

REVIEW

Open Access



# Methods for enhancing the reproducibility of biomedical research findings using electronic health records

Spiros Denaxas<sup>1,2\*</sup> , Kenan Direk<sup>1,2</sup>, Arturo Gonzalez-Izquierdo<sup>1,2</sup>, Maria Pikoula<sup>1,2</sup>, Aylin Cakiroglu<sup>3</sup>, Jason Moore<sup>4</sup>, Harry Hemingway<sup>1,2</sup> and Liam Smeeth<sup>5</sup>

\*Correspondence:

s.denaxas@ucl.ac.uk

<sup>1</sup>Institute of Health Informatics, University College London, 222 Euston Road, NW1 2DA London, UK

<sup>2</sup>Farr Institute of Health Informatics Research, 222 Euston Road, London, UK

Full list of author information is available at the end of the article

## Abstract

**Background:** The ability of external investigators to reproduce published scientific findings is critical for the evaluation and validation of biomedical research by the wider community. However, a substantial proportion of health research using electronic health records (EHR), data collected and generated during clinical care, is potentially not reproducible mainly due to the fact that the implementation details of most data preprocessing, cleaning, phenotyping and analysis approaches are not systematically made available or shared. With the complexity, volume and variety of electronic health record data sources made available for research steadily increasing, it is critical to ensure that scientific findings from EHR data are reproducible and replicable by researchers. Reporting guidelines, such as RECORD and STROBE, have set a solid foundation by recommending a series of items for researchers to include in their research outputs. Researchers however often lack the technical tools and methodological approaches to actuate such recommendations in an efficient and sustainable manner.

**Results:** In this paper, we review and propose a series of methods and tools utilized in adjunct scientific disciplines that can be used to enhance the reproducibility of research using electronic health records and enable researchers to report analytical approaches in a transparent manner. Specifically, we discuss the adoption of scientific software engineering principles and best-practices such as test-driven development, source code revision control systems, literate programming and the standardization and re-use of common data management and analytical approaches.

**Conclusion:** The adoption of such approaches will enable scientists to systematically document and share EHR analytical workflows and increase the reproducibility of biomedical research using such complex data sources.

**Keywords:** Electronic health records, Reproducibility, Transparency, Biomedical research

## Background

Electronic health records (EHR), data generated and captured during routine clinical care encounters across health care settings, have been recognized as an invaluable research resource [1, 2]. Increasingly, EHR are linked with genotypic data to enable precision medicine by examining how genetic variants influence susceptibility towards disease, validate drug targets, or modify drug response at a scale previous unobtainable. EHR

research resources such as CALIBER [3], the Precision Medicine Cohort Initiative [4], the UK Biobank [5] and the eMERGE Network [6] enable researchers to investigate disease aetiology and prognosis [7–14] at unprecedented phenotypic depth and breadth by recreating the longitudinal patient pathway spanning genome to phenome and birth to death.

The replication of scientific findings using independent investigators, methods and data is the cornerstone of how published scientific claims are evaluated and validated by the wider scientific community [15–17]. Academic publications arguably have three main goals: a) to disseminate scientific findings, b) to persuade the community that the findings are robust and were achieved through rigorous scientific approaches and c) to provide a detailed description of the experimental approaches utilized. Peer-reviewed manuscripts should, in theory, describe the methods used by researchers in a sufficient level of detail as to enable other researchers to replicate the study.

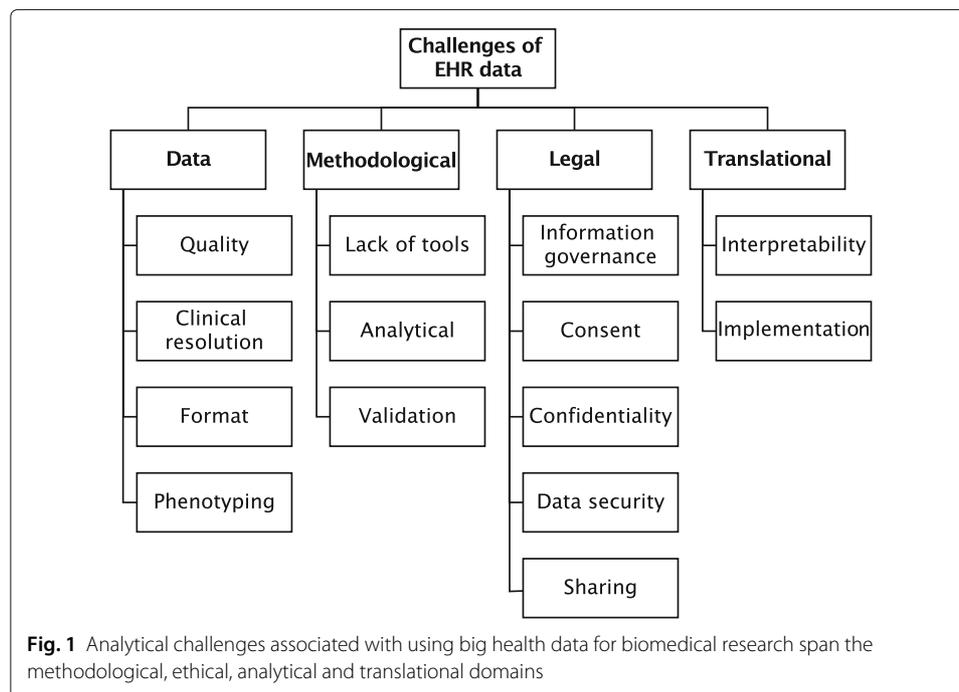
A recent literature review [18] of research studies using national structured and linked UK EHR illustrated how a substantial proportion of studies potentially suffer from poor reproducibility: only 5.1% of studies published the entire set of controlled clinical terminology terms required to implement the EHR-derived phenotypes used. Similar patterns were discovered in a review of over 400 biomedical research studies with only a single study making a full protocol available [17]. With the volume and breadth of scientific output using EHR data steadily increasing [19], this nonreproducibility could potentially hinder the pace of translation of research findings.

No common agreed and accepted definition of *reproducibility* currently exists across scientific disciplines. Semantically similar concepts such as “replicability” and “duplication” are often used interchangeably [20]. In this work, we define *reproducibility* as the 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, covariates and populations and a detailed description of the analytical and statistical methods used along with details on the software and the programming 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*).

### **Electronic health record analytical challenges**

EHR data can broadly be classified as: a) structured (e.g. recorded using controlled clinical terminologies such as as the International Classification of Diseases—10th revision (ICD-10) or Systematic Nomenclature of Medicine – Clinical Terms (SNOMED-CT [21]), b) semi-structured (e.g. laboratory results and prescription information that follow a loose pattern that varies across data sources), c) unstructured (e.g. clinical text [22]) and d) binary (e.g. medical imaging files). Despite the numerous advantages EHR data offer, researchers face significant challenges (Fig. 1).

