Article published online: 2023-01-25

“fhircrackr”: An R Package Unlocking Fast Healthcare Interoperability Resources for Statistical Analysis

Julia Palm[1] Frank A. Meineke[2] Jens Przybilla[2,3] Thomas Peschel[2]

[1] Institute of Medical Statistics, Computer and Data Sciences, Jena University Hospital, Jena, Thüringen, Germany
[2] Institute for Medical Informatics, Statistics and Epidemiology, University of Leipzig, Leipzig, Germany
[3] Clinical Trial Centre Leipzig, University of Leipzig, Leipzig, Germany

Address for correspondence Julia Palm, MSc, Institute of Medical Statistics, Computer and Data Sciences, Jena University Hospital, Bachstraße 18, 07743 Jena, Germany (e-mail: julia.palm@med.uni-jena.de).

Appl Clin Inform 2023;14:54–64.

Abstract

Background The growing interest in the secondary use of electronic health record (EHR) data has increased the number of new data integration and data sharing infrastructures. The present work has been developed in the context of the German Medical Informatics Initiative, where 29 university hospitals agreed to the usage of the Health Level Seven Fast Healthcare Interoperability Resources (FHIR) standard for their newly established data integration centers. This standard is optimized to describe and exchange medical data but less suitable for standard statistical analysis which mostly requires tabular data formats.
Objectives The objective of this work is to establish a tool that makes FHIR data accessible for standard statistical analysis by providing means to retrieve and transform data from a FHIR server. The tool should be implemented in a programming environment known to most data analysts and offer functions with variable degrees of flexibility and automation catering to users with different levels of FHIR expertise.
Methods We propose the fhircrackr framework, which allows downloading and flattening FHIR resources for data analysis. The framework supports different download and authentication protocols and gives the user full control over the data that is extracted from the FHIR resources and transformed into tables. We implemented it using the programming language R [1] and published it under the GPL-3 open source license.
Results The framework was successfully applied to both publicly available test data and real-world data from several ongoing studies. While the processing of larger real-world data sets puts a considerable burden on computation time and memory consumption, those challenges can be attenuated with a number of suitable measures like parallelization and temporary storage mechanisms.
Conclusion The fhircrackr R package provides an open source solution within an environment that is familiar to most data scientists and helps overcome the practical challenges that still hamper the usage of EHR data for research.

Keywords
- Fast Healthcare Interoperability Resources
- electronic health records
- health information interoperability
- data analysis

received
August 31, 2022
accepted after revision
November 9, 2022

DOI https://doi.org/10.1055/s-0042-1760436.
ISSN 1869-0327.

Background and Significance

Background

The digitalization of the health sector produces a growing interest in the use of electronic health record (EHR) data for clinical research. With the rise of the coronavirus disease 2019 pandemic, the need to share large-scale, near real-time patient data has shifted into focus[1–3] and there has been increasing activity in building new infrastructures[4,5] for data integration and data sharing.

In the context of this growing activity, the Health Level Seven (HL7) Fast Healthcare Interoperability Resources (FHIR) standard[6] has developed into one of the most popular interoperability standards worldwide. From the creation of FHIR profiles[7] to the mapping[8] and usage of the FHIR standard for data sharing, data visualization, and data quality assessment,[9–13] a vast number of projects using FHIR continues to grow all over the world.

Following this trend, the German Medical Informatics Initiative (MII)[14] established 29 data integration centers in university hospitals across four consortia to integrate EHR data using FHIR. The FHIR standard focuses on the exchange of health care data and defines around 160 so-called resource types, each of which stands for a distinct entity of information that can be transferred using standard web technologies.[15] While FHIR is well-established as a standard in clinical data exchange, for example, for medical applications, Duda et al[16] have found that it is less commonly used in clinical research. Since FHIR is optimized to describe and transfer medical data of individual patients, it is less tailored to statistical analysis of patient cohorts and other applications requiring tabular data. FHIR resources are represented as interlinked JSON or XML objects instead of the matrix-like structures and cannot be handled by most statistical software. The proposed framework is based on the XML representation of FHIR, which is also used throughout this article.

We developed a framework for extracting and reshaping data from FHIR resources using the well-established XPath expression language,[17] which allows analysts to create tables from FHIR data that can be processed by any standard statistical or other software that expects tabular data. The framework is implemented in the open-source R package[18] fhircrackr[19] and can be downloaded from the comprehensive R archive network.[20]

In the remainder of this article, we will introduce the basics of the FHIR standard followed by a conceptual description of how the fhircrackr framework builds on this standard. Then, we will illustrate the framework with a reproducible example using open access test data and conclude with a discussion of the framework in light of currently ongoing studies, which use real-world EHR data for statistical analyses in several areas of biomedical research.

FHIR Basics

The FHIR standard is used to describe data originating from clinical routine documentation. In this standard, all data elements are represented as so-called FHIR resources. A single FHIR resource holds a specific piece of information like a laboratory measurement, demographic information on a patient, or the description of a medical procedure. A full list of currently defined FHIR resources can be found at: *https://www.hl7.org/fhir/resourcelist.html*. Two simplified examples are shown in ►**Fig. 1**, where information on a patient and a measurement of her body temperature are represented in the form of a Patient resource and an Observation resource, respectively.

Ideally, every FHIR resource holds the machine-readable information that is needed to interpret the data including its provenance, the actual data point (e.g., the body temperature measured), the metadata (e.g., the unit and date), and references to other resources the data relates to (e.g., the patient the temperature is taken from). The nested structure of the

```
<Patient>
    <id value="111"/>
    <name>
        <use value="official"/>
        <family value="Everywoman"/>
        <given value="Eve"/>
    </name>
    <telecom>
        <system value="phone"/>
        <value value="555-555-2003"/>
        <use value="work"/>
    </telecom>
    <gender value="female"/>
    <birthDate value="1973-05-31"/>
    <address>
        <use value="home"/>
        <line value="2222 Home Street"/>
    </address>
</Patient>
```

```
<Observation>
    <id value="222"/>
    <code>
        <coding>
            <system value="http://loinc.org"/>
            <code value="8310-5"/>
            <display value="Body temperature"/>
        </coding>
        <test value="Body temperature"/>
    </code>
    <subject>
        <reference value="Patient/111"/>
    </subject>
    <effectiveDateTime value="2022-05-07"/>
    <valueQuantity>
        <value value="36.5"/>
        <unit value="C"/>
        <system value="http://unitsofmeasure.org"/>
        <code value="Cel"/>
    </valueQuantity>
</Observation>
```

Fig. 1 Simplified Patient and Observation resources.

resource is needed because EHR data, like other kinds of data that is exchanged in discrete units, has an inherently nested structure. In the FHIR R4, the most recent version of the standard, there are around 160 resource types, each of them devoted to a specific kind of medical information. In theory, each FHIR resource is individually retrievable from the server, but for statistical analysis, they are mostly downloaded in bundles. A FHIR bundle is a collection of multiple resources that can be exchanged and processed as a single entity. The FHIR standard is a RESTful specification, which uses the Hypertext Transfer Protocol (HTTP) for communication. To download data from the server, the standard describes the so-called FHIR Search paradigm, in which the user specifies the scope of the downloaded data with suitable search parameters and downloads the respective resources using HTTP GET or POST. The details of this paradigm are beyond the scope of this article, however, extensive documentation can be found at: *https://www.hl7.org/fhir/search.html*.

