Author’s response to reviews

Title: Methods for enhancing the reproducibility of biomedical research findings using electronic health records.

Authors:

Spiros Denaxas (s.denaxas@ucl.ac.uk)

Kenan Direk (k.direk@ucl.ac.uk)

Arturo Gonzalez-Izquierdo (a.gonzalez-izquierdo@ucl.ac.uk)

Maria Pikoula (m.pikoula@ucl.ac.uk)

Aylin Cakiroglu (aylin.cakiroglu@crick.ac.uk)

Jason Moore (jhmoore@aupenn.edu)

Harry Hemingway (h.hemingway@ucl.ac.uk)

Liam Smeeth (Liam.Smeeth@lshtm.ac.uk)

Version: 1 Date: 07 Jul 2017

Author’s response to reviews:

Thank you for considering our manuscript and for the helpful feedback and suggestions.

Both reviewers acknowledge the importance of the topic as an unmet need in the field of research using complex EHR records. As stated below, the primary audience of this manuscript is the emerging cohort of health data scientists, researchers with no formal training or exposure to scientific software development drawn from a diverse set of scientific backgrounds (e.g. clinicians) and which are increasingly required to implement complex data management and analytical approaches for EHR data. As such, we present a series of methods and approaches which are both approachable from a technical standpoint and will provide a step-change in terms of how research is undertaken in this field.

We have addressed the entirety of the reviewer’s comments and suggestions as indicated below.
Reviewer #1

I have read the manuscript „Methods for enhancing the reproducibility of biomedical research findings using electronic health records“ with great interest. It explains some useful programming technics to make research with data extracted from electronic health records reproducible. Very valuable examples are provided. This is a very important topic as reproducibility should be the basis of all science.

Thank you very much, we agree!

However, this manuscript lacks the necessary clarity and structure to be published. The authors should state clearly the intended meaning of "reproducibility" in the context of this work. Is reproducibility just providing complete information? Or is "reproducibility" the ability to execute the same code on the same data and reach the same results? Or do the authors intent something along the lines of transferability (e.g. when the code is executed on similar data similar results are obtained)? After a clear definition has been provided, the authors should state which aspect of reproducibility is supported by which recommendation.

The reviewer makes a good point with regards to how “reproducibility” is defined and we would further argue that no common agreed definition of the term currently exists across scientific disciplines. Semantically similar concepts such as “replicability” and “duplication” are often used interchangeably.

In our work, we defined “reproducibility” as the provision of sufficient methodological detail about a study so it could, in theory or in actuality, be exactly repeated by investigators. In the context of EHR research, this would involve the provision of a detailed (and ideally machine-readable) study protocol, information on the phenotyping algorithms used to defined study exposures, outcomes and populations and a detailed description of the analytical and statistical methods used along with software and code. This in turn, will enable independent investigators to apply the same methods on a similar dataset and attempt to obtain consistent results (a process often referred to as “results replicability”). To some extent, reproducibility and replicability are not mutually exclusive but are steps towards the same direction of reproducible research across researchers.

We have revised the manuscript to address the reviewers point and include this explicit definition on the scope of our work in the Backgrounds section of the manuscript. In addition, we have provided a further citation which provides a more elaborate description and review of reproducibility in the sciences (Goodman S., et al., Science Translational Medicine, 2016)
Major points:

- STROBE is only mentioned in the abstract.

Thank you for spotting this omission; we have included STROBE in the main body of the manuscript and provided a citation for it (Von Elm, E. et al., Intl Journal of Surgery, 2014).

- (p. 4 l. 53 ff.) The explanation of modular programming is misleading. Modularity refers to the logical structure of the code (e.g. separation of concerns). The physical layout into one or many files is only secondary.

We agree on the definition of modularity which is the same as the one we were using in the manuscript but have reworded the text to address the reviewer’s comment and provided a further citation (Parnas D.L, Comm ACM, 1972).

- (p. 5 l. 46 ff.) Test driven development is not a category of unit testing. You seem to refer to test automation.