A primary use-case of EHR data is to accurately extract phenotypic information (i.e. disease onset and progression), a process known as *phenotyping*, for use in observational and interventional research [5]. Phenotyping however is a challenging and time-consuming process as raw EHR data require a significant amount of preprocessing before they can

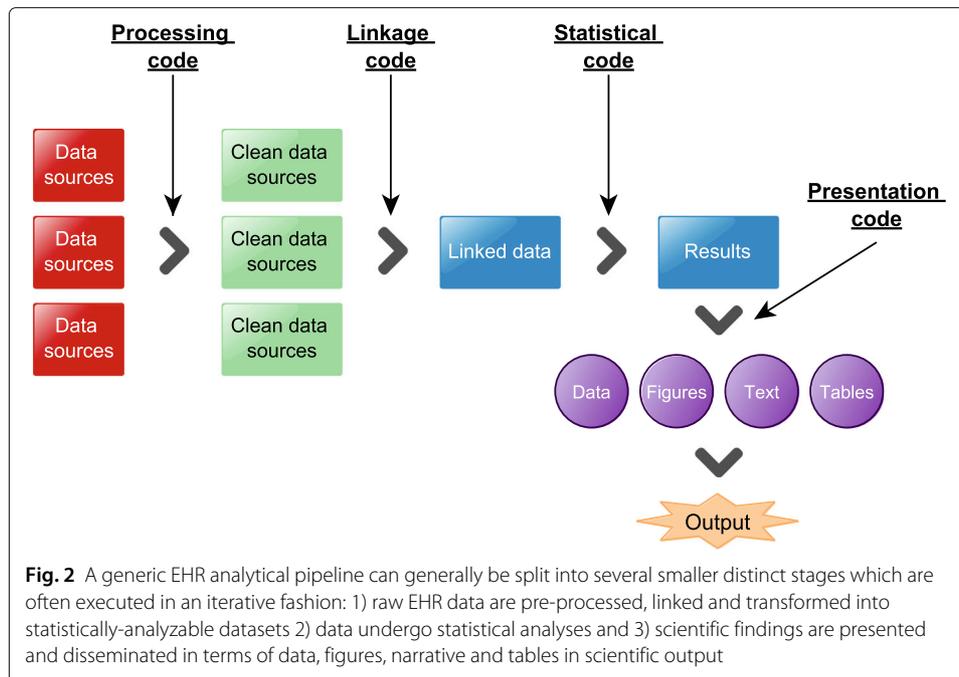


be transformed into research-ready data for statistical analyses [23]. The context and purpose in which data get captured (e.g. clinical care, audit, billing, insurance claims), diagnostic granularity (e.g. post-coordination in SNOMED-CT vs. fixed-depth in ICD) and data quality vary across sources [24].

EHR data preprocessing however, is not performed in a reproducible and systematic manner and as a result, findings from research studies using EHR data potentially suffer from poor reproducibility. Phenotyping algorithms defining study exposures, covariates and clinical outcomes are not routinely provided in research publications or are provided as a monolithic list of diagnostic terms but often miss critical implementation information. For example, a phenotyping algorithm using diagnostic terms in hospital care should consider whether a term is marked as the primary cause of admission or not but this important distinction is often omitted from manuscripts. Common data manipulations (Fig. 2) on EHR datasets are repeated *ad nauseam* by researchers but neither programmatic code nor data are systematically shared. Due to the lack of established processes for sharing and cross-validating algorithms, their robustness, generalizability and accuracy requires a significant amount of effort to assess [25]. In genomics for example, cross-referencing annotations of data produced by related technologies is deemed essential [26] (e.g. reference Single Nucleotide Polymorphism (SNP) id numbers, genome annotations), but such approaches are not widely adopted or used in biomedical research using EHR data.

#### Electronic health records research reproducibility

Significant progress has been achieved through the establishment of initiatives such as REporting of studies Conducted using Observational Routinely-collected Data (RECORD) [27, 28] and the STrengthening the Reporting of OBservational studies in Epidemiology (STROBE) [29, 30]. RECORD and STROBE are international guidelines



for studies conducted using routinely-collected health data. The guidelines focus on the systematic reporting of implementation details along the EHR analytical pipeline: from study population definitions and data linkage to algorithm details for study exposures, covariates and clinical outcomes (Table 1). Often however, researchers lack the tools and methods to actuate the principles behind these guidelines and fail to integrate them into their analytical process from the start but rather try to incorporate them before publication in an ad hoc fashion. This lack of familiarity with best practices around scientific software development tools and methods prevents researchers from creating, maintaining and sharing high-quality EHR analytical pipelines enabling other researchers to reproduce their research.

We argue that EHR research can greatly benefit from adopting practices used in adjunct scientific disciplines such as computer science or computational biology in

**Table 1** Reporting of studies Conducted using Observational Routinely collected Data (RECORD) recommendations on reporting details around EHR algorithms used to define the study populations, exposures and outcomes

RECORD guideline principle id number	Description
6.1	The methods of study population selection (such as codes or algorithms used to identify subjects) should be listed in detail.
7.1	A complete list of codes and algorithms used to classify exposures, outcomes, confounders, and effect modifiers should be provided.
13.1	Describe in detail the selection of the persons included in the study (i.e., study population selection) including filtering based on data quality, data availability and linkage.
22.1	Authors should provide information on how to access any supplemental information such as the study protocol, raw data or programming code.

order to reduce the potential irreproducibility of research findings using such complex data sources. In this manuscript, we review and identify a series of methods, tools and approaches used in adjacent quantitative disciplines and make a series of recommendations on how they can be applied in the context of biomedical research studies using EHR. These can be used to potentially address the problem of irreproducibility by enabling researchers to capture, document and publish their analytical pipelines. Where applicable, we give examples of the described methods and approaches in R. Adopting best-practices from scientific software development can enable researchers to produce code that is well-documented, robustly tested and uses standardized programming conventions which in turn extend its maintainability. The primary audience of our work is the increasingly expanding cohort of *health data scientists*: researchers from a diverse set of scientific backgrounds (such as for example clinicians or statisticians) that have not been exposed to formal training in computer science or scientific software development but are increasingly required to create and use sophisticated tools to analyze the large and complex EHR datasets made available for research.

## Methods and results

We searched published literature, gray literature and Internet resources for established approaches and methods used in computer science, biomedical informatics, bioinformatics, computational biology, biostatistics, and scientific software engineering. We evaluated and described the manner in which they can be used for facilitating reproducible research using EHR and address the core challenges associated with this process. Reproducibility has been identified as a key challenge and a core value of multiple adjunct scientific disciplines e.g. computer science [31–33] bioinformatics [34], microbiome research [35], biostatistics [36], neuroimaging [37] and computational biology [38].

We identified and evaluated the following methods and approaches (Table 2):

1. *Scientific software engineering principles*: modular and object oriented programming, test-driven development, unit testing, source code revision control;
2. *Scalable analytical approaches*: standardized analytical methods, standardized phenotyping algorithms and
3. *Literate programming*

### Scientific software engineering

The nature and complexity of EHR data often requires a unique and diverse set of skills spanning medical statistics, computer science and informatics, data visualization, and clinical sciences. Given this diversity, it's fair to assume that not all researchers processing and analysing EHR data have received formal training in scientific software development. For the majority of researchers, unconscious practices can creep into the developed code, which if never made publicly available, will never be discovered and yet underpin most published scientific claims. No researcher wants to be put into the position of retracting their manuscript from a journal or having to contact a scientific consortium to ask they repeat months of analyses due to an error discovered in their analytical code. While these issues are not unique in EHR research, they are amplified given its multidisciplinary nature.

**Table 2** Methods and approaches that can enable the reproducibility of biomedical research findings using electronic health records

Method/approach	Recommendations
Scientific software engineering principles	<p>Create generic functions for common EHR data cleaning and preprocessing operations which can be shared with the community</p> <p>Produce functions for defining study exposures, covariates and clinical outcomes across datasets which can be maintained across research groups and reused across many research studies</p> <p>Create modules for logically grouping common EHR operations e.g. study population definitions or datasource manipulation to enable code maintainability</p> <p>Create tests for individual functions and modules to ensure the robustness and correctness of results</p> <p>Track changes in analytical code and phenotyp definitions using controlled clinical terminology terms by making use of a source code revision control system</p> <p>Use formal software engineering best-practices to document workflows and data manipulation operations</p>
Standardized analytical approaches	<p>Build and distribute libraries for common EHR data manipulation or statistical analysis and include sufficient detail (e.g. command line arguments) for all tools used</p> <p>Produce and annotate machine-readable EHR phenotyping algorithms that can be systematically curated and reused by the community</p> <p>Use Digital Object Identifiers (DOIs) for transforming research artifacts into shareable citable resources and cross-reference from research output</p> <p>Deposit research resources (e.g. algorithms, code) in open-access repositories or software scientific journals and cross-reference from research output</p> <p>Virtual machines can potentially be used to encapsulate the data, operating system, analytical software and algorithms used to generate a manuscript and where applicable can be made available for others to reproduce the analytical pipeline.</p>
Literate programming	<p>Encapsulate both logic and programming code using literate programming approaches and tools which ensure logic and underlying processing code coexist</p>

There is a subtle but prevalent misconception that analytical code does not constitute *software* as it's written for a statistical package (e.g. R [39] or Stata [40]) and not in a formal programming language (e.g. Python [41] or Java [42]). As a result, the majority of researchers inadvertently fail to acknowledge or adopt best-practice principles around scientific software engineering. This could not be further from the truth as, by definition, code written for transforming raw EHR into research-ready datasets and undertaking statistical analyses is both complex and sophisticated due to the inherent complexity and heterogeneity of the data. While not directly a technical solution, facilitating scientists to obtain up to date training in best practices through training initiatives such as Software Carpentry [43], can potentially enable them to produce better quality code.