The structure of the FHIR standard stands in contrast to data formats that are usually employed in data analysis. Most statistical software expects a matrix-like structure, where columns represent features and rows represent observations on the set of features formed by those columns. A single body temperature measurement, for example, could be one cell in a matrix that has patients in rows and measurements of vital signs in columns. In contrast, the FHIR representation of this single cell would be an entire json/xml object that contains not only the measured temperature value but also references to the patient it is taken from and other metadata. Information that is contained in a single table in a classical data analysis setting is expressed in a large number of interlinked resources when the data is represented in FHIR. Developing a framework bridging the gap between those two representations is the primary goal of this work.

Objectives

The objective of this work is to establish a tool that makes FHIR data accessible for standard statistical analyses by providing means to retrieve and transform data from a FHIR server. The tool should have the following characteristics:

- Build on the communication interface defined by HL7 FHIR.
- Support common authentication protocols used for sensitive data.
- Be implemented in a programming environment known to most data analysts.
- Offer functions with variable degrees of flexibility and automation catering to users with different levels of FHIR expertise.

Methods

Two steps are necessary to make FHIR data available for analysis: (1) download the data from the server and (2) transform the data into a table format. To perform those tasks, the fhircrackr framework defines a set of classes and related functions, an overview over which can be found in ►Fig. 2. Below, we will describe the necessary steps sequentially.

Downloading FHIR Data

To access FHIR data, the fhircrackr framework builds on the FHIR Search paradigm defined by HL7. The framework

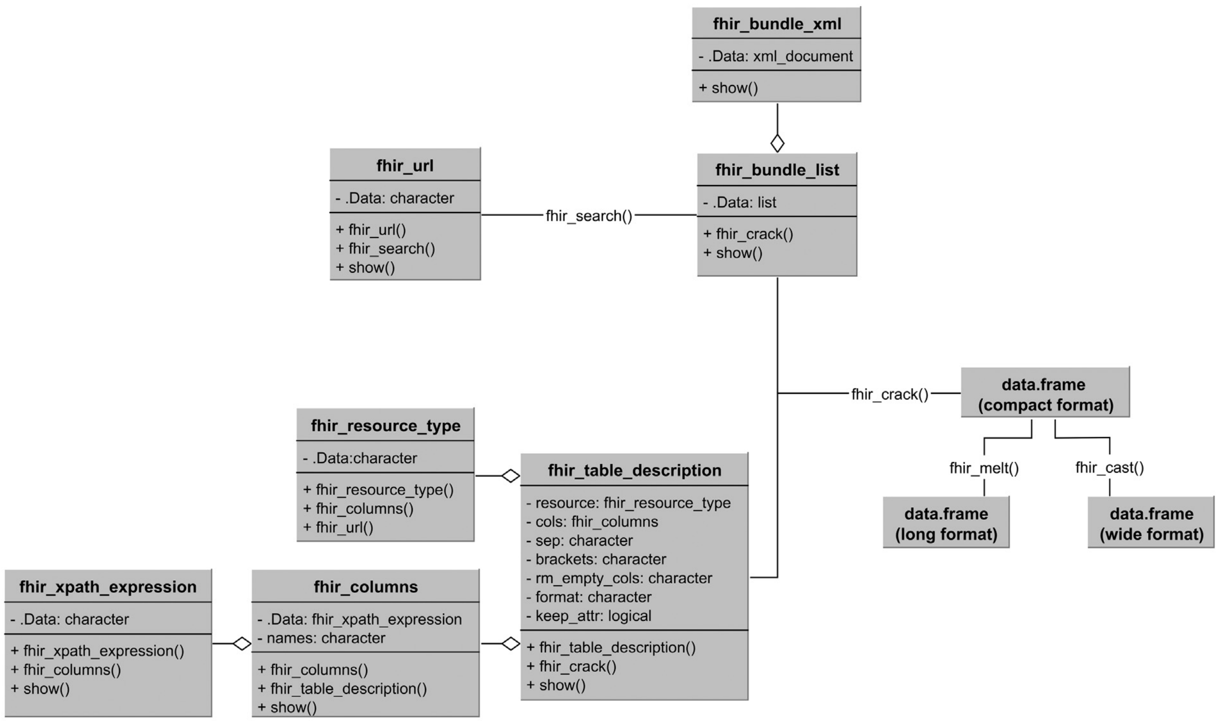


Fig. 2 Overview over the classes and functions of the fhircrackr framework as described in this article. For each class, the name, attributes with data type and most important methods are displayed from top to bottom.

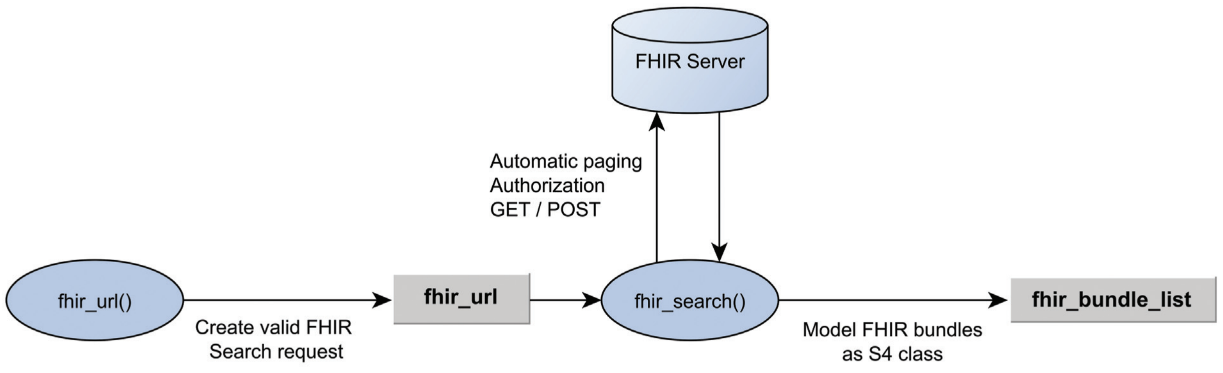


Fig. 3 Workflow of the downloading process: *fhir_url()* creates an object of class *fhir_url*, which is sent to the server by *fhir_search()* and the result is a *fhir_bundle_list* object.