We agree with the reviewer’s comment; we have revised the text in the manuscript to avoid confusion and have provided a further citation (Janzen D. et al., IEEE Computer, 2005).

- (p. 10, l. 15 ff., table 3) In-line code documentation is intermixed with report generation. In R package roxygen2 is used to create in-line code documentation like you display in Box 1. (Check for example how help files are generated for CRAN.) Other languages use similar doxygen dialects.

We have further revised the manuscript and table to reflect this comment and make the distinction between report generation and literate programming distinct. We have corrected Table 3 to include the roxygen2 package.

Minor points:

- (p. 4 l. 39) Your description of code development implies stages that follow after each other (e.g. waterfall model). Modern software development is usually more iterative.

We agree and have revised the text in the manuscript and the Figure 2 caption accordingly to address this point.
The whole sections seems to be limited to functional programming languages. Features of object oriented languages like classes are not mentioned. This matters as modular programming is more evolved in object oriented programming.

In the field of EHR research, the majority of analytical applications produced thus far are in the form of large monolithic scripts. One of the aims of our manuscript is indeed to get researchers exposed to scientific programming techniques and best-practices and adopt them but we consider modular programming (i.e. the separation of code into distinct modules) a better fit and more realistic to achieve by researchers that have not been exposed to programming in depth compared to OO approaches (i.e. the separation of code into objects and the definition of their relationships using concepts such as polymorphism, inheritance and encapsulation).

We have however revised the Modular programming section of the manuscript accordingly and added further details on main concepts of object oriented programming.

The list of methods and approaches is not taken up in the subsections of the methods section. In table 2 this the order is changed.

Thank you for pointing this inconsistency out – we have revised the text and Table 2 accordingly to make headings and wording used across the manuscript consistent.

How does version control relate to metadata? What implementation details are released by CALIBER and again how is this related to version control?

We provide one example of how version control is actively used, in the CALIBER EHR research platform, to enable the versioning of contextual and implementation details on EHR phenotyping algorithms. These algorithms are often developed iteratively and refined when new data or new phenotype definition become available, or changes in the underlying healthcare process model cause the data generation or capture process to change. By storing the algorithms and the metadata (such as for example diagnostic terms and their position, date ranges, and other logical rules and restrictions) within a version control system, researchers can track these changes over time and the reasoning behind them compared to the typical approach where no versioning systems are used.

We have revised the manuscript to add further clarity to this point.
Here you mention several properties of bioconductor that enable reproducible research. (e.g. high quality documentation, programming conventions). These properties could be discussed in the introduction and possibly also in more depth in the main part after the meaning of "reproducibility" is clarified.

We thank the reviewer for the suggestion and have revised the Background section of the manuscript accordingly to take this point into consideration.

Why do platforms such as galaxy increase scalability? Why does virtualization increase scalability?

Domain-specific platforms (i.e. Galaxy) and domain-agnostic workflow systems (i.e. Taverna) enable researchers to break down larger monolithic tasks or experiments into a series of small, repeatable, well defined tasks, each with rigidly defined inputs, parameters, and outputs. This in turn allows them to identify which parts of the workflow are a bottleneck or in some cases which parts could benefit from parallelization that can increase throughput. They also allow the integration of workload managers and complex queuing mechanisms that can also potentially lead to better management of resources and processing throughput.

We have revised the text of the manuscript and added an additional citation in order to address the reviewers comment (de la Garza L. et al., BMC Bioinformatics, 2016).

Why is an extraction process imported into a database? This sounds like an implementation detail specific to your setting.

This was a grammatical error which we have corrected in the revised manuscript.

Very minor, but because this work recommends best practices, I would not use a plain text editor for any serious programming. Even very basic IDEs make one's life much easier simply because of syntax highlighting.

We thank the reviewer for the suggestion; we agree and have revised the manuscript to address this point.

You are probably referring to Table 3 (and not Table 2).