There is no optimal manner in which scientific software can be developed for tackling a particular research question as this is intrinsically an extremely problem-dependent set of tasks. Adopting scientific software engineering best practices however can provide EHR researchers with the essential bedrock of producing, curating and sharing high-quality analytical code for re-use by the scientific community. In general, scientific code

development can be divided into several different phases which are usually executed and evaluated in smaller, rapid iterations: planning, coding, testing, debugging, documentation and analysis. In each of these phases, a number of tools and methods exist that enable researchers to manage provenance, readability and usability of their code. We review several of the most critical ones: modular programming, test-driven development and source code revision control in sections below.

### **Modular and object oriented programming**

Adopting a modular programming approach will allow EHR researchers to organize their code more efficiently and subsequently enable its documentation and re-use, both by them and other researchers. Modular programming is essentially a software design technique that emphasizes separating the functionality of a program into smaller, distinct, independent and interchangeable modules [44]. This translates to splitting code (that is often produced as a single, large, monolithic file) into several smaller modules that contain similar operations or concepts and that in turn can be stored as independent source code files. These modules can contain anything from a collection of functions to process a particular set of data fields (e.g. convert laboratory test measurements from one measurement unit to another) or entire modules dealing with the intricacies of one particular data source (e.g. extract and rank the causes of hospitalization from administrative data).

Common EHR data transformation or analysis operations (Table 3) can be created as functions which can then be shared across modules. Defining functions that can be repeated across different modules significantly reduces the complexity and increases the maintainability of code. The majority of software applications used in EHR research allow both the sourcing of external files and libraries. For example, in R, the *source* command sources an external R file and the *library* command loads an external library into the current namespace. Functions can be easily defined using the *function* command.

Adjacent to modular programming is the concept of object oriented programming (OOP) [45]. OOP is a software programming approach based on the concept of *objects* which contains both data (*attributes*) and procedural code (*methods*) to work on the data and can interact with other objects. Formal definitions of objects (i.e. available attributes and methods) are provided by *classes* and *objects* themselves are instances of classes. Central to OOP is the concept of *encapsulation* which abstracts the data and methods of an object from other objects which are only allowed to interact with them through a predefined template called an *interface*. Interfaces in OOP are a paradigm which allows the enforcement of certain predefined properties on a particular class object. Finally, the concept of *inheritance* allows objects to be organized in a hierarchical manner where each level defined a more specific object than the parent level and inherits all the attributes and methods of the parent level. Methods of classes can have a number of preconditions and postconditions defined i.e. predicates that must always hold true just before or right after the execution of a piece of code or the execution is invalidated [46]. Finally, formal software design modelling languages, such as the Unified Modelling Language (UML), [47] can assist researchers in designing and visualizing complex software applications and architectures. Furthermore, modelling languages can be used as a common point of reference and communication across multidisciplinary EHR research groups as they provide non-technical, unambiguous graphical representations of complex approaches. An example of a very simple UML class diagram is provided in Fig. 3.

**Table 3** Example of an R function for converting lipid measurements between mmol/L and mg/dL units

```

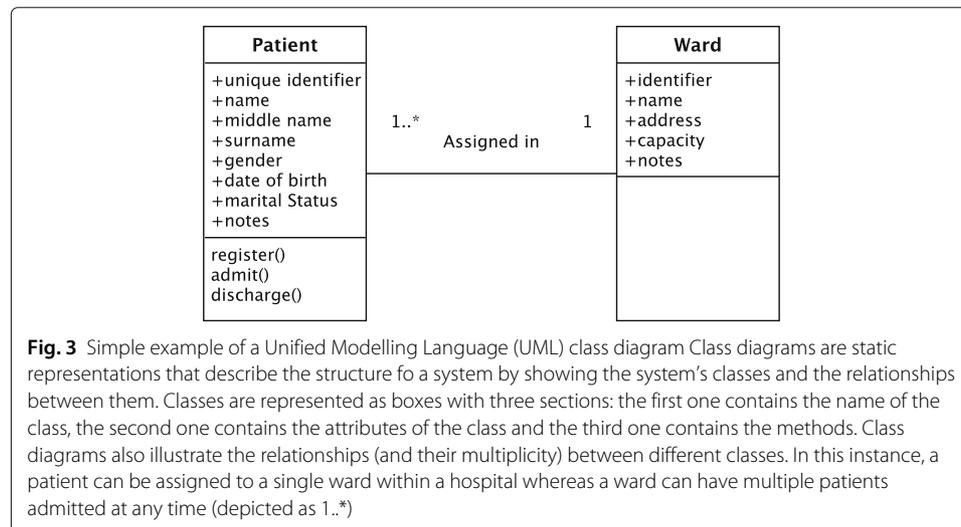
#’ Convert lipid measurements between units mmol/L and mg/dL
#’ @param value the value to be converted
#’ @param units the units to be converted to (mmol/L or mg/dL)
#’ default is to accept mmol/L and convert to mg/dL
#’ @return numeric
#’ @export convert_lipid_measurement
convert_lipid_measurement <- function(value, to_units='mg/dL'){
  if ( missing(value) ){
    stop("Please specify a lipid value.")
  }
  if ( value <= 0 ){
    stop("Invalid value.")
  }
  if ( to_units == 'mg/dL'){
    return( value * 38.67 )
  } else if ( to_units == 'mmol/L' ) {
    return( value / 38.67 )
  } else {
    stop("Units not supported: use mg/dL or mmol/L")
  }
}

```

Function arguments (value and units) are validated prior to performing the calculation and an error is raised if incorrect or missing parameters are supplied

**Test-driven development and unit testing**

Test-driven development (TDD) is a software development approach where automated tests are created prior to developing functional code [48]. In addition to testing, TDD involves writing automated tests of a program’s individual units, a process defined as *unit testing* [49]. A unit is the smallest logical component of a larger software application that can be tested. The majority of tools and languages used in EHR research have some



**Fig. 3** Simple example of a Unified Modelling Language (UML) class diagram Class diagrams are static representations that describe the structure for a system by showing the system’s classes and the relationships between them. Classes are represented as boxes with three sections: the first one contains the name of the class, the second one contains the attributes of the class and the third one contains the methods. Class diagrams also illustrate the relationships (and their multiplicity) between different classes. In this instance, a patient can be assigned to a single ward within a hospital whereas a ward can have multiple patients admitted at any time (depicted as 1..\*)

mechanism to directly facilitate code testing. For example, in Stata code testing can be implemented using the *testcase* class in Mata while in SAS [50] unit testing is facilitated through the FUTS [51] or SASUnit [52] libraries. TDD can enable complex EHR pre-processing and analytical code to be adequately and thoroughly tested iteratively over the lifetime of a research project.

Several R libraries exist (e.g. *testthat* [53], *RUnit* [54] and *svUnit* [55]) that enable researchers to create and execute unit tests. *RUnit* and *svUnit* are R implementations of the widely used *JUnit* testing framework [56] and contain a set of functions that check calculations and error situations (Table 4). Such tests can then be integrated within a continuous integration framework [57], a software development technique that enables the automatic execution of all tests whenever an underlying change in the source code is made and in a way ensures that errors are detected earlier. More advanced methods, such as executing formal software verification methods [58] against a predefined specification can be useful for larger and more complex projects.