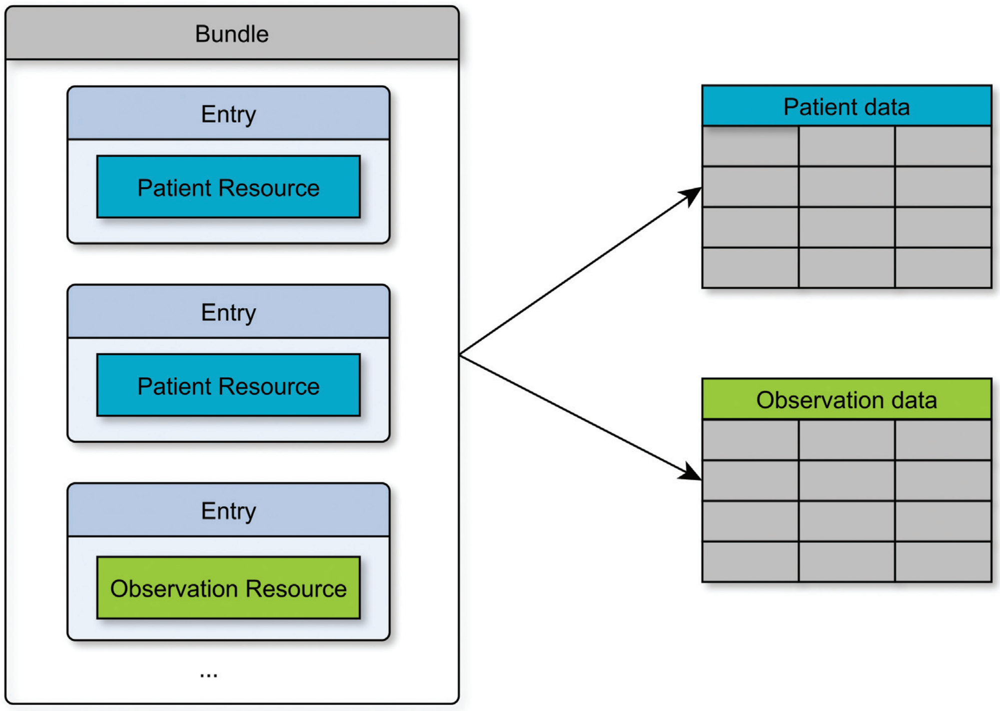


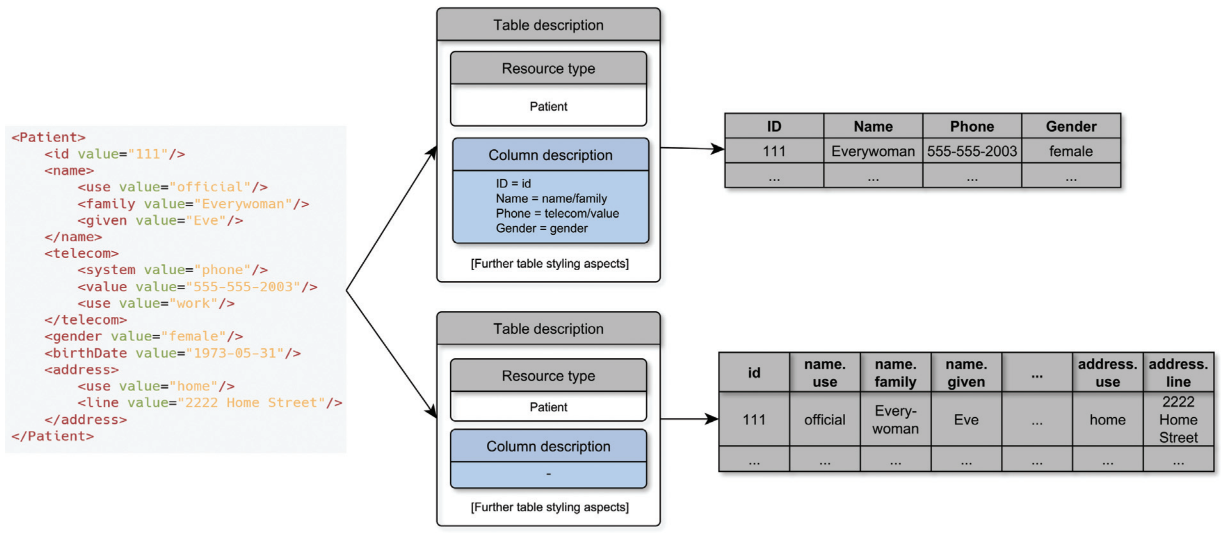
Fig. 4 Each resource type results in one table.

provides functionalities for the construction of FHIR search expressions via the function *fhir_url()* and uses the function *fhir_search()* to download the bundles. *fhir_search()* allows for basic authentication and Open Authorization (OAuth 2.0[21]) as well as automatic paging (sequential download of search result bundles) when the result is distributed over several bundles. Downloads can be done via GET or POST, depending on the use case, and can easily be parallelized when necessary. See ►**Fig. 3** for a representation of the downloading workflow. The downloading functions build on well-established functions from the httr-Package,[22] but extend them by providing mechanisms for automatic paging as well as helper functions for building valid FHIR search requests and handling of common errors. The result of the downloading process is a list of FHIR bundles, which is represented as an S4 class[23] called *fhir_bundle_list*. Within this list, a single bundle is an object of class *fhir_bundle_xml*, an S4 class extending *xml_document* from the xml2 R package.[24] The FHIR bundles can be transformed into tables (referred to as "flattening") as described in the next section.

