all focus groups and their capabilities in the maturity levels where they could be placed based on the interviews and literature, it is time to verify that this meets the requirements of field experts, which starts the Implementing & Exploitation phase of Figure 3. This verification will be done with a sample of five Zorg en Zekerheid employees, who work with health data on a daily basis. This sample consists of a a junior, medior and senior data analyst. The other two employees are a data engineer and data analyst consultant as shown in Table 2.

Function	# of respondents
Data Consultant	1
Senior Data analyst	1
Medior Data analyst	1
Junior Data analyst	1
Data Engineer	1

Table 2: Participants of the Focus Session

This makes sure that there are varying experiences and specialisms involved in this focus group. The choice for a focus group interview over regular interviews is made because of the multiple advantages that focus groups offer. According to Wilkinson (2004), focus groups are more efficient than regular interviews, as they involve more respondents in a specific time frame. Furthermore, respondents might feel more comfortable in a focus group session as this is a more natural setting than one-on-one interviews (Wilkinson, 2004). The focus group will take 90 minutes and consists of 5 stages. After a brief introduction of the research topic and the goal of the session, the first exercise begins. The maturity model that has been developed will be used to evaluate the current maturity of the organization. This is done such that they will get an idea of how the model works and whether the calculations meet their expectations.. After this exercise, the participants will evaluate the current capabilities. During this part, participants will be asked to arrange the capabilities on a scale of 1 to 12. They will have to do this for each focus area by moving post-its across the wall. To stimulate discussion, the participants were split up into two groups. Using prepared post-its, a big matrix was formed on the wall with focus areas on the vertical axis and maturity levels on the horizontal axis. Both groups received a set of post-its with the name of each capability written on it such that they could rearrange the capabilities on how they would place it based on their experience. They also used the pre-evaluated maturity model as a reference. After discussing the results of this exercise, the idea was that the participants could bring in missing capabilities and focus areas. However due to a lack of time, this was changed and merged with the previous exercise, thus the participants both rearranged the capabilities and added or removed other capabilities and focus areas. After the missing capabilities and focus areas have been discussed and there is a consensus within the group, the final maturity model will be formed. After that, there will be a conclusion and the session will be finalized and closed. The full program for the focus session is displayed in Table 3.

Activity	Description	Duration
Introduction	The thesis will be introduced to the participants. The main focus will lay on how the model should be used to measure maturity	10 minutes
Evaluate organization	Participants will use the current maturity model to measure how mature the organization is if the current model would be used	25 minutes
Evaluate Capabilities	Participants will evaluate how much maturity each capability adds by ranking them with a 1-12 score for each focus area	30 minutes
Add missing capabilities and Focus areas	This activity will offer the participants of the focus session the possibility to suggest any missing capabilities and focus areas	15 minutes
Final evaluation	Based on the participants input, the final model will be formed	10 minutes

Table 3: Distribution of respondents

4 Results

4.1 Interview results

The performed interviewed can be placed on two categories, namely with Zorg en Zekerheid employees and with LUMC employees. They both work with personal data about health care, although they differ in the environment where they work with the data. Zorg en Zekerheid is health insurance company and therefore has a business environment. LUMC is a faculty of Leiden University which will deliver 4 researchers as respondents, while Zorg en Zekerheid deliver 6 respondents. Besides that, the invitation letter was also redirected to the research coordinator at UMCG, who is affiliated with Leiden University. The distribution on function title is shown in Table 4.

The following subsections will highlight the differences in the interviews between researchers and Zorg en Zekerheid employees. Those differences are also shown in Table 5. This table contains the amount of time that a respondent mentioned something that belongs in a specific code group. This is done such that the different preferences between the two organizations can be analyzed. As the amount of respondents in each group were not equal, the amount in the ZZ column has been divided by 6 and in the LUMC column by 5. This way, the quantity of each code group is per person and thus comparable. The last column contains the difference between the amount of quotation between Zorg en Zekerheid and LUMC. The higher this value, the higher the difference of importance between those two organizations. For example, the first row shows that the code group ”Documentation should stick to a format” has been named 0.5 times per respondent by Zorg en Zekerheid employees and 0.6 times by LUMC researchers. This shows that it is named 0.1 time more often by LUMC employees.

Organization	Function	# of respondents
Zorg en Zekerheid	Data Consultant	2
Zorg en Zekerheid	Senior Data analyst	2
Zorg en Zekerheid	Data Engineer	1
Zorg en Zekerheid	Senior Data Engineer	1
LUMC	Assistant Professor	1
LUMC	Scientific Director	1
LUMC	Data manager	1
LUMC	Data scientist	1
UMCG	Reproducibility coordinator	1

Table 4: Distribution of respondents. In the labled interviews, the UMCG researcher is treated as an LUMC researcher

Code Groups	ZZ	LUMC	Most Grounded	Difference
Documentation should stick to a format	0.5	0.6	LUMC	0.1
Version control supports reproducibility	1.333333333	1.2	ZZ	0.133333333
Importance of reproducibility	2	1.6	ZZ	0.4
Code should be reviewed	0.333333333	0.6	LUMC	0.266666667
Advanced measures when basics are implemented	0	1	LUMC	1
Code must contain documentation	1.333333333	1.6	LUMC	0.266666667
Sensitive data make reproducibility harder	0.166666667	1	LUMC	0.833333333
Documentation must contain the reason for a piece of code	0.5	1	LUMC	0.5
Documentation must contain the location of dependencies	0	0.2	LUMC	0.2
Reproducibility in science is mostly for other researchers	0	1.4	LUMC	1.4
Organization should set guidelines for reproducibility	0.166666667	0.6	LUMC	0.433333333
Programmers quality influences reproducibility	0	0.4	LUMC	0.4
Measures to support reproducibility with sensitive data	0.166666667	1	LUMC	0.833333333
Programmer is responsible for reproducibility	0.166666667	0.2	LUMC	0.033333333
Code review is optional	0	0.2	LUMC	0.2
No documentation required	0.333333333	0.2	ZZ	0.133333333
Sensitive data does not affect reproducibility	0.5	0.6	LUMC	0.1
Documentation should contain used variables	0.833333333	1	LUMC	0.166666667
Code should stick to a format	1.333333333	0.4	ZZ	0.933333333
Agreements support reproducibility	0.833333333	0.2	ZZ	0.633333333
Clean code supports reproducibility	0.833333333	0.2	ZZ	0.633333333
Team is responsible for guidelines	0.5	0.6	LUMC	0.1
Data editing in code improves reproducibility	0.333333333	0.6	LUMC	0.266666667
Uses SQL	0	0	Identical	0
Level of reproducibility depends on impact/probability of reusing	0.166666667	0	ZZ	0.166666667

Table 5: Different code groups per organization

4.2 Comparing LUMC and Zorg en Zekerheid

The interviews at Zorg en Zekerheid were spread across two teams namely the Data Center, consisting of Data engineers and the Knowledge center, consisting of Data analysts and Data consultants. Both teams only share code within their organization, mostly just within their own team. This means that there are less issues with sharing data as almost everyone that an analysis is shared with is allowed to see the data. This can be extracted from Table 5, where LUMC scores higher at code groups about data sharing. This is measured by comparing the amount of times that a code group has been labeled per person. Examples are advanced measures to enable computational reproducibility and sensitive data sharing. On the other hand, Zorg en Zekerheid respondents experience that sharing code internally is more important. This can be related from the following code groups *Code should stick to a format* and *Agreements support reproducibility*. The first code group describes that there should be a format where all code complies with. This is more relevant for code that is internally shared since this makes it easier for programmers to find the structure of the code. Researchers, who also share code externally, will contribute less from this because it will be very hard to agree on a format with all researchers world wide. The second code group that was mentioned is more important for Zorg en Zekerheid employees based on the same principles: Agreements between colleagues can easily be made and followed up, but it is very hard to do that with all researchers.
The code group *Reproducibility in Science is mostly important for other researchers* has only been used by researchers. This is logic, as this only describes the opinion of researchers. Some respondents shared the opinion that there is not directly any benefit from making their research reproducible. They mentioned that it feels like wasting time to make their analysis completely reproducible as the chances that their work will be used for further research will be low. To counter this problem, it is important that they are motivated to make their work computational reproducible. This can be done by letting the organization set guidelines for reproducibility. As shown in Table 5, researchers seem to give this a higher priority in order to counter the lack of reproducibility as LUMC respondents mentioned this 0.43 times more often than Zorg en Zekerheid respondents. The last code group where a notable difference between both respondent groups was measured is *Clean code supports reproducibility*. This is about removing unnecessary code parts, using recognizable names and removing files that are not required from the repository. Zorg en Zekerheid respondents have mentioned this more that LUMC employees.

4.3 Model Deliverable

After having prioritized all codes and codes groups using MoSCoW prioritization, which is shown in Appendix B and Appendix C, the focus areas and capabilities could be formed. Table 10 shows that some code groups are either classified with ”W” or with ”-”. The latter means that the code was not relevant or possible to transform to a focus area. The following code groups received that classification and will not be used as a focus area:

- Sensitive data makes reproducibility harder
- Reproducibility in science is mostly for other researchers
- Sensitive data doesn't affect reproducibility
- Uses SQL

As SQL does not necessarily impact computational reproducibility, this code group has been excluded. Furthermore, there are two code groups about sensitive data which were conflicting excluded as focus area. Therefore, those code groups can not be used as focus areas. However, measures on sensitive data are still present in the model. Lastly, the code group which states that reproducibility in science is mostly for other researchers has been excluded since this is not necessarily an area where improvement is possible. Besides that, it can be used in the guidelines that an organization can set for reproducibility standards. Although these code groups will not be used as focus areas some are used as a capability. The full list of how these code groups are used in the final maturity model is shown in Table 12 in Appendix G. In this table, each code groups contains a description of what the code group means and which actions should follow based on this code group. Once the description of the code group shows that an action leads to improved computational reproducibility, the code group can be transformed into a focus area. However, some of these transformed code groups can be seen as ”sub-actions” of other focus areas. If that is the case, that code group will be transformed to a capability of the specific focus area. For example, we have the code groups ”Version control supports reproducibility” and the code group ”Using GIT for version control” both code groups will result in an improvement in computational reproducibility, but the code group about GIT is a smaller part of version control, and therefore a capability of the focus area version control. The results of applying this table to create the maturity model is shown in Appendix H. The maturity model can be found in Table 13. For each focus area, the capabilities are displayed with the letters A to E. A description of what the letters of the capabilities mean is shown in Table 14 in

Appendix H. As stated before, the capabilities are formed partly with the help of Table 12. Other capabilities were created based on what was further learned from the interviews and assumed steps into implementing these focus areas.