**Table 4** Using the *RUnit* library to perform unit tests for a function converting measurements of lipids from mmol/L to mg/dL

---

```
library(RUnit)
test.lipids <- function() {
  checkEquals(
    convert_lipid_measurement(1,to_units = 'mg/dL' ),38.67
  )
  checkEquals(
    convert_lipid_measurement(1,to_units = 'mmol/L'), 0.02585984
  )
  checkEquals(
    convert_lipid_measurement(1, ), 38.67
  )
  checkException(
    convert_lipid_measurement(), FALSE
  )
  checkException(
    convert_lipid_measurement(1, to_units = 'random'), FALSE
  )
  checkException(
    convert_lipid_measurement(-100), FALSE
  )
}

test.lipids()
Error in convert_lipid_measurement() : Please specify a lipid value.
Error in convert_lipid_measurement(1, to_units = "random") :
  Units not supported: use mg/dL or mmol/L
Error in convert_lipid_measurement(-100) : Invalid value.
[1] TRUE
```

---

Combinations of valid, invalid, and missing function parameters are tested and the output returned from the function is examined

### **Source code revision control**

EHR research invariably generates a substantial amount of programming code across the entire EHR pipeline. Even minor changes, accidental or intended (e.g. updating a disease exposure definition), in the code can have large consequences in findings. Given the collaborative and iterative nature of EHR research, it is essential for researchers to have the ability to track changes in disease or study population definitions over time and share the code used in a transparent manner. The standard solution for tracking the evolution of code over time is to use a version control system (Table 5) such as Git [59] or subversion [60]. Version control systems, widely used in software engineering, are applications that enable the structured tracking of changes to individual text-based files both over time and across multiple users [61]. Version control also enables *branching*: the duplication of the core code for the purpose of parallel development independent of the parent code-base. Branching enables the isolation of code for the purposes of altering or adding new functionality or implementing different approaches. In the case of phenotyping algorithms, several branches of the same project may contain alternate implementations of the algorithm with the same core features but slight variations. If only one approach is needed in the end, the relevant branch can then be merged back into the main working code-base (Fig. 4). The use of version control systems in EHR research can enable researchers to keep versioned implementations of exposure, outcomes and phenotype definitions and document the reasons for any changes over time. Computable versions of phenotyping algorithms [62] can also be stored within a version control system for the same reasons.

Phenotyping algorithms defining disease cases and controls are often developed iteratively and refined when new data become available or changes in the underlying healthcare process model cause the data generation or capture process to change. In the CALIBER EHR research platform for example [3], phenotyping algorithms and their associated metadata are stored and versioned in a private version control system. This includes the actual SQL code for querying the raw data, the implementation details and logic of the algorithm, the diagnostic terms and their relative position used and any other relevant metadata (such as author, date of creation, date of validation) in a bespoke text-based format. This enables researchers to keep track of changes of definitions at the desired time granularity and facilitates the collaborative creation of algorithms. The metadata and implementation details are then made available through the CALIBER Data Portal [63] for other researchers to download and use.

### **Scalable analytical approaches**

#### ***Standardized analytical methods***

Scientific software is often at first developed behind closed doors and public release is only considered around or after the time of publication [64]. The standardization of common analytical approaches and data transformation operations in EHR research will potentially enable the reproducibility of scientific findings and fuel a sustainable community around the use of EHR data for research. Adjunct scientific disciplines have adopted this principle through the creation of large software libraries that contain a variety of common analytical approaches [65, 66]. For example, Bioconductor [67] was established in 2001 as an open source software for performing bioinformatics operations based on R. It serves as a common software platform that enables the development and implementation of open and accessible tools. Bioconductor promotes high quality

**Table 5** Example of using git to initialize an empty repository and track changes in a versioned file defining a study cohort

---

```
// Initialize an empty repository
$git init .
Initialized empty Git repository in lipid-cvd-analyses/.git/

// Add a file (defineCohort.R) to the repository.
$git commit -am "Initial cohort definition file"
[master (root-commit) 8ffd1bc] Initial file
 1 file changed, 1 insertion(+)
 create mode 100644 defineCohort.R

// Perform some changes to the file and commit the revised file.
$git commit -am "Changed cohort definition"
[master ae09d9c] Changed cohort definition
 1 file changed, 1 insertion(+), 1 deletion(-)

// Print a log of commits in reverse chronological order.
$git log defineCohort.R
commit ae09d9c2a14498121df6f79dac48b1334e0bf3c1
Author: spiros <spiros@example.com>
Date:   Mon Mar 20 11:40:02 2017 +0000

    Changed definition - include patients 18 years old

commit 8ffd1bc00bce8bb6acae245c36005b7d4034bb76
Author: spiros <spiros@example.com>
Date:   Mon Mar 20 11:39:30 2017 +0000

    Initial file

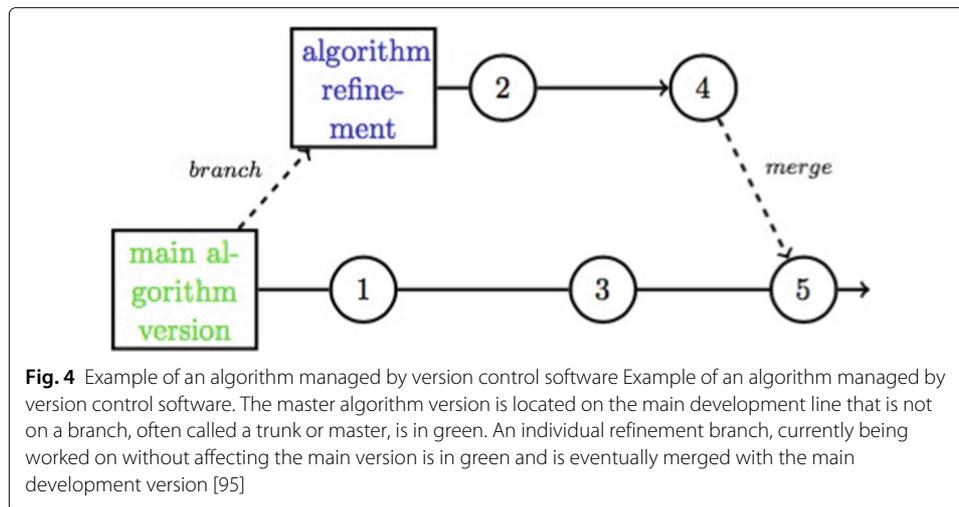
// Display differences between two file versions. Hunk headers
// enclosed in @@ denote the location of the change within
// each file. Lines which are common to both files begin with
// a space character. If a line has been added it begins
// with + while a file removed begins with -.

$git diff ae09d9c2a14498121df6f79dac48b1334e0bf3c1
8ffd1bc00bce8bb6acae245c36005b7d4034bb76 defineCohort.R

diff --git a/file2 b/file2
index 268faed..f0af9ec 100644
--- a/ defineCohort.R
+++ b/ defineCohort.R
@@ -1,2 +1 @@
 patientLowerAge <- 30
-patientLowerAge <- 18
```

---

documentation, and enables standard computing and statistical practices to produce reproducible research. The documentation across sections of each project is clear, accurate and appropriate for users with varying backgrounds on the programming languages and analytic methods used. There is particular emphasis on programming conventions,



**Fig. 4** Example of an algorithm managed by version control software. The master algorithm version is located on the main development line that is not on a branch, often called a trunk or master, is in green. An individual refinement branch, currently being worked on without affecting the main version is in blue and is eventually merged with the main development version [95]

guidance, and version control, all of which greatly benefit from being performed in a standardized manner.

Adopting similar approaches in EHR research can arguably allow researchers to use and re-use standardized data cleaning, manipulation and analysis approaches. The reproducibility pipeline may require a more explicit structure where specific analytic workflows are tied to complete processes illustrating the decision tree from data preparation to data analysis. The ability to reproduce biomedical research findings depends on the interconnection of stages such as detailing the data generation processes, phenotyping definitions and justifications, different levels of data access where applicable, specifics on study design (e.g. matching procedures or sensitivity analyses) and the statistical methods used. Building generic and re-usable software libraries for EHR data is challenging due to the complexity and heterogeneity across data sources. While some libraries for manipulating and analysing EHR data exist [68], these are narrowly focused on specific data sources are challenging to generalize across other sources, countries or healthcare systems. Building and curating software libraries following the best practices outlined in this manuscript and disseminating them with standard scientific output is recommended in order to grow and sustain a community of tools and methods that researchers can use. Examples such as Bioconductor can offer inspiration on how to build an active community around these libraries that will facilitate and accelerate their development, adoption and re-use by the community.