Flattening FHIR Data

Resource Types

Depending on the type of search request, a FHIR bundle can contain more than one resource type. When the resources are flattened, the fhircrackr framework constructs one table for each resource type, as shown in ►**Fig. 4**. This is necessary because the variability between resources of different types is larger than the variability between resources of the same type. Putting the data of all resource types in a single table would thus create unreasonably large and sparsely populated tables.



```
<Patient>
    <id value="111"/>
    <name>
        <use value="official"/>
        <family value="Everywoman"/>
        <given value="Eve"/>
    </name>
    <telecom>
        <system value="phone"/>
        <value value="555-555-2003"/>
        <use value="work"/>
    </telecom>
    <gender value="female"/>
    <birthDate value="1973-05-31"/>
    <address>
        <use value="home"/>
        <line value="2222 Home Street"/>
    </address>
</Patient>
```

ID	Name	Phone	Gender
111	Everywoman	555-555-2003	female
...	...	...	...

id	name.use	name.family	name.given	...	address.use	address.line
111	official	Every-woman	Eve	...	home	2222 Home Street
...	...	...	...	...	...	...

Fig. 5 Two modes of flattening. The upper row defines selected columns in the column description element; the lower row extracts all available elements as columns.

Elements to Extract

There are two modes of flattening the resources, depicted in ►Fig. 5. Either (1) the entire structure is flattened, that is, every element that occurs in at least one of the resources will be turned into a column of the resulting table, or (2) a predefined subset of the elements is extracted from the resources. The former approach requires little knowledge about the individual structure of the resources but can result in very wide and sparse tables. The latter approach, in contrast, requires some knowledge about the structure of the resources, but will result in more concise and densely populated tables. ►Fig. 6 provides an overview over the workflow of the flattening process: All information needed to flatten the resources is represented in the class *fhir_table_description*, which contains the type of resource to target (an object of class *fhir_resource_type*) and a list of XPath 1.0[17] expressions (an object of class *fhir_columns*) specifying which elements of the resource to extract into the columns of the resulting table. The function *fhir_crack()* runs the extraction on all resources in the bundles, with one row being created per resource. If a specified element is not present in one of the resources, the corresponding cell in the resulting table is left empty.

The generic extraction process based on a user-specified XPath scheme ensures that the framework does not depend on any predefined resource structure, apart from it being valid xml. This has the advantage of making the tool compatible with both standard FHIR resources as well as arbitrary extensions to those resources and makes sure that the tool works on all versions of the FHIR standard, like the past STU3, current R4, and future R5 version.

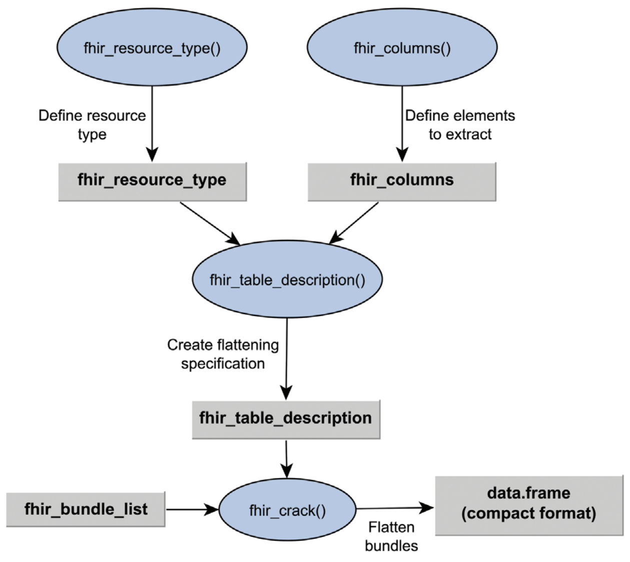
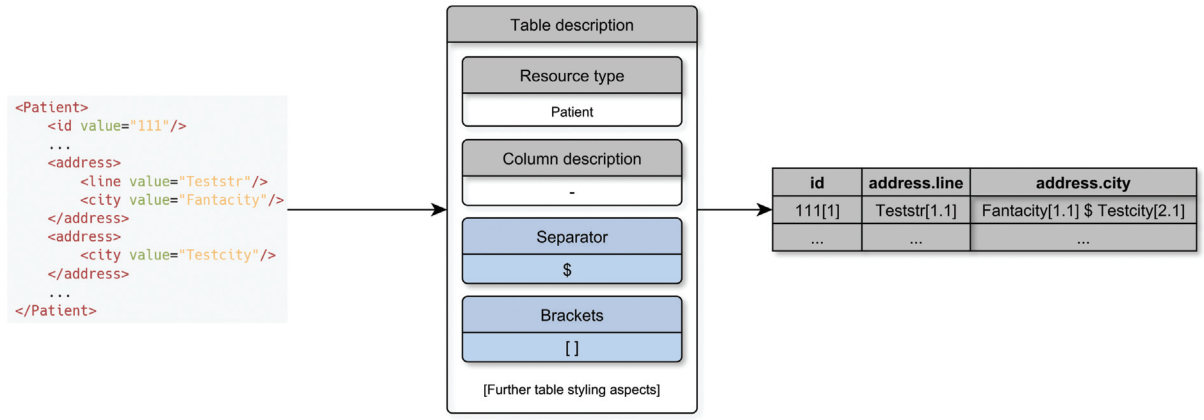


Fig. 6 Workflow of the flattening process. Information on the resource type and columns to extract is combined in the *fhir_table_description* object, which is used by *fhir_crack()* to flatten the *fhir_bundle_list* object into a *data.frame*.

Repeating Elements

To represent complex EHR data adequately, FHIR resources can contain repeating elements, that is, elements with a cardinality greater than one. The FHIR core specification of the Patient resource,[25] for example, defines a 0* cardinality for the element "Patient.address," meaning that a single Patient resource can contain zero, one, or multiple address elements. Creating a new column for each instance of the element would greatly inflate the resulting table, as a single resource with 10 address elements would result in 10 address columns. This effect worsens as the number of elements with multiple instances increases. To counteract this issue, the fhircrackr default puts values of repeating elements in the same column, separated by a string that can be defined in the table description, as depicted in ►Fig. 7. To be able to tell apart which value belongs to which parent element, the fhircrackr can assign an index to each entry, which is surrounded by prespecified brackets. Together with the XPath or header of the column, the indices allow a reconstruction of the original FHIR structure, including the precise position of every element. We are aware that this



```
<Patient>
    <id value="111"/>
    ...
    <address>
        <line value="Teststr"/>
        <city value="Fantacity"/>
    </address>
    <address>
        <city value="Testcity"/>
    </address>
    ...
</Patient>
```

id	address.line	address.city
111[1]	Teststr[1.1]	Fantacity[1.1] $ Testcity[2.1]
...	...	...