As mentioned in Section 3, a validation session in the form of a focus group has been organized to verify the maturity model in Table 13. During this session, experts from Zorg en Zekerheid gave their opinion and feedback on the maturity model through several exercises. Since they are persons who have a lot of experience in the field and work with software on health data on a daily basis, their feedback is taken very seriously and has lead to several adjustments of the model. Most of those adjustments are about capabilities that should be moved to another maturity level, while some adjustments contain swapping the order of capabilities. Lastly, the feedback has lead to the removal of focus area 7. Capability 7B has therefore been moved to focus area 13. The full model, after the focus session, is shown in Table 6.

Focus Area: Maturity Level:	0	1	2	3	4	5	6	7	8	9	10	11	12
Organizational													
1. Code Reviews	A		B			C			D			E	
2. Organizational Guidelines	A	B			C					D			
3. Training for Programmers	A			B				C					
4. Agreeing on Measures	A	B		C				D					
5. Complexity/Impact dependent Measures	A		B	C			D						
Technical													
6. Advanced Features	A								B				C
7. Dependencies in Documentation	A		B	C					D	E			
8. Variables in Documentation	A			B					C				
9. Clean code on delivery	A		B		C	D							
10. Manipulating Data	A			B				C		D			
Organizational and Technical													
11. Version Control	A			B	C				D			E	
12. Documentation	A				B	C			D			E	
13. Data sharing measures	A		B		C				D				E
14. Code Standardization	A					B							

Table 6: Final Maturity Model

As Table 6 shows, each capability in the final maturity model is noted with the letter A to E. To assess if a capability and therefore a maturity level is reached, each capability has a description that shows which conditions should be satisfied to reach the capability and the maturity level it belongs to. Those descriptions for the final model are shown in Table 7.

Focus area	A	B	C	D	E
1	There is no code review performed	Before a coding project is finalized a colleague runs the code	The code is run in a clean compiler without packages being installed	The code review is performed with the help of a pre defined checklist	Besides checking whether the code functions, other guidelines like format, privacy and security are checked as well
2	Organization does not have guidelines regarding computational reproducibility	Organization does not directly have guidelines but gives freedom to teams	Organizations have a vision/demands in reproducibility, teams are free in how they adhere to it	Organizations have guidelines on how to reach computational reproducibility	
3	Less skilled programmers are not differed from better programmers	Less skilled programmers receive passive training to improve their skills (like datacamp licenses)	Less skilled programmers are linked with better programmers on projects		
4	There are no agreements about how code should be made reproducible	Some agreements are made	Agreements are well documented and accessible for everyone	Agreements are checked by colleagues	
5	No difference in impact of analysis	Less impact-full analysis are excluded from reproducibility rules	Project/Analysis has to be classified in level of impact by someone else	Different reproducibility rules apply for each level of impact	

Focus area	A	B	C	D	E
6	No advanced measures are being taken	Code is saved as a package including the used data and dependencies	A virtual environment is created for each coding project such that all dependencies are present		
8	Documentation does not cover the location of dependencies	Documentation contains the source of external data	Documentation contains an overview of used packages/libraries	Documentation shows the used version of the used packages/libraries	Documentation includes the version of the programming language
9	No guidelines about how variables should be documented	Documentation contains which variables are used and can be changed	It is documented how each variables should be changed and when they should be changed		
10	No guidelines on cleaning code	Code is straight to the point and uses recognizable variable names	Unused parts of code are removed	The code doesn't contain unused libraries/packages	
11	No guidelines of logging data changes	Data can be changed outside of the code	Changes to the data can be done externally but are well documented	Changes to the data are made in a different code file	All manipulation steps are done in the same file as where the analysis is made

Focus area	A	B	C	D	E
12	No version control is applied	A text file is used logging each new version	New versions are stored under a different name	Teams store their code in a version control system with GIT	The team has guidelines for how their code should be stored in GIT
13	No or optional documentation	Code contains inline documentation of code parts	Documentation contains an overview of used packages/libraries	A separate text file is created with information of the code	Documentation follows a structured format
14	Nothing is shared when working with sensitive data	Code is shared with others, without data being issued	Code is shared, limited information on the dataset is shared, like structure or column names	Data is shared with the code but only in a pseudonymized way (n¿=10 for example)	A synthetic dataset is shared with the code. Dataset is representative for original data
15	There is no specific format for code	There is a general format for code			

Table 7: Description of the capabilities for each focus area

5 Discussion

5.1 Key Findings

This research has focused on developing a Focus Area Maturity Model that enables organizations who work with health data to measure how strong their computational reproducibility is and how they can improve it. As our respondents come from roughly two types of organizations, namely a health insurance company and a researching university, there are differences in the issues they experience when reproducing the code of someone else. Because they experience different problems, they also have different preferences on the contents of the FAMM. Researchers, for example, struggle a lot more with how sensitive data is shared. The sensitive data can not be shared if it can be traced to one or a small group of persons, which means that certain measures should be taken in order to safely share sensitive data. The focus of those researchers lies more on how sensitive data can be shared. They currently experience that this is barely done properly, resulting in irreproducibility, or at least an unnecessary amount of effort that is required to reproduce an analysis. They urge for measures to share sensitive data in a safe manner. Examples of those measures are sharing the metadata of the original dataset, sharing pseudonymized data or sharing sensitive data. The contradicting part of these measures is that the respondents say that it sometimes feels like wasting time to make their own research completely reproducible. They mention that the chances of their work being reproduced by other researchers is very low and that spending time and effort in creating synthetic data is not efficient. Furthermore, they acknowledge that making their research reproducible is mostly beneficial for the researcher who is reproducing the work, rather than for the researcher of the original work.

Instead, employees of companies experience way less issues regarding sensitive data. Most of the analysis on sensitive data are shared internally with colleagues who have the same rights when in comes to viewing data. Their preferences when it comes to improving reproducibility can be found in measures that are mainly internal realizable. Examples of those measures are making agreements on how software should be made reproducible, delivering clean code and developing a standard format for code.

Besides the differences between both type of data analysts, there are also similarities between their ideas about computational reproducibility. Both groups of respondents say that version control is very important for computational reproducibility. This is done such that changes in the code are visible for future reproduction and make it easier to find out why results may have changed. Ideally, version control should be managed in a Version

Control System, like GIT. An extra advantage is that such a system is suited to share code and data with others. Furthermore, they say that documentation is one of the most important aspects of computational reproducibility. Documentation can take multiple forms and enables new data analysts, when executed properly, to get familiar with the code and find required datasets and dependencies.

Before starting this research, we had the following research question:

"How can a maturity model aid in improving computational reproducibility in health data analysis?"

The first sub-question that had to be answered is

What are currently encountered issues when trying to reproduce other's work that is build to analyze health data?

This question has been answered with our literature review and the interviews that have been performed. The literature review showed general issues that occur when reproducing data analysis. The interviews gave deeper insights on the issues that the respondents experience in practice and gave good examples of those issues. Both the literature review and the interviews also helped us answering the second sub-question which was :

What are the components and capabilities that strengthen computational reproducibility?

After researching this question, it turned out that most of the issues that are encountered whilst reproducing data analysis, are also the components and capabilities that strengthen reproducibility. In hindsight this was logic, as these issues would not hinder computational reproducibility if this were not important to computational reproducibility. In addition to naming these issues as the only factors that influence computational reproducibility, the interviews also gave us insight in measures that the organizations where the respondents work have already implemented. We also got insights in desired measures of the respondents and measures that the respondents have seen at past organizations. By combining the strengths and weaknesses that the respondents have experienced and that are described by the literature, we could answer the third sub-question:

What should a maturity model contain in order to measure computational reproducibility of software processing health data?

Combining those factors together turned out the be what was required for the maturity model. Together with the literature review, the interviews gave a good mix of focus areas and capabilities. After bringing all the focus areas and capabilities into the maturity model, this model was evaluated, during a focus session. During this session, the respondents have used the maturity model to evaluate the maturity of their organization and were allowed to adapt the model such that it gave a good representation of how it should work in practice. Due to the feedback on the initial model, a final model is created that answers the research question. With our Focus Area Maturity Model, we measure the maturity of computational reproducibility on 14 focus areas, divided in three categories. Those focus areas consist of 56 capabilities which are spread over the 13 maturity level counting FAMM.

5.2 Comparison with other models

To our knowledge, this is the first research that develops a FAMM on the topic of computational reproducibility concerning health data. Other types of maturity models have been used while scoping on software development. For example, a Fixed Level Maturity Model about software is the Capability Maturity Model, designed by Paulk et al. (1993). This is a traditional maturity model that focuses on software development on general, and not necessarily on computational reproducibility. A shortcoming of the CMM in contrast to our model is that the CMM only contains five maturity levels, which is often seen with FLMMs. This means that it is assumed that the organization shows a linear growth over all capabilities. In practice, this is barely the case and therefore, a Focus Area Maturity Model is a better way of measuring computational reproducibility and showing how computational reproducibility can become more mature. Furthermore, to reach a higher maturity level, multiple capabilities should be satisfied, which means that maturity is not measured for each focus area. Being extremely mature at a certain focus area, can still give a low maturity score if other focus areas are not that mature yet. Our model contains twelve maturity levels and the maturity of each focus area can be evaluated. Except for maturity level 1, Paulk et al. (1993) has composed the maturity levels into several key process areas which have to be satisfied in order to reach a certain maturity level. The first maturity level, the initial level, does not have any key process areas and it is assumed that there are no process implemented that enhance the quality of software development. At maturity level 2, which Paulk et al. (1993) have called the Repeatable level, the key process areas are mainly focusing on basic project management, so these are not comparable with our focus areas and capabilities. In the defined stage, which is the third maturity level, we find some reproducibility focused process areas organization process focus, organization process definition, inter-group coordination, training program and peer review. The first two process areas are about the organization setting guidelines and rules for software development. This can be compared with our Organizational guidelines focus areas. Inter-group coordination is about different groups collaborating. This is not used in our maturity model, but it can not be denied that collaborating may improve reproducibility. Furthermore the process areas training program and peer review are somewhat equal to our focus areas training programmers and code review. Stage four, the managed maturity level, there are two key process areas which are about testing and evaluating the results of a project. We do not have any focus areas which focus on verifying the software output. In the fifth maturity level, optimizing, the processes are focused on improving the current processes. A comparable focus area in our model could be the advanced features focus area, where there is space for measures that can be included if other measures are already implemented.