A key aspect of developing software tools for data processing is estimating the expected data growth and designing modules and tools accordingly to accommodate future increases in data. Given the steady increase in size and complexity of EHR data, workflow management systems used in bioinformatics such as Galaxy [69], Taverna [70], and Snakemake [71] can enable the development of scalable approaches and tools in EHR research. Workflow management systems enable researchers to break down larger monolithic tasks or experiments into a series of small, repeatable, well defined tasks, each with rigidly defined inputs, run-time parameters, and outputs. This allows researchers to identify which parts of the workflow are a bottleneck or in some cases which parts could benefit from parallelization 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 [72]. Pipelines can be built to obtain snapshots of the data, validate using predefined set of rules or for consistency (e.g. against controlled clinical terminologies) and then transform into research-ready datasets for statistical analysis. Such pipelines can then potentially be shared and distributed using container technologies such as Docker [73] or package managers like Conda [74]. Docker is an open-source platform that uses Linux Containers (LXC) to completely encapsulate and make software applications portable. Docker containers require substantially less computational resources than virtual machine-based solutions and allow users to execute applications in a fully virtualized environment using any Linux compatible language (e.g. R, Python, Matlab [75], Octave [76]). Docker libraries can be exported, versioned and archived, thus ensuring that the programmatic environment will be identical across compatible platforms.

Limited backwards compatibility can often hinder the reproducibility of previous scientific findings. For example, newer versions of a statistical analysis tool might not directly support older versions of their proprietary data storage format or an analytical tool might be compiled using a library that is no longer available in newer versions of an operating system. These issues can potentially be mitigated through the use of virtualization [77]. Virtual machines (e.g. Oracle VirtualBox [78], VMware [79]) are essentially containers that can encapsulate a snapshot of the OS, data and analytical pipelines in a single binary file. This can be done irrespective of the “host” operating system that is being used and these binary files are compatible across other operating systems. Other researchers can then use these binary files to replicate the analytical pipeline used for the reported analysis.

### ***Standardized phenotyping algorithms***

No widely accepted approach currently exists for storing the implementation and logic behind EHR phenotyping algorithms in a machine-readable and transportable format. The translation from algorithm logic to programming code is performed manually, and as a result, is prone to errors due to the complexity of the data and potential ambiguity of algorithms [25]. In their work, Mo and colleagues describe the ideal characteristics of such a format such as the ability to support logical and temporal rules, relational algebra operations, integrate with external standardized terminologies and provide a mechanism for backwards-compatibility [80]. The creation and adoption of computational representations of phenotyping algorithms will enable researchers to define and share EHR algorithms defining exposures, covariates and clinical outcomes and share them in a standardized manner. Furthermore, machine-readable representations of EHR phenotyping algorithms will enable their integration with analytical pipelines and will benefit from many of the approaches outlined in this manuscript such as version control, workflow systems and standardized analytical libraries [62, 81]. Finally, algorithm implementations can also be uploaded in open-access repositories [82] or software journals [83] where they could be assigned a unique Digital Object Identifier (DOI) and become citeable and cross-referenced in scientific output.

### **Literate programming**

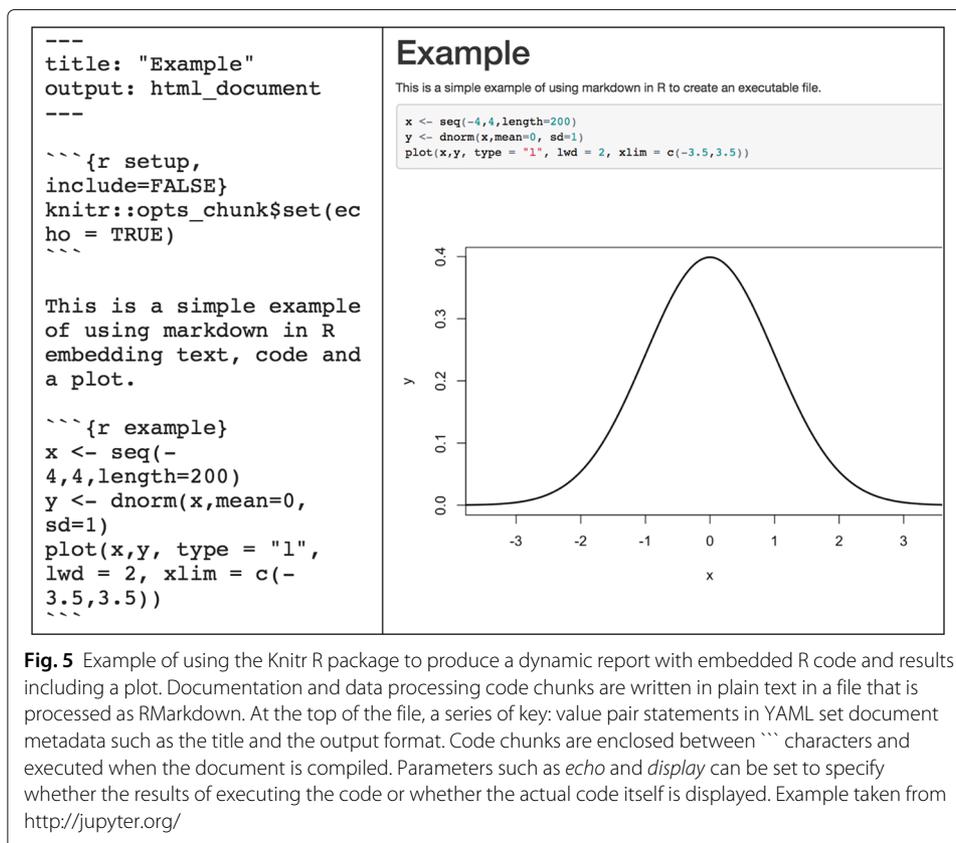
Publishing study data online or in secure repositories alongside the code to preprocess and analyse it may be possible in some biomedical research domains, but is typically not

an option for EHR research given the strict information governance restrictions and legal frameworks researchers operate under. Additionally, EHR sources typically contain the entire patient history and all their interactions with health care settings but only a subset of the original data is used and it is therefore equally important to document and disseminate the process of data extraction as well as the post-processing and analysis.

Extracting the appropriate dataset for research involves specifying lists of relevant controlled clinical terminology terms, timing windows for study population and patient phenotypes, and eligibility criteria. The work of defining the extraction criteria is usually performed by data managers in conjunction with domain experts such as clinicians. The algorithm parameters and implementation details are subsequently converted to machine instructions (e.g. SQL), executed and resulting data are usually exported from a relational database. Although the rationale behind the extraction process and the machine readable code should bear equivalent information content, it is extremely difficult for a human reader to understand the underlying logic and assumptions by reading the code itself [80]. It is also very challenging to fully reproduce the extraction using only the human readable instructions of the agreed protocol given the ambiguity of algorithms and the complexity of the data. Similar challenges exist for the preprocessing of EHR data, such as the definition of new covariates and clinical outcomes, as well as for the analysis and post-processing (such as plotting) of the extracted dataset and results.

A simple and time-honoured solution to this challenge is the provision of documentation alongside the code used for the extraction/preprocessing/analysis of EHR data. This approach however is often problematic as documentation can often be out-of-date with regards to the code, might be incomplete as it is often written after the analysis is finished and for large projects, linking the correct pieces of documentation to the specific locations in the code can be cumbersome. A potential approach to solving this challenge is *literate programming*. The concept of literate programming was introduced by Donald E Knuth [84] and is not limited to a specific analytical tool or programming language. Literate programming is the technique of writing out program logic in a human language with included (separated by a primitive markup) code snippets and macros. In practice, both the rationale behind the data processing pipeline as well as the processing code itself are authored by the user using an appropriate integrated development environment (IDE). The resulting plain text document subsequently undergoes two processes, the first in which the code itself is executed (often referred to as *tangling*) and one in which the formatted documentation is produced (often referred to as *weaving*). The result is a well formatted rich text document, for example Hypertext Markup Language (HTML), which can often include the output of the executed code (e.g. plots, summary tables, analysis results) alongside the original code snippets and documentation (Fig. 5).