Fig. 7 Repeating elements are put in the same column and separated as specified by the table description.

default behavior violates the first normal form of database normalization,[26] which states that there should be no columns with sets of values. In consultation with various statisticians, we have however adopted the presented solution, since searching such complex fields in the cases considered here is easier to handle in common statistical packages than distributing the values over several columns or tables.

To resolve columns with sets of values, the fhircrackr framework provides functionalities to automatically split those values, either into a wide format with multiple columns (called casting), or into a long format with multiple rows (called melting). ►Fig. 8 illustrates the two approaches: In a cast table, the names of each column reflect the original position of the element and each row represents one resource. In a molten table, the number of columns stays the same and multiple entries are spread over several rows. As opposed to casting, melting is done with respect to a specific element (the address element in this example) and values from all other elements (e.g., the id element) are copied into the newly created rows. ►Fig. 9 represents the workflow of casting or melting a table with multiple entries.

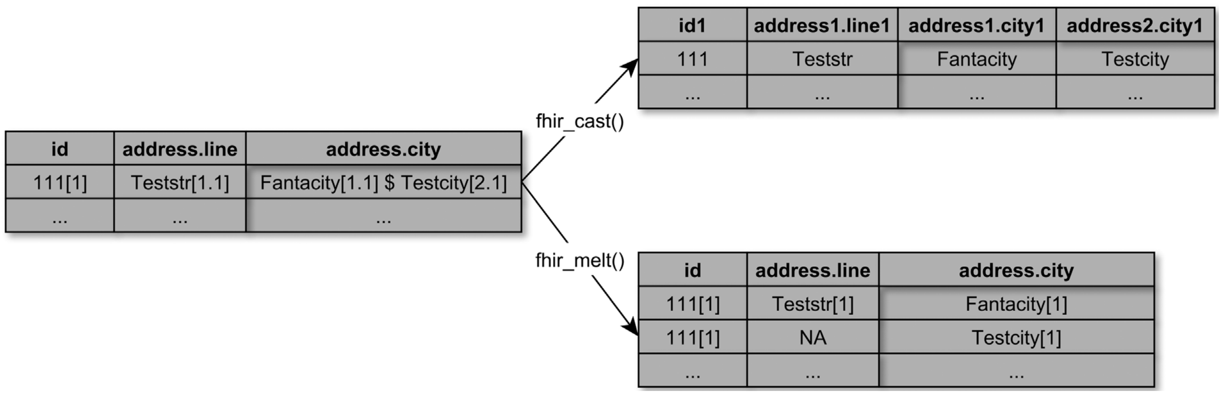
The steps described so far only aim at making the data available in tabular form. In addition, common techniques of data preprocessing, exploration, and visualization should precede any statistical analysis, which however is beyond the scope of this article. A good summary of those methods specifically for EHR data is given by Yu et al.[27]

Recreating Resources

Apart from turning FHIR resources into flat tables, the fhircrackr framework also allows data to flow in the opposite direction, creating FHIR resources from flat tables. Because the framework is able to remember the original structure of the resources through column headers and indexing of entries, it is possible to recreate the original resources from the flattened tables. This allows the user to selectively remove or add single elements as well as running more complex computations to augment the original FHIR resources before sending them back to a server.

R Package fhircrackr

We implemented the above functionalities in a package called fhircrackr using the programming language R[1] and published it under the GPL-3 open source license on the Comprehensive R Archive Network (see "Software Availability" below). The package requires R Version 4.1.0 or higher and can be run on any operating system that can run R. While the package does not have specific hardware requirements and can be operated on a standard office notebook, the actual



id	address.line	address.city
111[1]	Teststr[1.1]	Fantacity[1.1] $ Testcity[2.1]
...	...	...

id1	address1.line1	address1.city1	address2.city1
111	Teststr	Fantacity	Testcity
...	...	...	...

id	address.line	address.city
111[1]	Teststr[1]	Fantacity[1]
111[1]	NA	Testcity[1]
...	...	...

Fig. 8 Different ways of expanding multiple entries. *fhir_cast()* distributes multiple values of the address.city element over multiple columns, *fhir_melt()* distributes them over multiple rows.

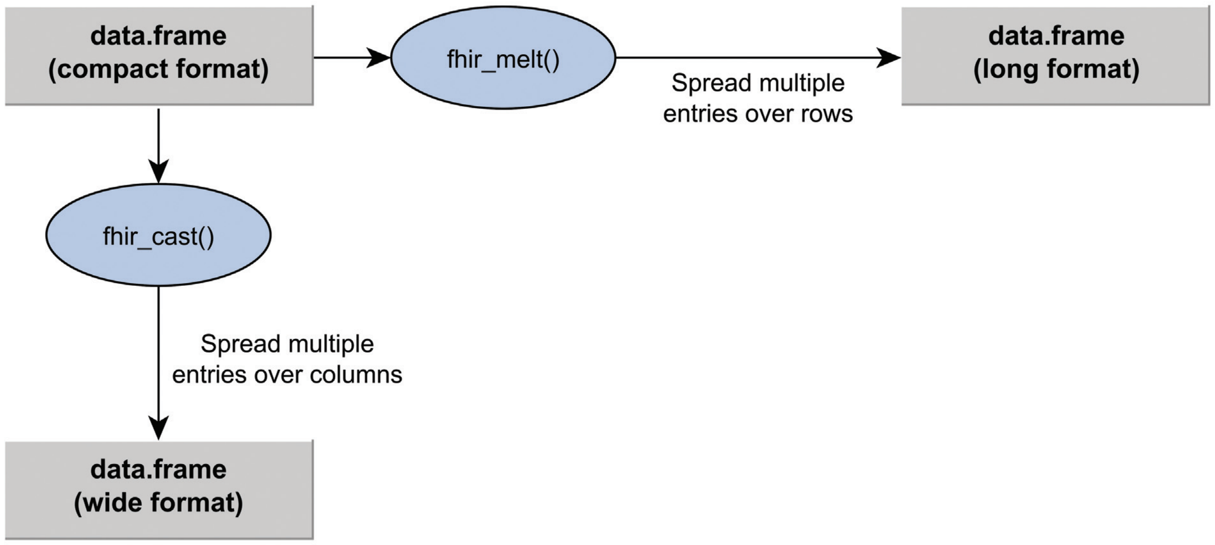

data.frame (compact format)
fhir_melt()
Spread multiple entries over rows
data.frame (long format)
fhir_cast()
Spread multiple entries over columns
data.frame (wide format)

Fig. 9 Workflow of expanding multiple entries. Starting from a compact *data.frame*, *fhir_melt()* and *fhir_cast()* expand this *data.frame* into long and wide format, respectively.

requirements regarding RAM and CPU strongly depend on the use case and the scope of the data being processed.