An FAMM that has a scope of software development is the research software maturity model (RSMM) by Deekshitha et al. (2024). This model is focused on the whole concept of software development and therefore does not contain detailed focus areas or capabilities on reproducibility. The RSMM is more suited for software project management, than improving reproducibility of that software. Despite that, they also have some focus areas which evaluate reproducibility. Firstly, the RSMM contains a focus area code quality, which contains capabilities that vary from code reviews to continuously stability measurement. They also measure of easy code can be used. It can be assumed that code that is easy to use, is also easier to reproduce. Deekshitha et al. (2024) also mention a focus area Documentation. That focus area contains the capabilities ”A read me file is provided”, ”Common example usage is provided” and ”A documentation infrastructure is provided”. This shows that the RSMM is less detailed on documentation. There is also a reproducibility focus area, where the lowest maturity is gained from having instructions about how to reproduce, while the highest maturity is being added by having a replication package.

When focusing on healthcare institutions, the Hospital Information System Maturity Model by J. Carvalho et al. (2019) is a FAMM which focuses on data analytics within hospitals. Although this model is specifically focusing on data analytics and the required software, it is more generally focusing on the bigger picture of data analytics and the required infrastructure instead of scoping on computational reproducibility. However, their data analysis focus area contains some capabilities that focus on data management, which our model does as well. The strategy focus area maturity depends on whether the organization has guidelines or a strategy, which influences computational reproducibility in our model. The people focus area shows some comparable capabilities, namely that people need to be trained in order to go the second maturity level. Furthermore, standardized agreements are required for maturity level four, which our model has as a capability as well.

Besides the capabilities that are mentioned above, the maturity models do not dive deeper on computational reproducibility. That means that, as far as our knowledge reaches, our model is the only FAMM that measures and improves computational reproducibility for organizations that work with health data.

5.3 Limitations and Further research

This research follows a design science approach which relies on a literature review and and expert interviews. The interview questions have been created based on what is deemed important from the literature and own experience with data analysis on health data. Those questions have been asked to respondents of two working areas, namely Zorg en Zekerheid and LUMC. This might be a limitation, because adding more organizations might deliver new insights and other best practices, making the model more valuable and allow professionals to learn from each other. Our analysis shows that employees from Zorg en Zekerheid barely experience issues with sharing their analysis externally, which might create the view that these issues do not occur with companies. However, sharing data does actually happen in the field of health care. Further development of our model could be done by gathering respondents from departments or companies who do share data and code externally. The validation of the maturity model has been done solely with Zorg en Zekerheid employees, this might bias the model to much in the favor of those respondents, while it should cover both work fields. For further research, it might be a good idea to add weights to the focus areas of the model, those weights should then depend on the type of organization. The results in Section 4 show that some issues are more important to researchers than to employees of a company and vice versa. This is not included in the model and their is no difference in the importance of the focus group for different organizations. Validating with researchers and gathering information on the weights they prefer could enhance the maturity model further.

6 Conclusion

During this thesis, research was done on how a maturity model can contribute to improving computational reproducibility when working with health data. After performing a literature review on both computational reproducibility and maturity models, developing a Focus Area Maturity Model (FAMM) was the desired solution. That is because a FAMM, in contrast to other maturity models, allows for incremental improvement and enables the capabilities to vary in the amount of maturity that is being added. To answer the sub-questions that are used to answer the main research question, two organizations that perform data analysis on health data were selected. Both organizations, Zorg en Zekerheid and LUMC, delivered respondents for interviews. Zorg en Zekerheid respondents work as data analysts, data engineers or consultant. The functions that LUMC respondents have are assistant professor, scientific director, data manager and data scientist. Furthermore there was an UMCG respondent who works as reproducibility coordinator. That respondents is linked to LUMC respondents.

The interviews and literature review were both used to answer the research questions. Firstly, it was important to understand what the current pain points are in terms of computational reproducibility. The interviews showed that missing documentation and not working dependencies are the biggest issues when trying to reproduce other's data analysis. Furthermore, the interviews showed that there are differences between data analysts that work as researchers or those that work for a company. Based on the interviews, researchers seem to find it more important that sharing sensitive data goes well. Sharing data is important for reproducibility as this enables validating outcomes of an analysis. However, sharing data may not harm privacy. This is not much of an issue for the Zorg en Zekerheid respondents as most of their analysis is only shared internal, with people who have the same authorities when it comes to seeing data. The fact that analysis is mostly shared internally at Zorg en Zekerheid can also be retrieved from the fact that their respondents mentioned more about making agreements about reproducibility and creating a standard format for code.

All the insights from these interviews have been combined with what is already known from the literature that has been reviewed. That has lead to the development of a Focus Area Maturity Model, which initially consisted of fifteen focus areas and 53 capabilities. After those capabilities were positioned on the matrix, the maturity model was evaluated with employees from Zorg en Zekerheid. As they are experts on the field of data analysis, they could give a good opinion on how these focus areas and capabilities would work in practice. This evaluation was held in the form of a focus group, where five respondents collectively assessed the focus areas and the capabilities. This evaluation session led to some adjustments to the maturity model. The final model contains fourteen focus areas and 56 capabilities.

References

Adams, W. C. (2015). Conducting semi-structured interviews. *Handbook of practical program evaluation*, 492–505. https://doi.org/10.1002/9781119171386.ch19

Baker, M. (2016). 1, 500 scientists lift the lid on reproducibility. *Nature*, *533*(7604), 452–454. https://doi.org/10.1038/533452a

Becker, J., Niehaves, B., Poeppelbuss, J., & Simons, A. (2010). Maturity models in is research. *ECIS 2010 Proceedings*. https://aisel.aisnet.org/cgi/viewcontent.cgi?article=1096&context=ecis2010

Carvalho, J. V., Rocha, A., Vasconcelos, J., & Abreu, A. (2019). A health data analytics maturity model for hospitals information systems. *International Journal of Information Management*, *46*, 278–285. https://doi.org/https://doi.org/10.1016/j.ijinfomgt.2018.07.001

Carvalho, J. V., Rocha, Á., & Abreu, A. (2016). Of the paper: Maturity of hospital information systems: Most important influencing factors.

Carvalho, J., João Vidal de, Rocha, Á., Van De Wetering, R., & Abreu, A. (2019). A maturity model for hospital information systems. *Journal of Business Research*, *94*, 388–399. https://research.ou.nl/ws/portalfiles/portal/41289598/A_Maturity_model_for_hospital_information_systems.pdf

Cook, C., & Visconti, M. (1994). Documentation is important. *CrossTalk*, *7*(11), 26–30.

Cooper, N., et al. (2017). A guide to reproducible code in ecology and evolution. http://hdl.handle.net/10141/622618

Deekshitha, Bakhshi, R., Maassen, J., Ortiz, C. M., van Nieuwpoort, R., & Jansen, S. (2024). Rsmm: A framework to assess maturity of research software project. https://arxiv.org/abs/2406.01788

Filazzola, A., & Lortie, C. (2022). A call for clean code to effectively communicate science. *Methods in Ecology and Evolution*, *13*(10), 2119–2128. https://doi.org/10.1111/2041-210X.13961

Freedman, L. P., Cockburn, I. M., & Simcoe, T. S. (2015). The economics of reproducibility in preclinical research. *PLoS biology*, *13*(6), e1002165. https://doi.org/10.1371/journal.pbio.1002165

Gonzales, A., Guruswamy, G., & Smith, S. R. (2023). Synthetic data in health care: A narrative review. *PLOS Digital Health*, *2*(1), e0000082. https://doi.org/10.1371/journal.pdig.0000082

Grüning, B., Chilton, J., Köster, J., Dale, R., Soranzo, N., Van Den Beek, M., Goecks, J., Backofen, R., Nekrutenko, A., & Taylor, J. (2018). Practical computational reproducibility in the life sciences. *Cell Systems*, *6*(6), 631–635. https://doi.org/10.1016/j.cels.2018.03.014

Hevner, A. R. (2007). A three cycle view of design science research. *Scandinavian Journal of Information Systems*, *19*(2), Article 4. https://aisel.aisnet.org/cgi/viewcontent.cgi?article=1017&context=sjis

Hurley, D. G., Budden, D. M., & Crampin, E. J. (2015). Virtual reference environments: A simple way to make research reproducible. *Briefings in bioinformatics*, *16*(5), 901–903. https://doi.org/10.1093/bib/bbu043

Jati, P. H. P., Lin, Y., Nodehi, S., Cahyono, D. B., & van Reisen, M. (2022). Fair versus open data: A comparison of objectives and principles. *Data Intelligence*, *4*(4), 867–881. https://doi.org/10.1162/dint_a_00176

Kappagantula, A., & Fds, D. (2025). Git version control: A comprehensive guide to modern software development. *INTERNATIONAL JOURNAL OF RESEARCH IN COMPUTER APPLICATIONS AND INFORMATION TECHNOLOGY*, *8*, 548–562. https://doi.org/10.34218/IJRCAIT_08_01_043

Kery, M. B., Radensky, M., Arya, M., John, B. E., & Myers, B. A. (2018). The story in the notebook: Exploratory data science using a literate programming tool. https://doi.org/10.1145/3173574.3173748

Knuth, D. E. (1984). Literate programming. *The computer journal*, *27*(2), 97–111. https://doi.org/10.1093/comjnl/27.2.97

Kowalczyk, S., & Shankar, K. (2011). Data sharing in the sciences. *Annual review of information science and technology*, *45*(1), 247–294.