The most popular modern day literate programming tool in R is roxygen [85] while popular report generation packages are Knitr [86] and Sweave [87]. Another widely used tool is Jupyter [88], which is often used with (but not limited to) Python. For example, Johnson et al. published all code necessary to reproduce the data description of the MIMIC-III [89] database in the form of Jupyter notebooks on a GitHub repository [90]. The use of Jupyter notebooks was encouraged, and a specially developed software platform that integrated with notebooks was provided for a datathon using the MIMIC-III database, rendering all resulting analysis fully reproducible [91]. Most EHR research



analytical tools support literate programming (Table 6) and its use can greatly facilitate closing the gap between code and narrative.

Taking the literate programming paradigm ever further, *compendia* [92] are containers for distributing and disseminating the different elements that comprise a piece of computational research. These elements are also fundamental in the concept of literate programming, however in the case of *compendia*, the data are also contained in the output [93].

### Conclusion

The challenge of reproducibility in science has been widely recognized and discussed [94]. Scientists using EHR data for biomedical research face a number of significant challenges which are further amplified due to the complexity and heterogeneity of the data sources used and the cross-disciplinary of the field. It is crucial for researchers to

**Table 6** Examples of packages and libraries supporting literate programming and report generation in popular analytical/statistical software packages

Statistical/Analytical tool	Relevant packages
R	RMarkdown, Knitr, Sweave, Roxygen
Stata	MarkDoc, Weaver, Ketchup
Python	Jupyter Notebook, Doxygen
Matlab	Doxygen (limited)
Octave	Doxygen [96]
SAS	SASWeave [97], StatRep

adopt best-practices across disciplines in order to enable the reproducibility of research findings using such data. In this manuscript we identify and present a set of principles, methods and tools from adjunct scientific disciplines that can be utilized to enable reproducible and transparent biomedical research using EHR. Enabling reproducible research using EHR is an ongoing process that will greatly benefit the scientific and wider community.

#### Acknowledgements

We would like to thank Václav Papež for his help in preparing the figures included in this manuscript.

#### Funding

This study was supported by the National Institute for Health Research (RP-PG-0407–10314), Wellcome Trust (086091/Z/08/Z), the Medical Research Council Prognosis Research Strategy Partnership (G0902393/99558) and the Farr Institute of Health Informatics Research, funded by The Medical Research Council (MR/K006584/1), in partnership with Arthritis Research UK, the British Heart Foundation, Cancer Research UK, the Economic and Social Research Council, the Engineering and Physical Sciences Research Council, the National Institute of Health Research, the National Institute for Social Care and Health Research (Welsh Assembly Government), the Chief Scientist Office (Scottish Government Health Directorates) and the Wellcome Trust. LS's contribution was supported by the Wellcome Trust, grant number WT082178. This work was supported by the Francis Crick Institute which receives its core funding from Cancer Research UK (FC001110), the UK Medical Research Council (FC001110), and the Wellcome Trust (FC001110).

#### Availability of data and materials

Final  $\LaTeX$  manuscript and R code examples will be uploaded to GitHub.

#### Authors' contributions

SD and LS designed and supervised the study; SD, AGI, MP, KD drafted the manuscript; All authors critically revised the manuscript and provided final approval; SD is guarantor. All authors read and approved the final manuscript.

#### Ethics approval and consent to participate

Not applicable.

#### Consent for publication

Not applicable.

#### Competing interests

The authors declare that they have no competing interests.

#### Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

#### Author details

<sup>1</sup>Institute of Health Informatics, University College London, 222 Euston Road, NW1 2DA London, UK. <sup>2</sup>Farr Institute of Health Informatics Research, 222 Euston Road, London, UK. <sup>3</sup>The Francis Crick Institute, 1 Midland Road, NW1 1AT London, UK. <sup>4</sup>Institute of Biomedical Informatics, University of Pennsylvania, Richards Medical Research Laboratories, 3700 Hamilton Walk, 19104 Philadelphia, USA. <sup>5</sup>EHR Research Group, Department of Non-communicable Disease Epidemiology, London School of Hygiene and Tropical Medicine, Keppel Street, WC1E 7HT London, UK.

Received: 10 April 2017 Accepted: 28 August 2017

Published online: 11 September 2017

#### References

- Weber GM, Mandl KD, Kohane IS. Finding the missing link for big biomedical data. *JAMA*. 2014;311(24):2479–80.
- Hemingway H, Asselbergs FW, Danesh J, Dobson R, Maniadakis N, Maggioni A, Ghislaine JM, van Thiel MC, Brobert G, Vardas P, Anker SD, Grobbee DE, Denaxas S. On behalf of the Innovative Medicines Initiative 2nd programme, Big Data for Better Outcomes, BigData@Heart Consortium. Big data from electronic health records for early and late translational cardiovascular research: challenges and potential. *Eur Heart J*. 2017;ehx487. <https://doi.org/10.1093/eurheartj/ehx487>.
- Denaxas SC, George J, Herrett E, Shah AD, Kalra D, Hingorani AD, Kivimaki M, Timmis AD, Smeeth L, Hemingway H. Data resource profile: cardiovascular disease research using linked bespoke studies and electronic health records (caliber). *Int J Epidemiol*. 2012;41(6):1625–38.
- Collins FS, Varmus H. A new initiative on precision medicine. *N Engl J Med*. 2015;372(9):793–5.
- Sudlow C, Gallacher J, Allen N, Beral V, Burton P, Danesh J, Downey P, Elliott P, Green J, Landray M, et al. UK biobank: an open access resource for identifying the causes of a wide range of complex diseases of middle and old age. *PLoS Med*. 2015;12(3):1001779.
- Gottesman O, Kuivaniemi H, Tromp G, Faucett WA, Li R, Manolio TA, Sanderson SC, Kannry J, Zinberg R, Basford MA, et al. The electronic medical records and genomics (emerge) network: past, present, and future. *Genet Med*. 2013;15(10):761–71.