Results

Application to Test Data

To illustrate the basic functionalities of the package, we have set up a simple project on publicly available test data[28] at: *https://github.com/POLAR-fhiR/fhircrackr_examples*. In this project, we investigate the relationship between body mass index (BMI) and hypertensive disease (ICD-10 Code I10 - I15) diagnosed as a comorbidity in a small patient cohort. To achieve this, we need to (1) compute the BMI for each patient from downloaded Observation resources containing body weight and body height measurements and (2) relate it to the diagnosis information given in the Encounter and Condition resources of those patients. First, the code example shows how the function *fhir_url()* is used to build the appropriate FHIR search request which is then sent to the server using *fhir_search()*. The resulting list of FHIR bundles contains Patient and Observation resources. To flatten those resources into tables, two table descriptions are defined as shown in ►**Fig. 10**.

The function *fhir_crack()* is then used to create the two tables and after some data preprocessing, the BMI is computed and a list of patient IDs is used to download the associated Encounter and Condition resources. According to the MII implementation guide,[29] which profiles an Encounter in the German health care system that is used in the test data, the diagnoses of a Patient are referenced in the Encounter resource and there can be multiple diagnoses listed in one encounter, as shown in ►**Fig. 11**. The table created from those Encounters will therefore contain multiple entries in the diagnosis columns, as shown in the top row of ►**Fig. 12**. Those multiple diagnoses have to be spread over multiple rows using *fhir_melt()*, as shown in the middle row of ►**Fig. 12** and subsequently be freed from indices with *fhir_rm_indices()* as shown in the bottom row of the figure. Then, they can be filtered for comorbidity diagnoses. Finally, the patients with a hypertension comorbidity diagnosis can be identified and the relationship between hypertension and BMI can be examined with suitable statistical routines.

Application to Real-World Data

To date, fhircrackr is used in a number of ongoing studies within the German MII. In the use case POLAR[30] it is used for a distributed analysis in 13 university hospitals to detect health risks in patients with polypharmacy. The clinical trial HELP[31] uses the package to evaluate a decision support system for bloodstream infections in five university hospitals. Several smaller studies (e.g., refs. 32 and 33) showcase the usage of the fhircrackr in publicly available GitHub repositories.[34,35]

In all these applications, the analysis is distributed in the form of plain or dockered R scripts via version control platforms such as GitHub, GitLab, or DockerHub. The scripts generate tabular data on different levels of abstraction from simple feature extraction to more complex local aggregation for distributed learning approaches. All these approaches are easy to implement once the FHIR data has been made available as tables in R.

Discussion

Usage of Real-World Data

The use of nested information from real-world EHR data in data science applications has demonstrated difficulties, and some of them can be solved by fhircrackr.

The first challenge when downloading and processing FHIR data is performance, regarding both memory and computation time. Most FHIR server implementations are optimized to handle many small, parallel requests referring to a single patient or encounter, but are slow or even return incomplete results when processing requests for bigger, cross-case/cross-patient chunks of data. A comprehensive comparison of downloading times for different FHIR server

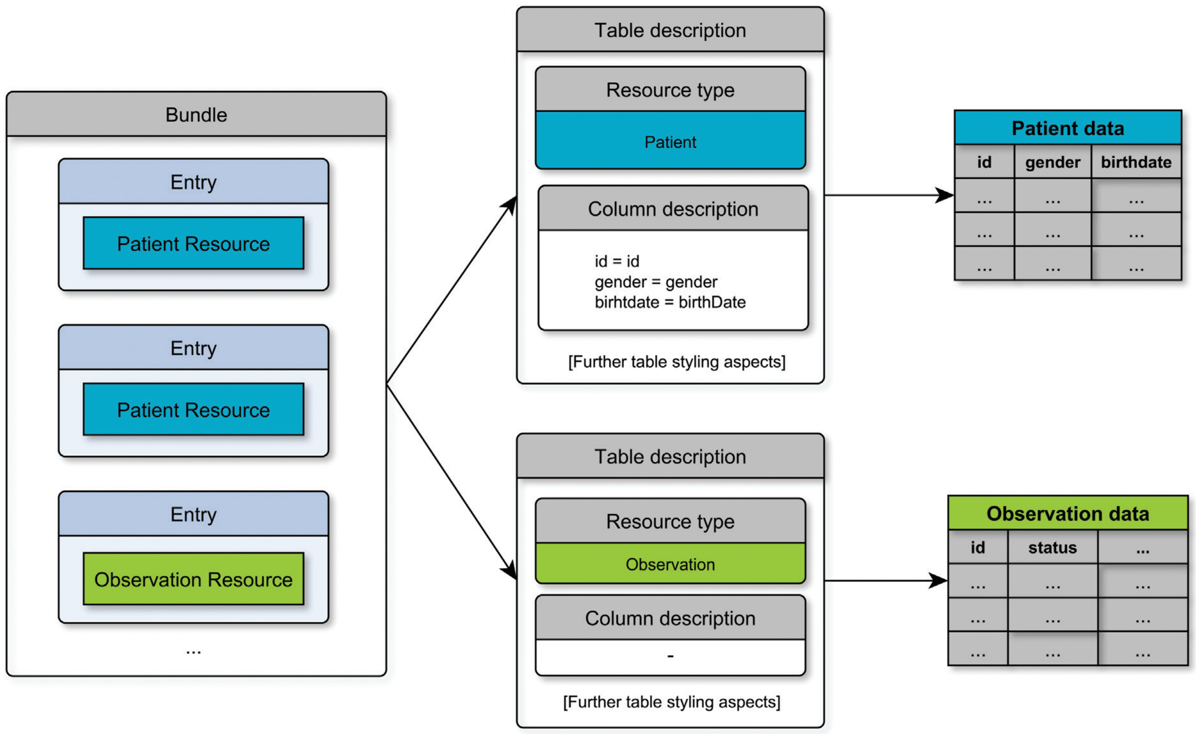


Fig. 10 Table descriptions for Patient and Observation resources.

```
<Encounter>
    <id value="UKB001-E-1"/>
    ...
    <diagnosis>
        <condition>
            <reference value="Condition/UKB001-CD-2"/>
        </condition>
        <use>
            <coding>
                <system value="http://terminology.hl7.org/CodeSystem/diagnosis-role"/>
                <code value="CC"/>
                <display value="Chief complaint"/>
            </coding>
        </use>
    </diagnosis>
    <diagnosis>
        <condition>
            <reference value="Condition/UKB001-CD-3"/>
        </condition>
        <use>
            <coding>
                <system value="http://terminology.hl7.org/CodeSystem/diagnosis-role"/>
                <code value="CM"/>
                <display value="Comorbidity diagnosis"/>
            </coding>
        </use>
    </diagnosis>
    ...
</Encounter>
```

Fig. 11 Excerpt from an Encounter resource that has two entries for the Encounter.diagnosis element, one for the chief complaint and one for a comorbidity diagnosis.