Kravchenko, T., Bogdanova, T., & Shevgunov, T. (2022). Ranking requirements using moscow methodology in practice. In R. Silhavy (Ed.), *Cybernetics perspectives in systems* (pp. 188–199). Springer International Publishing. https://doi.org/10.1007/978-3-031-09073-8_18

Lakens, D. (2016). Improving your statistical inferences. https://lakens.github.io/statistical_inferences/14-computationalreproducibility.html

Lasrado, L. A., Vatrapu, R., Andersen, K. N., for Information Systems, A., (AISeL), A. E. L., & School, C. B. (2015). Maturity models development in is research: A literature review. *Issue Nr 6 (2015)*. https://aisel.aisnet.org/cgi/viewcontent.cgi?article=1005&context=iris2015

McNutt, M. (2014). Reproducibility. *Science*, *343*(6168), 229–229. https://doi.org/10.1126/science.1250475

Mettler, T. (2010). Thinking in terms of design decisions when developing maturity models. *International Journal of Strategic Decision Sciences*, *1*, 77–88. https://doi.org/10.4018/jsds.2010100105

Paulk, M., Curtis, B., Chrissis, M., & Weber, C. (1993). Capability maturity model, version 1.1. *Software, IEEE*, *10*, 18–27. https://doi.org/10.1109/52.219617

Piccolo, S. R., & Frampton, M. B. (2016). Tools and techniques for computational reproducibility. *GigaScience*, *5*(1). https://doi.org/10.1186/s13742-016-0135-4

Sandve, G. K., Nekrutenko, A., Taylor, J., & Hovig, E. (2013). Ten simple rules for reproducible computational research. *PLoS Computational Biology*, *9*(10), e1003285. https://doi.org/10.1371/journal.pcbi.1003285

Shepherd, B. E., Blevins Peratikos, M., Rebeiro, P. F., Duda, S. N., & McGowan, C. C. (2017). A pragmatic approach for reproducible research with sensitive data. *American Journal of Epidemiology*, *186*(4), 387–392. https://doi.org/10.1093/aje/kwx066

Spruit, M., & Roeling, M. (2014). Isfam: The information security focus area maturity model. *ECIS 2014 Proceedings*. https://core.ac.uk/download/pdf/301362294.pdf

Steenbergen, M., Bos, R., Brinkkemper, S., Van de Weerd, I., & Bekkers, W. (2010). The design of focus area maturity models. *6105*, 317–332. https://doi.org/10.1007/978-3-642-13335-0_22

Stohr, E. A., & Zhao, J. L. (2001). Workflow automation: Overview and research issues. *Information Systems Frontiers*, *3*(3), 281–296. https://doi.org/10.1023/a:1011457324641

Vila-Henninger, L. A. (2019). Turning talk into "rationales": Using the extended case method for the coding and analysis of semi-structured interview data in atlas. ti. *Bulletin of Sociological Methodology/Bulletin de Méthodologie Sociologique*, *143*(1), 28–52. https://doi.org/10.1177/0759106319852887

Wilkinson, S. (2004). Focus group research. *Qualitative research: Theory, method, and practice*, *2*, 177–199.

Zayas-Cabán, T., Okubo, T. H., & Posnack, S. (2022). Priorities to accelerate workflow automation in health care. *Journal of the American Medical Informatics Association*, *30*(1), 195–201. https://doi.org/10.1093/jamia/ocac197

Ziemann, M., Poulain, P., & Bora, A. (2023). The five pillars of computational reproducibility: Bioinformatics and beyond. *Briefings in Bioinformatics*, *24*(6). https://doi.org/10.1093/bib/bbad375

Zolkifli, N. N., Ngah, A., & Deraman, A. (2018). Version control system: A review. *Procedia Computer Science*, *135*, 408–415. https://doi.org/10.1016/j.procs.2018.08.191

Appendices

A Labels

Code	Grounded	Code Group
LUMC_1: Uses GIT for version control	1	Using GIT for version control
LUMC_1: Reproducible code is also in own interest	1	Importance of reproducibility
LUMC_1: Fixed format is important for documentation	1	Documentation should stick to a format
LUMC_1: Version control important for reproducibility	1	Version control supports reproducibility
LUMC_1: Code review for reproducibility	1	Code should be reviewed
LUMC_1: Containers and reviews are more advanced	1	Advanced measures when basics are implemented
LUMC_1: Documentation and version control are the basis for reproducibility	1	Version control supports reproducibility
LUMC_1: Documentation important for reproducibility	1	Code must contain documentation
LUMC_1: Poor documentation biggest stumbling block	1	Code must contain documentation
LUMC_1: Updated code after paper publication is bad for reproducibility	1	Version control supports reproducibility
LUMC_1: Sensitive data makes verification of code harder	1	Sensitive data make reproducibility harder
LUMC_1: Inline documentation contains reasoning behind steps	1	Documentation must contain the reason for a piece of code
LUMC_1: Broken dependencies main cause of poor reproducibility	1	Documentation must contain the location of dependencies
LUMC_1: Reproducibility contributes to research integrity	1	Importance of reproducibility
LUMC_1: Reproduction rarely goes smoothly	1	Reproducibility in science is mostly for other researchers

Code	Grounded	Code Group
LUMC_1: Guidelines for reproducibility must come from the organization	1	Organization should set guidelines for reproducibility
LUMC_1: Poor programmers need more dependencies	1	Programmers quality influences reproducibility
LUMC_1: Synthetic data helps reproducibility with sensitive data	1	Measures to support reproducibility with sensitive data
LUMC_1: Synthetic data feels like a waste of time	1	Measures to support reproducibility with sensitive data
LUMC_1: Executing guidelines is own responsibility	1	Programmer is responsible for reproducibility
LUMC_2: With a logical outcome, code is not checked	1	Code review is optional
LUMC_2: Reproducibility for colleague absence	1	Importance of reproducibility
LUMC_2: Among researchers no urgency to reproduce due to small chance it is needed	1	Reproducibility in science is mostly for other researchers
LUMC_2: CBS has expert for each dataset	1	Advanced measures when basics are implemented
LUMC_2: Documentation subordinate to planning	1	No documentation required
LUMC_2: Documentation available for each dataset	1	Code must contain documentation
LUMC_2: Little research is reused, making reproducibility feel useless	1	Reproducibility in science is mostly for other researchers
LUMC_2: Sensitive person-level data must be pseudonymized for researchers	1	Measures to support reproducibility with sensitive data
LUMC_2: Reproducibility only important when something goes wrong	1	Importance of reproducibility
LUMC_2: Too reproducible code can also cause errors with side effects	1	Programmers quality influences reproducibility
LUMC_3: Code must give the same result at each run	1	Importance of reproducibility

Code	Grounded	Code Group
LUMC_3: Documentation contains reasoning behind steps	1	Documentation must contain the reason for a piece of code
LUMC_3: Sensitive data does not hinder reproducibility	1	Sensitive data does not affect reproducibility
LUMC_3: More errors possible due to extra steps with sensitive data	1	Sensitive data make reproducibility harder
LUMC_3: Researcher uses GIT for version control	1	Version control supports reproducibility
LUMC_3: Random seed AI must be in documentation	1	Documentation should contain used variables
LUMC_3: Fixed format is important for documentation	1	Documentation should stick to a format
LUMC_3: Fixed format for code improves reproducibility	1	Code should stick to a format
LUMC_3: Sharing code via channels other than the process causes version issues	1	Agreements support reproducibility
LUMC_3: Code review for reproducibility	1	Code should be reviewed
LUMC_3: Git contributes to reproducibility	1	Version control supports reproducibility
LUMC_3: Ideally each team has its own data manager/engineer	1	Advanced measures when basics are implemented
LUMC_3: Reproducibility must be checked with audits	1	Organization should set guidelines for reproducibility
LUMC_3: Reproducible code contains fewer errors	1	Importance of reproducibility
LUMC_3: Guidelines for reproducibility must come from the organization	1	Organization should set guidelines for reproducibility
LUMC_3: Clean data makes reproducing easier	1	Clean code supports reproducibility
LUMC_3: Synthetic data helps reproducibility with sensitive data	1	Measures to support reproducibility with sensitive data
LUMC_3: Teams are themselves responsible for reproducibility	1	Team is responsible for guidelines

Code	Grounded	Code Group
LUMC_4: Edits on dataset are done in the code itself	1	Data editing in code improves reproducibility
LUMC_4: Publishing code as a package is good for reproducibility	1	Advanced measures when basics are implemented
LUMC_4: Documentation important for reproducibility	2	Code must contain documentation
LUMC_4: Documentation contains reasoning behind steps	2	Documentation must contain the reason for a piece of code
LUMC_4: Documentation must be up-to-date	1	Code must contain documentation
LUMC_4: Documentation should not contain an explanation of what the code does	1	Documentation must contain the reason for a piece of code
LUMC_4: Sensitive data does not hinder reproducibility	1	Sensitive data does not affect reproducibility
LUMC_4: Not fully reproducible due to ban on sharing sensitive data	1	Sensitive data make reproducibility harder
LUMC_4: Research is poorly reproducible due to no benefits	1	Reproducibility in science is mostly for other researchers
LUMC_4: Reproducibility due to transparent use of public money	1	Importance of reproducibility
LUMC_4: Fixed format is important for documentation	1	Documentation should stick to a format
LUMC_4: Frequently used code is probably more reproducible	1	
LUMC_4: Scientific research must be reproducible for reliability	1	Reproducibility in science is mostly for other researchers
LUMC_4: Used version of Python/R must be in documentation	1	Documentation should contain used variables
LUMC_4: Version control is difficult in a secure environment	1	Sensitive data make reproducibility harder
LUMC_5: Edits on dataset are done in the code itself	1	Data editing in code improves reproducibility
LUMC_5: Code is shared on GitHub for reproducibility	1	Version control supports reproducibility