Thank you for spotting this; we have corrected the reference to the table.
Reviewer #2

In this work the authors argue for methods and tools dealing with EHR data. They focus on the development of software in this field. In generally I agree to the statement that coding analytical software for EHR data should be carried out under the same aspects of quality assurance as any other software. But from this work I did not understand what are the specific requirements and constraints in the area of EHC software, that would distinguish it from other software development projects. In the field of software engineering there exist in the meantime a whole bunch of techniques and methods in order to produce software of good quality. This includes methods for analysis, design, coding testing and verification, and documentation. So why no just require the use of standard software development techniques (to phrase it in a short way)? Why are the selected methods (modularity, revision control, tests) are more important in the field of EHC data, than others are?

We recognize and acknowledge the reviewer’s point.

The primary audience of our work presented in this manuscript is the increasingly expanding cohort of health data scientists: researchers that have not been exposed to formal training in computer science or software development but are increasingly required to create and use sophisticated tools to analyze the large and complex EHR datasets made available for research. Phenotyping algorithms, one of the basic building blocks of EHR research, are often non-deterministic pieces of code which get refined over multiple iterations and as such cannot be treated in the same fashion as a monolithic piece of software. As such, we have selected and discuss a number of approaches which, in combination with training on more conventional methods, we considered to be readily usable by scientists and would readily enable them to produce high-quality scientific code.

Why are design techniques, e.g. the use of standardized diagrams as in UML, not mentioned at all, although they could help a lot to improve the quality of documentation?

We have revised the manuscript and to address the reviewers point and have provided an additional citation for the use of UML (Medvidovic N. et al, ACM Transactions on Software Engineering and Methodology, 2002)

Is testing alone sufficient, or are more rigorous methods needed, as e.g. formal verification?

We acknowledge the reviewers point but argue that testing is a sufficient first step to enable EHR researchers to produce high-quality code but we have revised the manuscript and added a further citation pointing to more advances testing strategies (Clarke E. M. et al. ACM Computing Surveys, 1996).
In summary, my impression is that a set of methods and tools is suggested, but it is not really described what are the specific advantages of exactly these methods and tools for the field of EHC. There is no description of what are the specific challenges of EHC software development that makes it different from software development in other areas. This gives the impression of a somehow arbitrary choice.

As previously described, the primary audience of this manuscript are health data scientists without formal exposure to computer science of software engineering. While we acknowledge the reviewers' comment on the availability of many available approaches and methods, we focused and elaborated on the ones that were both considered as readily usable by researchers and were strongly supported by common statistical packages and tools used.

Some more detailed comments:

- page 4: modular programming has been studied thoroughly in computer science leading to terms like encapsulation, abstraction (abstract data types), object-oriented programming etc. Why is this not mentioned?

We consider “modular programming” and “object-oriented programming” as two separate, but related, concepts.

In the field of EHR research, the majority of analytical applications produced thus far are in the form of large monolithic scripts. The primary aim of our manuscript is indeed to get researchers exposed to scientific programming techniques and best-practices and adopt them but we consider modular programming (i.e. the separation of code into distinct modules) a better fit and more realistic to achieve by researchers that have not been exposed to programming in depth compared to OOP approaches (i.e. the separation of code into objects and the definition of their relationships through interfaces and concepts like inheritance, polymorphism etc.).

We have however revised the Modular programming section of the manuscript accordingly and added further details on main concepts of object-oriented programming.

- page 4/5: The use of libraries in R is not the crucial point. You should consider the use of interfaces and libraries that implement certain interfaces.
We acknowledge the reviewer’s comment and the importance of interfaces and libraries however we feel that it’s not entirely relevant to the main audience of our manuscript or the approaches we are presenting in our manuscript.

The current modus operandi of EHR research does not directly involve the standardization or re-use of programming code or algorithms and one of the aims of our work is to enable researchers to adopt best-practices for scientific software development which includes, amongst others, creating and releasing libraries for commonly used statistical analysis applications such as R. An added advantage of using the built-in functionality of packages in common analytical tools will enable researchers otherwise not exposed to software development to create distributable code.