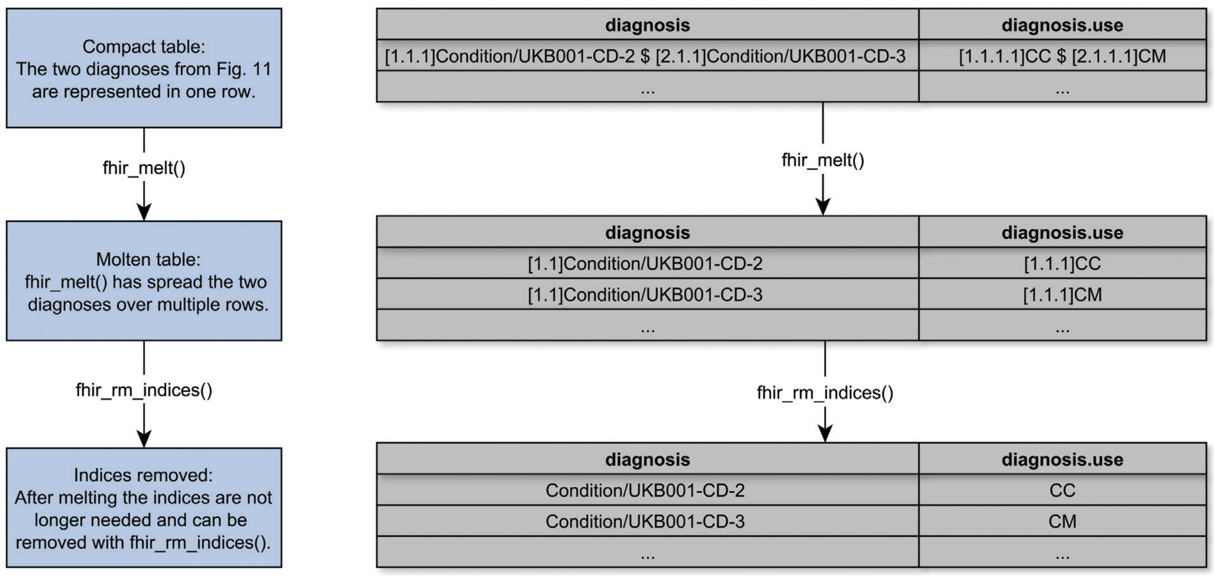

Compact table:
The two diagnoses from Fig. 11 are represented in one row.
fhir_melt()
Molten table:
fhir_melt() has spread the two diagnoses over multiple rows.
fhir_rm_indices()
Indices removed:
After melting the indices are not longer needed and can be removed with fhir_rm_indices().
diagnosis
diagnosis.use
[1.1.1]Condition/UKB001-CD-2 $ [2.1.1]Condition/UKB001-CD-3
[1.1.1.1]CC $ [2.1.1.1]CM
[1.1]Condition/UKB001-CD-2
[1.1.1]CC
[1.1]Condition/UKB001-CD-3
[1.1.1]CM
Condition/UKB001-CD-2
CC
Condition/UKB001-CD-3
CM
...

Fig. 12 Excerpt from the Encounters table. In the first step fhir_melt() spreads the two diagnosis entries over two rows, in the second step fhir_rm_indices() removes the bracketed indices.

implementations is still needed though some aspects have been discussed by Gruendner et al.[36] When using fhircrackr, the downloading time can be improved by (1) splitting requests in smaller chunks and sending them in parallel using the R package "parallel" and (2) by carefully choosing the most restrictive filter criterion in the first FHIR search request, when several FHIR search queries are combined. In terms of memory, the XML structure of the FHIR resources produces a lot of overhead, challenging memory capacity in large data applications. The use of JSON instead of XML objects might bring some merit in this case, but has not been adopted, as there is no counterpart of XPath expressions available in R so far. The same is true for FHIRPath, which would be another alternative to XPath expressions. To reduce memory consumption, fhircrackr allows for a variety of options to process FHIR data batch wise, caching the interim results, and offers built-in parallelization to speed up the flattening process.

The second challenge lies in data privacy protection. The fhircrackr package has been developed within the German MII, where analysis of EHR data is subject to strict rules and protection. Distributed analyses as implemented in the DataSHIELD framework[37] offer a privacy-preserving solution, but those frameworks can only process tabular data. In this case, fhircrackr could serve as a particularly useful connection piece, unlocking the use of DataSHIELD for FHIR data, since similarly to DataSHIELD, fhircrackr is written and executed in R.

The third challenge in using EHR data represented by FHIR resources is data quality. A successful secondary usage of EHR data hinges on a reliable quality assessment and careful metadata interpretation. Proprietary data formats employed by commercial clinical information systems often do not even supply important metadata information. Conversely, the FHIR standard not only provides this metadata, but also does so in a machine-readable format. A careless application of the fhircrackr framework could lose this important metadata when it is not selected for extraction, which at its core is a danger not unique to using fhircrackr but common to all analyses of EHR data. It is therefore vital that users carefully ascertain which data elements are necessary for adequate evaluation of data quality. Then, one solution for quality assessment is the harmonized quality assessment framework proposed by Kahn et al,[38] which has been adapted and implemented by Kapsner et al[39] in the R package DQAstats. As most data quality assessment tools do, this framework builds on tabular data. Transforming FHIR data into R tables thus allows for the use of established frameworks and technical solutions for data quality assessment. Kohane et al[40] emphasize how important a careful and critical appraisal of the opportunities and limitations of EHR-derived clinical studies is and describe the deficits this young field still shows.

Comparison with Existing Tools

Apart from the fhircrackr, a number of other technologies for tabulating FHIR data already exist. One of them is Bunsen, developed by Cerner[41] which allows users to analyze FHIR data with Apache Spark. While Bunsen is valuable for users already familiar with the cluster computing and large-scale data processing tools provided by the Spark framework, fhircrackr caters to a different user group rooted in more classical, biostatistical areas of research. An integration into the well-established parallelization processes of Spark gives Bunsen an advantage regarding efficiency. On the other hand, the Bunsen framework explicitly builds on certain releases of the FHIR standard (STU3/R4) and is thus less flexible than the much more generic fhircrackr framework.