Code	Grounded	Code Group
LUMC_5: Data center uses GIT for version control	1	Using GIT for version control
LUMC_5: Documentation contains currently used versions and packages	1	Documentation should contain used variables
LUMC_5: Sensitive data does not hinder reproducibility	1	Sensitive data does not affect reproducibility
LUMC_5: Human errors remain possible with tooling	1	Using GIT for version control
LUMC_5: Not fully reproducible due to ban on sharing sensitive data	1	Sensitive data make reproducibility harder
LUMC_5: Research is poorly reproducible due to no benefits	1	Reproducibility in science is mostly for other researchers
LUMC_5: Team is responsible for reproducibility	1	Team is responsible for guidelines
LUMC_5: Fixed format shortens reviewer time	1	Code should stick to a format
LUMC_5: Using the same data as the basis of analysis improves reliability	1	Data editing in code improves reproducibility
LUMC_5: Within a team there must be a driver for reproducibility	1	Team is responsible for guidelines
LUMC_5: Documentation most important factor for reproducibility	1	Code must contain documentation
LUMC_5: With Docker the correct environment is created	1	Advanced measures when basics are implemented
LUMC_5: Missing documentation biggest stumbling block	1	Code must contain documentation
LUMC_5: Lack of required version and packages affects reproducibility	1	Documentation should contain used variables
LUMC_5: Operating system must be in documentation	1	Documentation should contain used variables
LUMC_5: Using old repos is usually less reproducible	1	Reproducibility in science is mostly for other researchers
LUMC_5: Reproducibility is easier in Python than in R	1	Importance of reproducibility

Code	Grounded	Code Group
LUMC_5: Synthetic data helps reproducibility with sensitive data	1	Measures to support reproducibility with sensitive data
LUMC_5: Self-check on reproducibility with a clean system	1	Code should be reviewed
ZZ_1: Data center uses GIT for version control	1	Version control supports reproducibility
ZZ_1: DC uses SQL	1	Uses SQL
ZZ_1: DC uses fixed format for code	1	Code should stick to a format
ZZ_1: Documentation is the most important part of reproducibility	1	Code must contain documentation
ZZ_1: Due to sensitive data, more attention is paid to code content	1	Measures to support reproducibility with sensitive data
ZZ_1: One-off analyses are not well documented	1	Level of reproducibility depends on impact/probability of reusing
ZZ_1: It would be fine if the organization provided guidelines on reproducibility	1	Organization should set guidelines for reproducibility
ZZ_1: New products are reproducible due to agreements made	1	Agreements support reproducibility
ZZ_1: Level of documentation depends on product size	1	Level of reproducibility depends on impact/probability of reusing
ZZ_1: Poor documentation detrimental to colleagues	1	Code must contain documentation
ZZ_1: Team has agreements on code format	1	Code should stick to a format
ZZ_1: Team wants to document reasoning behind steps	1	Documentation must contain the reason for a piece of code
ZZ_2: Editing data in multiple programs is inconvenient	1	Data editing in code improves reproducibility
ZZ_2: Documentation depends on code complexity	1	Level of reproducibility depends on impact/probability of reusing
ZZ_2: Poor documentation causes errors	1	Code must contain documentation
ZZ_2: Used variables are always recorded	1	Documentation should contain used variables

Code	Grounded	Code Group
ZZ_2: Sensitive data does not hinder reproducibility	1	Sensitive data does not affect reproducibility
ZZ_2: Processes the software runs on must also be documented	1	Agreements support reproducibility
ZZ_2: Reproducibility important for government control	1	Importance of reproducibility
ZZ_2: Reproducibility currently depends on impact	2	Level of reproducibility depends on impact/probability of reusing
ZZ_3: Code must contain comments per step	1	Documentation must contain the reason for a piece of code
ZZ_3: Documentation important for reproducibility	1	Code must contain documentation
ZZ_3: Documentation contains work instruction	1	Documentation must contain the reason for a piece of code
ZZ_3: Documentation must contain the source of external data	1	Documentation should contain used variables
ZZ_3: Documenting is not fun	1	Code must contain documentation
ZZ_3: Knowledge sharing prevents production disruption	1	Agreements support reproducibility
ZZ_3: Reproducibility for colleague absence	1	Importance of reproducibility
ZZ_3: Reviewer should independently run the script for verification	1	Code should be reviewed
ZZ_3: Cleaning is important for reproducibility	1	Clean code supports reproducibility
ZZ_3: Team uses checklist to check reproducibility	2	Team is responsible for guidelines
ZZ_3: Fixed format is important for documentation	2	Documentation should stick to a format
ZZ_3: Version control important for reproducibility	2	Version control supports reproducibility
ZZ_3: Without version control more errors when collaborating	1	Version control supports reproducibility

Code	Grounded	Code Group
ZZ_4: Cleaning code is important for reproducibility	1	Clean code supports reproducibility
ZZ_4: Editing data in multiple programs is inconvenient	1	Data editing in code improves reproducibility
ZZ_4: Documentation must contain the source of external data	1	Documentation should contain used variables
ZZ_4: One-off analyses do not need to be reproducible	1	Level of reproducibility depends on impact/probability of reusing
ZZ_4: Sensitive data does not hinder reproducibility	2	Sensitive data does not affect reproducibility
ZZ_4: Repeatable things must be reproducible	2	Importance of reproducibility
ZZ_4: Reproducibility should be analyst's own responsibility	1	Programmer is responsible for reproducibility
ZZ_4: Reproducibility currently depends on impact	1	Level of reproducibility depends on impact/probability of reusing
ZZ_4: Reproducibility for collaboration	1	Importance of reproducibility
ZZ_4: Reproducibility for colleague absence	1	Importance of reproducibility
ZZ_4: Fixed format for code improves reproducibility	3	Code should stick to a format
ZZ_4: Version control important for reproducibility	2	Version control supports reproducibility
ZZ_5: Code must give the same result at each run	1	Importance of reproducibility
ZZ_5: Data center uses GIT for version control	1	Version control supports reproducibility
ZZ_5: DC does not use format for documentation	1	No documentation required
ZZ_5: DC uses fixed format for reproducible code	2	Code should stick to a format
ZZ_5: Documentation only important when issues occur	1	No documentation required

Code	Grounded	Code Group
ZZ_5: More errors possible due to extra steps with sensitive data	1	Sensitive data make reproducibility harder
ZZ_5: Human errors remain possible with tooling	1	Using GIT for version control
ZZ_5: Reproducible code ensures the same outcomes	1	Importance of reproducibility
ZZ_5: Fixed format is important for documentation	1	Documentation should stick to a format
ZZ_5: Fixed format for code improves reproducibility	1	Code should stick to a format
ZZ_5: Responsibility for reproducibility must lie within the team	1	Team is responsible for guidelines
ZZ_5: Version control most important for reproducibility	1	Version control supports reproducibility
ZZ_6: Agreements improve reproducibility	2	Agreements support reproducibility
ZZ_6: Analysis is currently rarely reproducible	1	Importance of reproducibility
ZZ_6: Documentation improves reproducibility	1	Code must contain documentation
ZZ_6: Random seed AI must be in documentation	1	Documentation should contain used variables
ZZ_6: Reproducibility important for client	1	Importance of reproducibility
ZZ_6: Reproducibility important for taking over work	1	Importance of reproducibility
ZZ_6: Reproducibility currently depends on impact	1	Level of reproducibility depends on impact/probability of reusing
ZZ_6: Cleaning code improves reproducibility	1	Clean code supports reproducibility
ZZ_6: Variables must be reproducible	1	Documentation should contain used variables
ZZ_6: Important to clean up and structure code	1	Clean code supports reproducibility
ZZ_6: Checking data improves reproducibility	1	Code should be reviewed

Code	Grounded	Code Group
ZZ_6: Documentation and cleaning important points	1	Clean code supports reproducibility
ZZ_6: Documentation and cleaning important points	1	Code must contain documentation
ZZ_6: Clear definition of documentation is necessary	1	Code must contain documentation
ZZ_6: One-off analyses do not need to be reproducible	1	Level of reproducibility depends on impact/probability of reusing
ZZ_6: Estimate must be reproducible	1	Importance of reproducibility

Table 8: All quoted codes and the code group it belongs to

B MoSCoW Matrix per code

Code	Code Group	Code Group grounded	M	S	C	W	-	MSCW Group
LUMC_1: Uses GIT for version control	Using GIT for version control	4		x				c
LUMC_1: Fixed format is important for documentation	Documentation should stick to a format	6		x				s
LUMC_1: Version control important for reproducibility	Version control supports reproducibility	14	x					m
LUMC_1: Reproducible code is also in own interest	Importance of reproducibility	20					x	m
LUMC_1:Code review for reproducibility	Code should be reviewed	5		x				c
LUMC_1:Containers and reviews are more advanced	Advanced measures when basics are implemented	5				x		c
LUMC_1:Documentation and version control are the basis for reproducibility	Version control supports reproducibility	14	x					m
LUMC_1:Documentation important for reproducibility	Code must contain documentation	16	x					m
LUMC_1:Poor documentation is the biggest stumbling block	Code must contain documentation	16					x	m

Code	**Code Group**	**Code Group grounded**	**M**	**S**	**C**	**W**	**-**	**MSCW Group**
LUMC_1:Updated code after paper publication is bad for reproducibility\	Version control supports reproducibility	14			x			m
LUMC_1:Sensitive data makes verification of code harder	Sensitive data make reproducibility harder	6			x			nvt
LUMC_1:Inline documentation contains the reasoning behind steps	Documentation must contain the reason for a piece of code	8	x					m
LUMC_1:Broken dependencies main cause of poor reproducibility	Documentation must contain the location of dependencies	1		x				c
LUMC_1:Reproducibility contributes to integrity of research	Importance of reproducibility	20	x					m
LUMC_1:Reproducing rarely goes smoothly	Reproducibility in science is mostly for other researchers	7					x	nvt
LUMC_1:Guidelines for reproducibility must come from the organization	Organization should set guidelines for reproducibility	4	x					c
LUMC_1:Poor programmers need more dependencies	Programmers quality influences reproducibility	2					x	nvt

Code	Code Group	Code Group grounded	M	S	C	W	-	MSCW Group
LUMC_1:Synthetic data helps reproducibility with sensitive data	Measures to support reproducibility with sensitive data	6		x				s
LUMC_1:Synthetic data feels like a waste of time	Measures to support reproducibility with sensitive data	6		x				s
LUMC_1:Executing guidelines is own responsibility	Programmer is responsible for reproducibility	2				x		c
LUMC_2: Reproducibility for colleague dropout	Importance of reproducibility	20	x					m
LUMC_2: With a logical outcome, code is not checked	Code review is optional	1			x			w
LUMC_2:Among researchers no urgency to reproduce due to the small chance that it is needed	Reproducibility in science is mostly for other researchers	7		x				nvt
LUMC_2:CBS has an expert for each dataset	Advanced measures when basics are implemented	5				x		c
LUMC_2:Documentation subordinate to planning	No documentation required	3			x			w
LUMC_2:Documentation is available for every dataset	Code must contain documentation	16	x					m