We have further refined the text in the manuscript to add clarity around this issue and add pointers to more advanced concepts such as interfaces.

- page 6: Testing software should be a natural course of action. But wouldn't TDD, in its original meaning, give a benefit. Why do you define it more broader? You should also distinguish unit tests, system tests (tests of integration) and acceptance tests. Units test alone, are not sufficient. Is testing sufficient at all?

We acknowledge the reviewers point but argue that testing is a sufficient first step to enable EHR researchers to produce high-quality code but we have revised the manuscript and added a further citation pointing to more advances testing strategies (Clarke E. M. et al. ACM Computing Surveys, 1996).

- I didn't understand the intention of Box 1? It occurs to me as a simple R function with some simple case-statement? Checking function arguments using if-statements is not really something new, did you ever try using invariants (preconditions, postconditions)?

We acknowledge the reviewers comment but argue that while this might be a simple function and a straightforward example, it effectively illustrates the benefits of using functions as a paradigm to produce modular code that is test-able and maintainable compared to the usual single monolithic file with duplicated snippets.

We have further refined the text in the manuscript to add clarity around this issue and add text and a new citations discussing advanced concepts such as preconditions and postconditions in OOP and the design by contract paradigm.
- Using a source control system in software development is, to my opinion, not worthwhile mentioning. If you mention it you should also consider the problems of parallel development and branching.

We respectfully disagree and argue that in a field such as EHR research where source control systems are not widely used and files are not versioned systematically, a tool like a modern source control system would greatly benefit researchers.

We have however further revised the manuscript to include a very brief summary of branching and its uses. We have additionally expanded the example given in CALIBER of how EHR phenotyping algorithms (which are refined over time as the underlying healthcare process that generates data changes) can be stored and versioned in a source code control system to allow researchers to track changes.

- page 10: In terms of documentation you should consider modelling techniques that use different kinds of diagrams (e.g. stat charts etc.) that could support the traceability and reproducability of results.

We thank the reviewer for the suggestion which we have incorporated in the manuscript.

Why do workflow systems enable the development of scalable tools, as opposed to other approaches?

We chose to elaborate further on workflow systems as they are commonly used in bioinformatics experiments which often face the same challenges in terms of data scale and complexity as EHR researchers and reproducibility is a key aspect of experiments. Domain-specific platforms (i.e. Galaxy) and domain-agnostic workflow systems (i.e. Taverna) enable researchers to break down larger monolithic tasks or experiments into a series of small, repeatable, well defined tasks, each with rigidly defined inputs, parameters, and outputs. This in turn allows them to identify which parts of the workflow are a bottleneck or in some cases which parts could benefit from parallelization that can to increase throughput. They also allow the integration of workload managers and complex queuing mechanisms that can also potentially lead to better management of resources and processing throughput.

We have revised the text of the manuscript and added an additional citation in order to address the reviewers comment (de la Garza L. et al., BMC Bioinformatics, 2016).
Again, there exist many modelling techniques to describe the algorithmic logic behind a solution. In software engineering MDA (model driven architecture) is an approach to develop a solution independently from the particular implementation environment.

As previously described, the primary audience of this manuscript are health data scientists without formal exposure to computer science of software engineering drawn from a diverse set of scientific backgrounds. While we acknowledge the reviewers comment on the availability of many available approaches and methods, we focused and elaborated on the ones that were both considered as readily usable by the wider target audience and were strongly supported by common statistical packages and tools used in the field.

Talking about literate programming: how is about using pseudo-code? This is used in many publications.

We would respectfully argue that in order for proper reproducibility to be achieved, the actual code used in the analyses should be shared rather than pseudo-code. Pseudo-code suffers from the same problems that textual representations of algorithms have, i.e. the translation to machine-code might suffer from ambiguity or biases which in turn comprise its quality and outcome.