7. Crosslin DR, McDavid A, Weston N, Nelson SC, Zheng X, Hart E, De Andrade M, Kullo IJ, McCarty CA, Doheny KF, et al. Genetic variants associated with the white blood cell count in 13,923 subjects in the emerge network. *Hum Genet.* 2012;131(4):639–52.
8. Timmis A, Rapsomaniki E, Chung S, Pujades-Rodriguez M, Moayyeri A, Stogiannis D, Shah A, Pasea L, Denaxas S, Emmas C, et al. Prolonged dual antiplatelet therapy in stable coronary disease: comparative observational study of benefits and harms in unselected versus trial populations. *Bmj.* 2016;353:3163.
9. Shah AD, Langenberg C, Rapsomaniki E, Denaxas S, Pujades-Rodriguez M, Gale CP, Deanfield J, Smeeth L, Timmis A, Hemingway H. Type 2 diabetes and incidence of cardiovascular diseases: a cohort study in 1·9 million people. *Lancet Diabetes Endocrinol.* 2015;3(2):105–13.
10. Rapsomaniki E, Timmis A, George J, Pujades-Rodriguez M, Shah AD, Denaxas S, White IR, Caulfield MJ, Deanfield JE, Smeeth L, et al. Blood pressure and incidence of twelve cardiovascular diseases: lifetime risks, healthy life-years lost, and age-specific associations in 1·25 million people. *Lancet.* 2014;383(9932):1899–911.
11. Rapsomaniki E, Shah A, Perel P, Denaxas S, George J, Nicholas O, Udumyan R, Feder GS, Hingorani AD, Timmis A, et al. Prognostic models for stable coronary artery disease based on electronic health record cohort of 102 023 patients. *Eur Heart J.* 2013;35(13):844–52.
12. Kho AN, Pacheco JA, Peissig PL, Rasmussen L, Newton KM, Weston N, Crane PK, Pathak J, Chute CG, Bielinski SJ, et al. Electronic medical records for genetic research: results of the emerge consortium. *Sci Transl Med.* 2011;3(79):79–1791.
13. Koudstaal S, Pujades-Rodriguez M, Denaxas S, Gho JMIH, Shah AD, Yu N, Patel RS, Gale CP, Hoes AW, Cleland JG, Asselbergs FW, Hemingway H. Prognostic burden of heart failure recorded in primary care, acute hospital admissions, or both: a population-based linked electronic health record cohort study in 2.1 million people. *Eur J Heart Fail.* 2016. doi:10.1002/ejhf.709.
14. Bell S, Daskalopoulou M, Rapsomaniki E, George J, Britton A, Bobak M, Casas JP, Dale CE, Denaxas S, Shah AD, et al. Association between clinically recorded alcohol consumption and initial presentation of 12 cardiovascular diseases: population based cohort study using linked health records. *Bmj.* 2017;356:909.
15. McNutt M. Reproducibility. *Science.* 2014;343(6168):229–9.
16. Begley CG, Ioannidis JP. Reproducibility in science. *Circ Res.* 2015;116(1):116–26.
17. Iqbal SA, Wallach JD, Khoury MJ, Schully SD, Ioannidis JP. Reproducible research practices and transparency across the biomedical literature. *PLoS Biol.* 2016;14(1):1002333.
18. Springate DA, Kontopantelis E, Ashcroft DM, Olier I, Parisi R, Chamapiwa E, Reeves D. Clinicalcodes: an online clinical codes repository to improve the validity and reproducibility of research using electronic medical records. *PLoS ONE.* 2014;9(6):99825.
19. Vezyridis P, Timmons S. Evolution of primary care databases in uk: a scientometric analysis of research output. *BMJ Open.* 2016;6(10):012785.
20. Goodman SN, Fanelli D, Ioannidis JP. What does research reproducibility mean? *Sci Transl Med.* 2016;8(341):341–1234112.
21. Donnelly K. Snomed-ct: The advanced terminology and coding system for ehealth. *Stud Health Technol Inform.* 2006;121:279.
22. Denaxas SC, Asselbergs FW, Moore JH. The tip of the iceberg: challenges of accessing hospital electronic health record data for biological data mining. *BioData Min.* 2016;9(1):29.
23. Denaxas SC, Morley KI. Big biomedical data and cardiovascular disease research: opportunities and challenges. *Eur Heart J-Qual Care Clin Outcome.* 2015;1(1):9–16.
24. Hripcsak G, Albers DJ. Next-generation phenotyping of electronic health records. *J Am Med Inform Assoc.* 2013;20(1):117–21.
25. Newton KM, Peissig PL, Kho AN, Bielinski SJ, Berg RL, Choudhary V, Basford M, Chute CG, Kullo IJ, Li R, et al. Validation of electronic medical record-based phenotyping algorithms: results and lessons learned from the emerge network. *J Am Med Inform Assoc.* 2013;20(e1):147–54.
26. Stein L. Creating a bioinformatics nation. *Nature.* 2002;417(6885):119–20.
27. Benchimol EI, Smeeth L, Guttman A, Harron K, Moher D, Petersen I, Sørensen HT, von Elm E, Langan SM, Committee RW, et al. The reporting of studies conducted using observational routinely-collected health data (record) statement. *PLoS Med.* 2015;12(10):1001885.
28. REporting of Studies Conducted Using Observational Routinely-collected Data (RECORD). <http://www.recordstatement.org/>. Accessed 28 July 2017.
29. Von Elm E, Altman DG, Egger M, Pocock SJ, Gøtzsche PC, Vandenbroucke JP, Initiative S, et al. The strengthening the reporting of observational studies in epidemiology (STROBE) statement: guidelines for reporting observational studies. *Int J Surg.* 2014;12(12):1495–9.
30. STrengthening the Reporting of OBServational Studies in Epidemiology (STROBE). <https://www.strobe-statement.org/>. Accessed 28 July 2017.
31. Davison A. Automated capture of experiment context for easier reproducibility in computational research. *Comput Sci Eng.* 2012;14(4):48–56.
32. Sandve GK, Nekrutenko A, Taylor J, Hovig E. Ten simple rules for reproducible computational research. *PLoS Comput Biol.* 2013;9(10):1003285.
33. Mesirov JP. Accessible reproducible research. *Science.* 2010;327(5964):415–6.
34. Tan TW, Tong JC, Khan AM, de Silva M, Lim KS, Ranganathan S. Advancing standards for bioinformatics activities: persistence, reproducibility, disambiguation and minimum information about a bioinformatics investigation (miabi). *BMC Genom.* 2010;11(4):27.
35. Ravel J, Wommack KE. All hail reproducibility in microbiome research. *Microbiome.* 2014;2(1):8.
36. Peng R. Reproducible research and biostatistics. *Biostatistics.* 2009;10(3):405.
37. Gorgolewski KJ, Alfaro-Almagro F, Auer T, Bellec P, Capotà M, Chakravarty MM, Churchill NW, Cohen AL, Craddock RC, Devenyi GA, et al. Bids apps: Improving ease of use, accessibility, and reproducibility of neuroimaging data analysis methods. *PLoS Comput Biol.* 2017;13(3):1005209.