Code	Code Group	Code Group grounded	M	S	C	W	-	MSCW Group
LUMC_2:Little research is reused, making reproducibility feel useless	Reproducibility in science is mostly for other researchers	7			x			nvt
LUMC_2:Sensitive person-level data must be pseudonymized for researchers	Measures to support reproducibility with sensitive data	6	x					s
LUMC_2:Reproducibility only important when something goes wrong	Importance of reproducibility	20				x		m
LUMC_2:Too reproducible code can also cause errors with side effects	Programmers quality influences reproducibility	2		x				nvt
LUMC_3: Code must give the same result at each run	Importance of reproducibility	20	x					m
LUMC_3: Documentation contains the reasoning behind steps	Documentation must contain the reason for a piece of code	8	x					m
LUMC_3: Sensitive data does not hinder reproducibility	Sensitive data does not affect reproducibility	6					x	nvt
LUMC_3: More errors possible due to extra steps with sensitive data	Sensitive data make reproducibility harder	6				x		nvt

Code	Code Group	Code Group grounded	M	S	C	W	-	MSCW Group
LUMC_3: Researcher uses GIT for version control	Version control supports reproducibility	14		x				m
LUMC_3: Random seed AI must be in documentation	Documentation should contain used variables	10	x					m
LUMC_3: Fixed format is important for documentation	Documentation should stick to a format	6		x				s
LUMC_3: Fixed format for code improves reproducibility	Code should stick to a format	10	x					m
LUMC_3:Sharing code via channels other than the process causes version issues	Agreements support reproducibility	6		x				s
LUMC_3:Code review for reproducibility	Code should be reviewed	5		x				c
LUMC_3:Git contributes to reproducibility	Version control supports reproducibility	14		x				m
LUMC_3:Ideally each team has its own data manager/engineer	Advanced measures when basics are implemented	5				x		c
LUMC_3:Reproducibility must be checked with audits	Organization should set guidelines for reproducibility	4			x			c
LUMC_3:Reproducible code contains fewer errors	Importance of reproducibility	20	x					m

Code	Code Group	Code Group grounded	M	S	C	W	-	MSCW Group
LUMC_3:Guidelines for reproducibility must come from the organization	Organization should set guidelines for reproducibility	4			x			c
LUMC_3:Clean data makes reproducing easier	Clean code supports reproducibility	6	x					s
LUMC_3:Synthetic data helps reproducibility with sensitive data	Measures to support reproducibility with sensitive data	6			x			s
LUMC_3:Teams are themselves responsible for reproducibility	Team is responsible for guidelines	6			x			s
LUMC_4: Documentation important for reproducibility	Code must contain documentation	16	x					m
LUMC_4: Sensitive data does not hinder reproducibility	Sensitive data does not affect reproducibility	6					x	nvt
LUMC_4: Fixed format is important for documentation	Documentation should stick to a format	6		x				s
LUMC_4: Edits on the dataset are done in the code itself	Data editing in code improves reproducibility	5		x				s
LUMC_4: Publishing code as a package is good for reproducibility	Advanced measures when basics are implemented	5				x		c

Code	Code Group	Code Group grounded	M	S	C	W	-	MSCW Group
LUMC_4: Documentation contains the reasoning behind steps	Documentation must contain the reason for a piece of code	8	x					m
LUMC_4: Documentation must be up-to-date	Code must contain documentation	16	x					m
LUMC_4: Documentation should not include an explanation of what the code does	Documentation must contain the reason for a piece of code	8		x				m
LUMC_4: Not fully reproducible due to ban on sharing sensitive data	Sensitive data make reproducibility harder	6					x	nvt
LUMC_4: Research is poorly reproducible due to no benefits	Reproducibility in science is mostly for other researchers	7					x	nvt
LUMC_4: Frequently used code is probably more reproducible	-	0-					x	nvt
LUMC_4: Scientific research must be reproducible for reliability	Reproducibility in science is mostly for other researchers	7	x					nvt
LUMC_4:Used version of Python/R must be in documentation	Documentation should contain used variables	10	x					m
LUMC_4:Version control is difficult in a secure environment	Sensitive data make reproducibility harder	6			x			nvt
LUMC_5: Edits on the dataset are done in the code itself	Data editing in code improves reproducibility	5		x				s

Code	Code Group	Code Group grounded	M	S	C	W	-	MSCW Group
LUMC_5: Data center uses GIT for version control	Using GIT for version control	4		x				c
LUMC_5: Sensitive data does not hinder reproducibility	Sensitive data does not affect reproducibility	6					x	nvt
LUMC_5: Human errors remain possible with tooling	Using GIT for version control	4					x	c
LUMC_5: Not fully reproducible due to ban on sharing sensitive data	Sensitive data make reproducibility harder	6					x	nvt
LUMC_5: Research is poorly reproducible due to no benefits	Reproducibility in science is mostly for other researchers	7					x	nvt
LUMC_5: Code is shared on GitHub for reproducibility	Version control supports reproducibility	14		x				m
LUMC_5: Documentation contains currently used versions and packages	Documentation should contain used variables	10	x					m
LUMC_5:Within a team there must be a driver for reproducibility	Team is responsible for guidelines	6		x				s
LUMC_5:documentation most important factor for reproducibility	Code must contain documentation	16	x					m

Code	Code Group	Code Group grounded	M	S	C	W	-	MSCW Group
LUMC_5:With Docker the correct environment is created	Advanced measures when basics are implemented	5				x		c
LUMC_5:Missing documentation biggest stumbling block	Code must contain documentation	16	x					m
LUMC_5:Lack of required version and packages affects reproducibility	Documentation should contain used variables	10	x					m
LUMC_5:Operating system must be in documentation	Documentation should contain used variables	10			x			m
LUMC_5:Using old repos is usually less reproducible	Reproducibility in science is mostly for other researchers	7					x	nvt
LUMC_5:Reproducibility is easier in Python than in R	Importance of reproducibility	20				x		m
LUMC_5:Synthetic data helps reproducibility with sensitive data	Measures to support reproducibility with sensitive data	6			x			s
LUMC_5:Self-check on reproducibility with a clean system	Code should be reviewed	5			x			c
LUMC_5: Team is responsible for reproducibility	Team is responsible for guidelines	6		x				s
LUMC_5: Fixed format shortens reviewer time	Code should stick to a format	10	x					m

Code	Code Group	Code Group grounded	M	S	C	W	-	MSCW Group
LUMC_5: Using the same data as the basis of analysis improves reliability	Data editing in code improves reproducibility	5		x				s
ZZ_1 Data center uses GIT for version control	Version control supports reproducibility	14		x				m
ZZ_1 DC uses SQL	Uses SQL	1				x		w
ZZ_1 DC uses fixed format for code	Code should stick to a format	10	x					m
ZZ_1 Documentation is the most important part of reproducibility	Code must contain documentation	16	x					m
ZZ_1 Because of sensitive data, the code content is monitored more closely	Measures to support reproducibility with sensitive data	6			x			s
ZZ_1 One-off analyses are not well documented	Level of reproducibility depends on impact/probability of reusing	9			x			m
ZZ_1 It would be fine if the organization provided guidelines on reproducibility	Organization should set guidelines for reproducibility	4		x				c
ZZ_1 New products are reproducible due to agreements made	Agreements support reproducibility	6	x					s
ZZ_1 Level of documentation depends on product size	Level of reproducibility depends on impact/probability of reusing	9			x			m

Code	Code Group	Code Group grounded	M	S	C	W	-	MSCW Group
ZZ_1 Poor documentation is detrimental to colleagues	Code must contain documentation	16	x					m
ZZ_1 Team has agreements on code format	Code should stick to a format	10	x					m
ZZ_1 Team wants to document the reasoning behind steps	Documentation must contain the reason for a piece of code	8	x					m
ZZ_3: documentation important for reproducibility	#N/B	#N/B	x					#N/B
ZZ_2: Editing data in multiple programs is inconvenient	Data editing in code improves reproducibility	5		x				s
ZZ_2: Sensitive data does not hinder reproducibility	Sensitive data does not affect reproducibility	6					x	nvt
ZZ_2: Reproducibility currently depends on impact	Level of reproducibility depends on impact/probability of reusing	9			x			m
ZZ_2: Documentation depends on code complexity	Level of reproducibility depends on impact/probability of reusing	9			x			m
ZZ_2: Poor documentation causes errors	Code must contain documentation	16	x					m
ZZ_2: Used variables are always recorded	Documentation should contain used variables	10	x					m

Code	Code Group	Code Group grounded	M	S	C	W	-	MSCW Group
ZZ_2: Processes the software runs on must also be documented	Agreements support reproducibility	6		x				s
ZZ_2: Reproducibility important for government control	Importance of reproducibility	20	x					m
ZZ_3: Documentation must contain the source of external data	Documentation should contain used variables	10	x					m
ZZ_3: Reproducibility for colleague dropout	Importance of reproducibility	20	x					m
ZZ_3: Code must contain comments per step	Documentation must contain the reason for a piece of code	8		x				m
ZZ_3: Documentation contains work instruction	Documentation must contain the reason for a piece of code	8			x			m
ZZ_3: Documenting is not fun	Code must contain documentation	16					x	m
ZZ_3: Knowledge sharing prevents production disruption	Agreements support reproducibility	6		x				s
ZZ_3: Reviewer should independently run the script for verification	Code should be reviewed	5		x				c
ZZ_3: Cleaning is important for reproducibility	Clean code supports reproducibility	6		x				s