38. Waltemath D, Adams R, Bergmann FT, Hucka M, Kolpakov F, Miller AK, Moraru II, Nickerson D, Sahle S, Snoep JL, et al. Reproducible computational biology experiments with sed-ml—the simulation experiment description markup language. *BMC Syst Biol*. 2011;5(1):198.
39. The R Project. <https://www.r-project.org/>. Accessed 5 Apr 2017.
40. StataCorp L, et al. Stata data analysis and statistical software. Spec Ed Release. 2007;10:733.
41. The Python Programming Language. <http://www.python.org>. Accessed 5 Apr 2017.
42. The Java Programming Language. <http://www.java.com>. Accessed 5 Apr 2017.
43. Wilson G. Software carpentry: getting scientists to write better code by making them more productive. *Comput Sci Eng*. 2006;8(6):66–9.
44. Parnas DL. On the criteria to be used in decomposing systems into modules. *Commun ACM*. 1972;15(12):1053–8.
45. Stefik M, Bobrow DG. Object-oriented programming: Themes and variations. *AI Mag*. 1985;6(4):40.
46. Meyer B. Applying 'design by contract'. *Computer*. 1992;25(10):40–51.
47. Medvidovic N, Rosenblum DS, Redmiles DF, Robbins JE. Modeling software architectures in the unified modeling language. *ACM Trans Softw Eng Methodol (TOSEM)*. 2002;11(1):2–57.
48. Janzen D, Saiedian H. Test-driven development concepts, taxonomy, and future direction. *Computer*. 2005;38(9):43–50.
49. Fucci D, Turhan B, Juristo N, Dieste O, Tosun-Misirli A, Oivo M. Towards an operationalization of test-driven development skills: An industrial empirical study. *Inf Softw Technol*. 2015;68:82–97.
50. The SAS Analytical Software. <https://www.sas.com/>. Accessed 5 Apr 2017.
51. FUTS SAS Testing Library. <https://info.thotwave.com/access-the-futs-framework-for-unit-testing-sas>. Accessed 5 Apr 2017.
52. SASUnit SAS Testing Library. <https://sourceforge.net/projects/sasunit/>. Accessed 5 Apr 2017.
53. Wickham H. testthat: Get started with testing. *R J*. 2011;3(1):5–10.
54. Burger M, Juenemann K, Koenig T. Runit: r unit test framework. R package version. 2009:0.4. <https://cran.rstudio.com/web/packages/RUnit/>.
55. Grosjean P, Grosjean MP. Package 'svunit'. 2013. <https://cran.r-project.org/web/packages/svUnit/index.html>.
56. Cheon Y, Leavens GT. A simple and practical approach to unit testing: The JML and JUnit way, vol. 2374. In: *ECCOOP*. Springer; 2002. p. 231–55.
57. Beaulieu-Jones BK, Greene CS. Reproducibility of computational workflows is automated using continuous analysis. *Nat Biotechnol*. 2017;35(4):342–346.
58. Clarke EM, Wing JM. Formal methods: State of the art and future directions. *ACM Comput Surv (CSUR)*. 1996;28(4):626–43.
59. Git Version Control System. <https://git-scm.com/>. Accessed 5 Apr 2017.
60. Subversion Version Control System. <http://subversion.apache.org>. Accessed 5 Apr 2017.
61. Pitt-Francis J, Bernabeu MO, Cooper J, Garny A, Momtahan L, Osborne J, Pathmanathan P, Rodriguez B, Whiteley JP, Gavaghan DJ. Chaste: using agile programming techniques to develop computational biology software. *Philos Trans R Soc Lond A: Math, Phys Eng Sci*. 2008;366(1878):3111–36.
62. Papez V, Denaxas S. Evaluation of semantic web technologies for storing computable definitions of electronic health records phenotyping algorithms. *Am Med Informa Assoc Annual Symp*. 2017. <https://arxiv.org/abs/1707.07673>.
63. The CALIBER Data Portal. <https://www.caliberresearch.org/portal/>. Accessed 5 Apr 2017.
64. Pilić A, Procter JB. Ten simple rules for the open development of scientific software. *PLoS Comput Biol*. 2012;8(12):1002802.
65. List M, Ebert P, Albrecht F. Ten simple rules for developing usable software in computational biology. *PLoS Comput Biol*. 2017;13(1):1005265.
66. Cock PJ, Antao T, Chang JT, Chapman BA, Cox CJ, Dalke A, Friedberg I, Hamelryck T, Kauff F, Wilczynski B, et al. Biopython: freely available python tools for computational molecular biology and bioinformatics. *Bioinformatics*. 2009;25(11):1422–3.
67. Gentleman RC, Carey VJ, Bates DM, Bolstad B, Dettling M, Dudoit S, Ellis B, Gautier L, Ge Y, Gentry J, et al. Bioconductor: open software development for computational biology and bioinformatics. *Genome Biol*. 2004;5(10):80.
68. Springate DA, Parisi R, Olier I, Reeves D, Kontopantelis E. rehr: An r package for manipulating and analysing electronic health record data. *PLoS ONE*. 2017;12(2):0171784.
69. Giardine B, Riemer C, Hardison RC, Burhans R, Elnitski L, Shah P, Zhang Y, Blankenberg D, Albert I, Taylor J, et al. Galaxy: a platform for interactive large-scale genome analysis. *Genome Res*. 2005;15(10):1451–5.
70. Wolstencroft K, Haines R, Fellows D, Williams A, Withers D, Owen S, Soiland-Reyes S, Dunlop I, Nenadic A, Fisher P, et al. The taverna workflow suite: designing and executing workflows of web services on the desktop, web or in the cloud. *Nucleic Acids Res*. 2013;41(W1):W557–W561.
71. Köster J, Rahmann S. Snakemake—a scalable bioinformatics workflow engine. *Bioinformatics*. 2012;28(19):2520–2.
72. de la Garza L, Veit J, Szolek A, Röttig M, Aiche S, Gesing S, Reinert K, Kohlbacher O. From the desktop to the grid: scalable bioinformatics via workflow conversion. *BMC Bioinforma*. 2016;17(1):127.
73. Fink J. Docker: a software as a service, operating system-level virtualization framework. *Code4Lib J*. 2014;25:1–3.
74. Conda.io. <https://conda.io/>. Accessed 5 Apr 2017.
75. Guide MU. The mathworks inc. Natick MA. 1998;4:382.
76. The Octave Analytical Software. <https://www.gnu.org/software/octave/>. Accessed 5 Apr 2017.
77. Hurley DG, Budden DM, Crampin EJ. Virtual reference environments: a simple way to make research reproducible. *Brief Bioinform*. 2015;16(5):901–3.
78. Oracle VirtualBox Virtualization Software. <https://www.virtualbox.org>. Accessed 5 Apr 2017.
79. VMware Virtualization Software. <http://www.vmware.com/>. Accessed 5 Apr 2017.
80. Mo H, Thompson WK, Rasmussen LV, Pacheco JA, Jiang G, Kiefer R, Zhu Q, Xu J, Montague E, Carrell DS, et al. Desiderata for computable representations of electronic health records-driven phenotype algorithms. *J Am Med Inform Assoc*. 2015;22(6):1220–30.

81. Pathak J, Kiefer RC, Bielinski SJ, Chute CG. Applying semantic web technologies for phenome-wide scan using an electronic health record linked Biobank. *J Biomed Semant.* 2012;3(1):10. doi:10.1186/2041-1480-3-10. <https://doi.org/10.1186/2041-1480-3-10>.
82. Figshare. <https://figshare.com/>. Accessed 5 Apr 2017.
83. The Journal of Open Source Software. <http://joss.theoj.org/>. Accessed 5 Apr 2017.
84. Knuth DE. Literate programming. *The Computer Journal.* 1984;27(2):97–111.
85. Roxygen Package. <https://cran.r-project.org/web/packages/roxygen2/index.html>. Accessed 5 July 2017.
86. Xie Y. *Dynamic Documents with R and Knitr*, vol 29. Florida: CRC Press; 2015.
87. Leisch F. Sweave: Dynamic generation of statistical reports using literate data analysis. In: *Compstat.* Springer; 2002. p. 575–80. <http://www.springer.com/us/book/9783790815177>.
88. Ragan-Kelley M, Perez F, Granger B, Kluyver T, Ivanov P, Frederic J, Bussonier M. The jupyter/ipython architecture: a unified view of computational research, from interactive exploration to communication and publication. In: *AGU Fall Meeting Abstracts.* 2014. p. 07. <http://adsabs.harvard.edu/abs/2014AGUFM.H44D.07R>.
89. MIMIC-III Source Code Repository. <https://github.com/MIT-LCP/mimic-code>. Accessed 5 Apr 2017.
90. Johnson AEW, Pollard TJ, Shen L, Lehman L-wH, Feng M, Ghassemi M, Moody B, Szolovits P, Celi LA, Mark RG. MIMIC-III, a freely accessible critical care database. *Sci Data.* 2016;3: Nature Publishing Group.
91. Aboab J, Celi LA, Charlton P, Feng M, Ghassemi M, Marshall DC, Mayaud L, Naumann T, McCague N, Paik KE, et al. A “datathon” model to support cross-disciplinary collaboration. *Sci Transl Med.* 2016;8(333):333–83338.
92. Gentleman R, Temple Lang D. Statistical analyses and reproducible research. *J Comput Graph Stat.* 2007;16(1):1–23.
93. Peng RD, Dominici F, Pastor-Barriuso R, Zeger SL, Samet JM. Seasonal analyses of air pollution and mortality in 100 us cities. *Am J Epidemiol.* 2005;161(6):585–94.
94. Ioannidis JP. Why most published research findings are false. *PLoS med.* 2005;2(8):124.
95. Version Control. [https://en.wikipedia.org/wiki/Version\\_Control](https://en.wikipedia.org/wiki/Version_Control). Accessed 28 July 2017.
96. Doxygen. <http://doxygen.org/>. Accessed 5 Apr 2017.
97. Lenth RV, Højsgaard S, et al. Sasweave: Literate programming using sas. *J Stat Softw.* 2007;19(8):1–20.

Submit your next manuscript to BioMed Central  
and we will help you at every step:

- We accept pre-submission inquiries
- Our selector tool helps you to find the most relevant journal
- We provide round the clock customer support
- Convenient online submission
- Thorough peer review
- Inclusion in PubMed and all major indexing services
- Maximum visibility for your research

Submit your manuscript at  
[www.biomedcentral.com/submit](http://www.biomedcentral.com/submit)