Code	Code Group	Code Group grounded	M	S	C	W	-	MSCW Group
ZZ_3: Team uses a checklist to check reproducibility	Team is responsible for guidelines	6		x				s
ZZ_3: Fixed format is important for documentation	Documentation should stick to a format	6		x				s
ZZ_3: Version control important for reproducibility	Version control supports reproducibility	14	x					m
ZZ_3: Without version control, more errors when collaborating	Version control supports reproducibility	14	x					m
ZZ_4: One-off analyses do not need to be reproducible	Level of reproducibility depends on impact/probability of reusing	9			x			m
ZZ_4: Reproducibility currently depends on impact	Level of reproducibility depends on impact/probability of reusing	9			x			m
ZZ_4: Cleaning code is important for reproducibility	Clean code supports reproducibility	6		x				s
ZZ_4: Editing data in multiple programs is inconvenient	Data editing in code improves reproducibility	5		x				s
ZZ_4: Documentation must contain the source of external data	Documentation should contain used variables	10	x					m

Code	Code Group	Code Group grounded	M	S	C	W	-	MSCW Group
ZZ_4: Sensitive data does not hinder reproducibility	Sensitive data does not affect reproducibility	6					x	nvt
ZZ_4: Repeatable things must be reproducible	Importance of reproducibility	20	x					m
ZZ_4: Reproducibility should be the analyst's own responsibility	Programmer is responsible for reproducibility	2			x			c
ZZ_4: Reproducibility for collaboration	Importance of reproducibility	20	x					m
ZZ_4: Reproducibility for colleague dropout	Importance of reproducibility	20	x					m
ZZ_4: Fixed format for code improves reproducibility	Code should stick to a format	10	x					m
ZZ_4: Version control important for reproducibility	Version control supports reproducibility	14	x					m
ZZ_5: Fixed format is important for documentation	Documentation should stick to a format	6		x				s
ZZ_5: Fixed format for code improves reproducibility	Code should stick to a format	10	x					m
ZZ_5: Code must give the same result at each run	Importance of reproducibility	20	x					m
ZZ_5: Data center uses GIT for version control	Version control supports reproducibility	14		x				m

Code	Code Group	Code Group grounded	M	S	C	W	-	MSCW Group
ZZ_5: DC does not use a format for documentation	No documentation required	3					x	w
ZZ_5: DC uses fixed format for reproducible code	Code should stick to a format	10	x					m
ZZ_5: Documentation only important when issues occur	No documentation required	3			x			w
ZZ_5: More errors possible due to extra steps with sensitive data	Sensitive data make reproducibility harder	6			x			nvt
ZZ_5: Human errors remain possible with tooling	Using GIT for version control	4			x			c
ZZ_5: Reproducible code ensures the same outcomes	Importance of reproducibility	20	x					m
ZZ_5: Responsibility for reproducibility should lie within the team	Team is responsible for guidelines	6			x			s
ZZ_5: Version control most important for reproducibility	Version control supports reproducibility	14	x					m
ZZ_6: Cleaning of code improves reproducibility	Clean code supports reproducibility	6		x				s
ZZ_6: Agreements improve reproducibility	Agreements support reproducibility	6		x				s

Code	Code Group	Code Group grounded	M	S	C	W	-	MSCW Group
ZZ_6: Analysis is currently rarely reproducible	Importance of reproducibility	20					x	m
ZZ_6: Documentation improves reproducibility	Code must contain documentation	16	x					m
ZZ_6: Random seed AI must be in documentation	Documentation should contain used variables	10	x					m
ZZ_6: Reproducibility important for client	Importance of reproducibility	20	x					m
ZZ_6: Reproducibility important for taking over work	Importance of reproducibility	20	x					m
ZZ_6: Reproducibility currently depends on impact	Level of reproducibility depends on impact/probability of reusing	9			x			m
ZZ_6: Variables must be reproducible	Documentation should contain used variables	10	x					m
ZZ_6:Important to clean up and structure code	Clean code supports reproducibility	6		x				s
ZZ_6:Checking data improves reproducibility	Code should be reviewed	5			x			c
ZZ_6:Documentation and cleaning most important points	Clean code supports reproducibility	6		x				s

Code	**Code Group**	**Code Group grounded**	**M**	**S**	**C**	**W**	**-**	**MSCW Group**
ZZ_6:One-off analyses do not need to be reproducible	Level of reproducibility depends on impact/probability of reusing	9			x			m
ZZ_6:Estimate must be reproducible	Importance of reproducibility	20	x					m
ZZ_6:Documentation and cleaning most important points	Code must contain documentation	16	x					m

Table 9: Moscow Prioritization of each Code

C MoSCoW Matrix per code group

Code Group	Quoted amount	M	S	C	W	-	ZZ m	ZZ s	ZZ C	ZZ w	LUMC m	LUMC s	LUMC c	LUMC w
Using GIT for version control	4			x			0	0	0.17	0	0	0.4	0	0
Documentation should stick to a format	6		x				0	0.33	0	0	0	0.6	0	0
Version control supports reproducibility	14	x					0.67	0.17	0.17	0	0.4	0.6	0.2	0
Importance of reproducibility	20	x					1.67	0	0	0	1	0	0	0.4
Code should be reviewed	5			x			0	0.17	0.17	0	0	0.4	0.2	0
Advanced measures when basics are implemented	5			x			0	0	0	0	0	0	0	1
Code must contain documentation	16	x					1	0.17	0	0	1.2	0	0	0
Sensitive data make reproducibility harder	6					x	0	0	0.17	0	0	0	0.4	0.2

Code Group	**Quoted amount**	**M**	**S**	**C**	**W**	**-**	**ZZ m**	**ZZ s**	**ZZ C**	**ZZ w**	**LUMC m**	**LUMC s**	**LUMC c**	**LUMC w**
Documentation must contain the reason for a piece of code	8	x					0.17	0.17	0.17	0	0.6	0.2	0	0
Documentation must contain the location of dependencies	1			x			0	0	0	0	0	0.2	0	0
Reproducibility in science is mostly for other researchers	7					x	0	0	0	0	0.2	0.2	0.2	0
Organization should set guidelines for reproducibility	4			x			0	0.17	0	0	0.2	0	0.4	0
Programmers quality influences reproducibility	2					x	0	0	0	0	0	0.2	0	0
Measures to support reproducibility with sensitive data	6		x				0	0	0.17	0	0.2	0.4	0.4	0

Code Group	**Quoted amount**	**M**	**S**	**C**	**W**	**-**	**ZZ m**	**ZZ s**	**ZZ C**	**ZZ w**	**LUMC m**	**LUMC s**	**LUMC c**	**LUMC w**
Programmer is responsible for reproducibility	2			x			0	0	0.17	0	0	0	0	0.2
Code review is optional	1				x		0	0	0	0	0	0	0.2	0
No documentation required	3				x		0	0	0.17	0	0	0	0.2	0
Sensitive data does not affect reproducibility	6					x	0	0	0	0	0	0	0	0
Documentation should contain used variables	10	x					0.83	0	0	0	0.8	0	0.2	0
Code should stick to a format	10	x					0.83	0	0	0	0.4	0	0	0
Agreements support reproducibility	6		x				0.17	0.5	0	0	0	0.2	0	0
Clean code supports reproducibility	6		x				0	0.83	0	0	0.2	0	0	0

Code Group	**Quoted amount**	**M**	**S**	**C**	**W**	**-**	**ZZ m**	**ZZ s**	**ZZ C**	**ZZ w**	**LUMC m**	**LUMC s**	**LUMC c**	**LUMC w**
Team is responsible for guidelines	6		x				0	0.17	0.17	0	0	0.4	0.2	0
Data editing in code improves reproducibility	5		x				0	0.33	0	0	0	0.6	0	0
Uses SQL	1				x		0	0	0	0.17	0	0	0	0
Level of reproducibility depends on impact/probability of reusing	9	x					0	0	1.33	0	0	0	0	0

Table 10: The MoSCoW Prioritization of the code groups. The eight columns on the right show how the code groups are prioritized among each organization

D Invitation letter LUMC researchers

Dear PHM researcher,

I am currently working on my graduation thesis for the ICT in Business and the Public Sector program at Leiden University. As part of this, I am developing a maturity model that assesses computational reproducibility and provides guidance on how it can be improved. Computational reproducibility refers to the extent to which research involving software and data can be reproduced — in other words, when someone else uses the original code and data, they should obtain the same results as in the original analysis.

Recent studies have shown that we are facing a reproducibility crisis, which costs researchers a significant amount of time and money. There is still much to gain in this area, and it would be great to contribute to its improvement.

My research focuses on studies and analyses in the healthcare domain, which is why I hope you would be willing to take some time to assist me. As a foundation for my maturity model, I would like to interview researchers in the healthcare field about their experiences and opinions regarding computational reproducibility. I would very much like to conduct one of these interviews with you. The interview will take approximately one hour and could be of great value to my research. Depending on your preference, it can take place either in person or online. I aim to conduct most of the interviews in April, so if possible, I kindly ask you to respond within two weeks. You can reach me via the email address below. Once the model is developed, a follow-up interview may be requested to review it.

If you wish, you are welcome to forward this invitation to other researchers in your network who have relevant experience in this domain. I hope you are willing to participate in an interview, and I look forward to your response.

Kind regards,

Wessel van Putten
w.van.putten@umail.leidenuniv.nl

E Invitation letter Zorg en Zekerheid employees

Dear colleague,

In addition to working as a data analyst at the Knowledge Centre for the past two years, I am currently conducting my graduation research for the MSc program ICT in Business and the Public Sector at Leiden University. As part of this research, I am developing a maturity model that assesses computational reproducibility and indicates how it can be improved. Computational reproducibility refers to the extent to which research using software and data can be reproduced—meaning that someone else, using the original code and data, would obtain the same results as in the original analysis.

Recent studies have shown that we are facing a reproducibility crisis, which costs researchers a great deal of extra time and money. Moreover, it is not uncommon for it to be difficult to reproduce a colleague's analysis when they are on leave, simply because part of the analysis information is missing. There is still much to be gained in this area, and it would be great to contribute to that improvement.

My research focuses on studies and analyses in the healthcare domain, and I hope you are willing to set aside some time to help me. As a basis for my maturity model, I would like to interview researchers and analysts in the healthcare sector about their experiences and views on computational reproducibility. I would very much like to conduct one of these interviews with you. The interview will take approximately one hour and can be of great value to my research. It can take place either online or at the office. After the model has been developed, you may also receive an invitation for a follow-up interview to review the model.

I hope you are available to be interviewed and to contribute to improving the reproducibility of our analyses. I look forward to hearing when you would be available for an interview.

Best regards,

Wessel van Putten

F Interview Guide

Computational reproducibility is about how research with software and data is reproducible, meaning that when anyone uses someone else's code and data, the generated output will be the same as in the original analysis.

Zorg en Zekerheid	At the moment, using data becomes more natural within our work. To make the data visible and useable, code has to be written. To boost the reliability of data and analysis on that data, the code should be reproducible. That enables colleagues to get to same results as originally. Reproducibility is extra important when the original programmer is unavailable due to illness or being on holiday for example.
Researchers LUMC	Recent research shows that there is a reproducibility crisis. Research that isn't reproducible brings unnecessary costs. That is because the researchers has to spent more time on executing the research. In addition to that, it may be possible that the research can not be completed at all. Besides having wasted time and money, this may hinder new discoveries.

1. How long do you work in your current job and what have you done in the past?
2. In what manner is computational reproducibility important for your job
3. How often can you reproduce someone else's work without having to change a lot of the code?
4. How often do you have to adjust a lot of the code in order to reproduce someone's code? Can you express both in fractions?
5. How do you make sure that your own work is computational reproducible?

Challenges

1. Which problems do you encounter while reproducing someones work
2. What were the concequences of these problems?
3. What does your organization/you do to limit these concequences or stimulate computational reproducibility

4. Does the sensitiviy of healt care data make reproducibility harder?

Control

1. How do you decide whether software is reproducible enough
2. What factors should be evaluated to assess reproducibility?
3. Which of these factors need most attention

Documentation of both code and data

1. How important is detailed documentation when reproducing someone's work or doing an analysis on an existing dataset?
2. Which contents does your documentation have
3. Do you have a specific format for your documentation?
4. Which problems do you encounter while reading someone else's documentation?
5. Do you have rules to avoid/solve these problems?
6. After editing the code, is the documentation editted?

Version control

1. How is version control currently performed (GIT, filenames etc...)
2. How does version control affect reproducibility
3. Which problems have you encountered in relation with version control

Datamanagement

1. How do you make sure that the data you are working with stays reproducible?
2. How do you keep track of changes in the data
3. Do you encounter any challenges in relation to laws or rules?
4. What do you do to conform to these laws or rules?

Responsibility

1. Which department/team should be responsible for implementing measures about computational reproducibility?
2. Should this be a team task or should there be specific roles for reproducibility?

G From code to Maturity Model

Code group	Description	How used
Using GIT for version control	An organization or team implements GIT in order to improve their version control	Capability of version control
Documentation should stick to a format	When writing documentation a standard file/format can be used. This lowers the treshold to start documenting and lowers the risk of forgetting to document important elements. On the other side, the reader is more familiar and can find what he or she is looking for	Capability of documentation
Version control supports reproducibility	The organization should set up version control in order to keep track of when and how code has been changed	Focus Area
Importance of reproducibility	Reproducibility is important such that scientific work can be verificated and is thrustable. In corporate environments, computational reproducibility enables colleagues to collaborate better and to take over someone else's work	Not used as focus area or capability, covered by other FA's
Code should be reviewed	Code reviewing is a low effort way of checking whether code works how it should thus whether it is reproducible	Focus Area
Advanced measures when basics are implemented	When the more basic/essential measures are implemented, reproducibility can be improved by implementing more advanced measures, those include storing software as a package and using virtual environments where all required input is already active	Focus Area

Code group	Description	How used
Code must contain documentation	To make sure that someone else is capable of understanding what the code does and what the purpose of the code is, documentation should be written	Focus Area
Sensitive data make reproducibility harder	Sensitive data limits the way of which data is allowed to be shared, due to the fact that it includes data on a personal level. Measures must be taken to find a mix between sharing data and mainting privacy	Capability of sensitive data measures
Documentation must contain the reason for a piece of code	To make testing and further development, it is important that the documentation contains why a specific choise was made, for example filtering out specific data	Focus Area
Documentation must contain the location of dependencies	Missing dependencies like external data or libraries/packages is one of the most occuring causes for not working code. Therefore, documentation must contain where does dependencies come from	Focus Area
Reproducibility in science is mostly for other researchers	When researchers make their work reproducible, it is done such that other researchers can continue with their work, whilst they don't directly benefit from in theirselves. Therefore it is often seen as wasted time	Not used
Organization should set guidelines for reproducibility	When an organization sets guideliness for reproducibility, there will be less discussion about the quality of reproducibility and there are strict rules and standards	Focus Area

Code group	Description	How used
Programmers quality influences reproducibility	Reproducibility is influenced by how good a programmer is. Bad programmers are expected to need more cumbersome code and more dependencies which increase the risk of less reprodociblility	Focus Area
Measures to support reproducibility with sensitive data	Sensitive data limits the way of which data is allowed to be shared, due to the fact that it includes data on a personal level. Measures must be taken to find a mix between sharing data and mainting privacy	Focus Area
Programmer is responsible for reproducibility	This says that the programmer should be responsible for reproducbility of the software. This can vary from deciding the measures itself to be held responsible of the guidelines are not being followed	Capability of guidelines
Code review is optional	This means that a code review should not be obligated in the reproduciblity process	Capability of reviewing
No documentation required	This means that documentation is not necessary as it is only used when the code does not work directly and thus is not reproducible	Capability of documentation
Sensitive data does not affect reproducibility	When taking the right measures, sensitive data does not affect reproducibility	Capability of sensitive data
Documentation should contain used variables	Missing dependencies like external data or libraries/packages is one of the most occuring causes for not working code. Therefore, documentation must contain where does dependencies come from	Focus Area

Code group	Description	How used
Code should stick to a format	When code has a standard lay-out or format it is easier to reproduce it, this plays a bigger role when code is shared within an organization as all programmers will use the same format	Focus Area
Agreements support reproducibility	When agreements between programmers/teams are made, those can improve the processess that stimulate reproducibility	Focus Area
Clean code supports reproducibility	When code is free of unnaccesary rules or silly variables, it makes reproducibility easier	Focus Area
Team is responsible for guidelines	The team is responsible for developing guidelines about how computational reproducibility is implemented	Capability in guidelines
Data editing in code improves reproducibility	When data is changed for an analysis, this should be done in the same file as where other code is placed, instead of doing it in a program like Excel. This must be done because otherwise there are chances that data editing steps are missed	Focus Area
Uses SQL	Team uses SQL to program	Not included in model
Level of reproducibility depends on impact/probability of reusing	The impact of an analysis varies. If it is only needed once and for internal use, making it reproducible is not always necesarry	Focus Area

Table 12: Overview of how the code groups are used in the final maturity model

H Maturity Model before Focus Session

Focus Area: Maturity Level:	0	1	2	3	4	5	6	7	8	9	10	11	12
Organizational													
1. Code Reviews	A		B			C			D			E	
2. Organizational Guidelines	A		B				C						D
3. Training for Programmers	A				B				C				
4. Agreeing on Measures	A			B		C				D			
5. Complexity/Impact dependent Measures	A		B				C		D				
Technical													
6. Advanced Features	A				B								C
7. Reason of Code in Documentation	A			B									
8. Dependencies in Documentation	A			B	C			D		E			
9. Variables in Documentation	A			B					C				
10. Clean code on delivery	A				B	C	D						
11. Manipulating Data	A	B			C					D	E		
Organizational and Technical													
12. Version Control	A												
13. Documentation	A				B				C				D
14. Data sharing measures	A		B			C			D				E
15. Code Standardization	A				B								

Table 13: Maturity Model Before Evaluation

Focus area	**A**	**B**	**C**	**D**	**E**
1	There is no code review performed	Before a coding project is finalized a colleague runs the code	The code is run in a clean compiler without packages being installed	The code review is performed with the help of a pre defined checklist	Besides checking whether the code functions, other guidelines like format, privacy and security are checked as well

Focus area	**A**	**B**	**C**	**D**	**E**
2	Organization does not have guidelines regarding computational reproducibility	Organization does not directly have guidelines but gives freedom to teams	Organizations have a vision/demands in reproducibility, teams are free in how they adhere to it	Organizations have guidelines on how to reach computational reproducibility	
3	Less skilled programmers are not differed from better programmers	Less skilled programmers receive passive training to improve their skills (like datacamp licenses)	Less skilled programmers are linked with better programmers on projects		
4	There are no agreements about how code should be made reproducible	Some agreements are made	Agreements are well documented and accessible for everyone	Agreements are checked by colleagues	
5	No difference in impact of analysis	Less impactful analysis are excluded from reproducibility rules	Project/Analysis has to be classified in level of impact by someone else	Different reproducibility rules apply for each level of impact	
6	No advanced measures are being taken	Code is saved as a package including the used data and dependencies	A virtual environment is created for each coding project such that all dependencies are present		

Focus area	A	B	C	D	E
7	Documentation doesn't contain the reason for the choices	Documentation contains why specific choices in statements are made			
8	Documentation does not cover the location of dependencies	Documentation contains the source of external data	Documentation contains an overview of used packages/libraries	Documentation shows the used version of the used packages/libraries	Documentation includes the version of the programming language
9	No guidelines about how variables should be documented	Documentation contains which variables are used and can be changed	It is documented how each variables should be changed and when they should be changed		
10	No guidelines on cleaning code	Unused parts of code are removed	The code does not contain unused libraries/-packages	Code is straight to the point and uses recognizable variable names	
11	No guidelines of logging data changes	Data can be changed outside of the code	Changes to the data can be done externally but are well documented	Changes to the data are made in a different code file	All manipulation steps are done in the same file as where the analysis is made
12	No version control is applied	A text file is used logging each new version	New versions are stored under a different name	Teams store their code in a version control system with GIT	The team has guidelines for how their code should be stored in GIT

Focus area	A	B	C	D	E
13	No or optional documentation	Code contains inline documentation of code parts	A separate text file is created with information of the code	Documentation follows a structured format	
14	Nothing is shared when working with sensitive data	Code is shared with others, without data being issued	Code is shared, limited information on the dataset is shared, like structure or column names	Data is shared with the code but only in a pseudonymized way ($n \geq 10$ for example)	A synthetic dataset is shared with the code. Dataset is representative for original data
15	There is no specific format for code	There is a general format for code			

Table 14: Description of the capabilities for each focus area