A recently developed tool that is more similar to fhircrackr is the Python-based FHIR-PYrate.[42] This tool provides similar functionalities as fhircrackr, that is, functions to download resources from a server and flatten them according to a user-specified scheme. A notable difference is the usage of

FHIRPath in FHIR-PYrate versus XPath in fhircrackr. Due to the limited time both packages have been available, there are no systematic performance comparisons yet. However, we believe that a sound analysis of EHR data requires the expertise and collaboration of experts from a variety of disciplines, such as clinicians, biomedical informaticians, and statisticians. Fostering the collaboration and exchange of knowledge between those fields of knowledge must therefore be of the highest priority for future usage of EHR data in clinical studies. To achieve this, we need a toolkit containing suitable tools for each of those diverse user groups. The development of tools for analysis of FHIR resources in different environments and for different frameworks is therefore essential and the tools mentioned above should be seen as complementing each other.

Conclusion

The increasing application of the HL7 FHIR standard calls for accessible tools that allow applied researchers to analyze EHR data with all its opportunities and limitations. To our knowledge, there are no established solutions for accessing and transforming the complex FHIR data in R to make it available for statistical analysis, reporting, or quality management.

The fhircrackr R package provides an open source solution within the R environment, which is familiar to most data scientists, and helps overcome the practical challenges that hamper sound statistical analyses of EHR data to date. First applications of the tool in medical research show that the analysis of EHR data requires special attention in a number of areas. From the technical side, the considerable amount of data to be processed calls for both intelligent design of analysis plans as well as adequate hardware to deal with these amounts of data. From the content side, EHR data is not collected for scientific purposes and thus should only be used with special attention to data quality and metadata interpretation. Learning from these lessons, we intend to improve the efficiency of the fhircrackr implementation further by carefully assessing and fixing possible inefficacies in the code. Furthermore, we will advocate collaboration of experts from the diverse fields that are needed for a successful usage of EHR data by disseminating our work in talks and workshops to help bridging the gap between the world of clinical documentation and biomedical research.

Software Availability Statement

The R package *fhircrackr* is available at: *https://github.com/POLAR-fhiR/fhircrackr* and *https://cran.r-project.org/package=fhircrackr* and is released under the GPL-3 open source license.

Simple examples of its usage can be found at https://github.com/POLAR-fhiR/fhircrackr_examples and in the vignettes of the package.

Clinical Relevance Statement

The fhircrackr framework provides an open source, easy to use tool to process FHIR resources for the secondary use of EHR data. Unlocking this data for medical data scientists and statisticians paves the way for innovative research improving patient relevant outcomes as well as health care systems on a broader level.

Multiple-Choice Questions

1. How can the user define which FHIR resource elements to extract with the fhircrackr?
   a. With JSONPath expressions
   b. With XPath expressions
   c. With FHIRPath expressions
   d. With plain text descriptions of the element

   Correct Answer: The correct answer is option b. The user can define elements to extract using XPath expressions, because the resources are represented as xml resources in the fhircrackr framework.

2. How does the fhircrackr handle repeating elements in a FHIR resource?
   a. Only the first element is extracted into one cell.
   b. Only the last element is extracted into one cell.
   c. All elements are extracted into one cell.
   d. The corresponding cell is left empty.

   Correct Answer: The correct answer is option c. Per default the fhircrackr extracts all repeating elements into the same cell of the resulting table, separated by a prespecified string.

Protection of Human and Animal Subjects

Artificial EHR data were used for developing and testing this software. No formal intervention was performed and no additional (patient-) data were collected.

Funding

This work was funded in part by the German Federal Ministry of Education and Research (BMBF) within the Medical Informatics Initiative in the SMITH Consortium and the POLAR project under the funding numbers 01ZZ1910[A-Z] (POLAR) and 01ZZ1803[A-Z] (SMITH).

Conflict of Interest

None declared.

References

1 Cosgriff CV, Ebner DK, Celi LA. Data sharing in the era of COVID-19. Lancet Digit Health 2020;2(05):e224
2 Dagliati A, Malovini A, Tibollo V, Bellazzi R. Health informatics and EHR to support clinical research in the COVID-19 pandemic: an overview. Brief Bioinform 2021;22(02):812–822
3 Khan MS, Dar O, Erondu NA, et al. Using critical information to strengthen pandemic preparedness: the role of national public health agencies. BMJ Glob Health 2020;5(09):e002830
4 Prokosch HU, Bahls T, Bialke M, et al. The COVID-19 Data Exchange Platform of the German University Medicine. Stud Health Technol Inform 2022;294:674–678
5 Brat GA, Weber GM, Gehlenborg N, et al. International electronic health record-derived COVID-19 clinical course profiles: the 4CE consortium. NPJ Digit Med 2020;3:109

6 Lehne M, Luijten S, Vom Felde Genannt Imbusch P, Thun S. The use of FHIR in digital health - a review of the scientific literature. Stud Health Technol Inform 2019;267:52–58
7 Matney SA, Heale B, Hasley S, et al. Lessons learned in creating interoperable Fast Healthcare Interoperability Resources profiles for large-scale public health programs. Appl Clin Inform 2019;10 (01):87–95
8 Kiourtis A, Mavrogiorgou A, Menychtas A, Maglogiannis I, Kyriazis D. Structurally mapping healthcare data to HL7 FHIR through ontology alignment. J Med Syst 2019;43(03):62
9 Garcia SJ, Zayas-Cabán T, Freimuth RR. Sync for genes: making clinical genomics available for precision medicine at the point-of-care. Appl Clin Inform 2020;11(02):295–302
10 Gordon WJ, Baronas J, Lane WJ. A FHIR human leukocyte antigen (HLA) interface for platelet transfusion support. Appl Clin Inform 2017;8(02):603–611
11 McClure RC, Macumber CL, Skapik JL, Smith AM. Igniting harmonized digital clinical quality measurement through terminology, CQL, and FHIR. Appl Clin Inform 2020;11(01):23–33
12 Dorr DA, D'Autremont C, Pizzimenti C, et al. Assessing data adequacy for high blood pressure clinical decision support: a quantitative analysis. Appl Clin Inform 2021;12(04):710–720
13 Mavrogiorgou A, Kiourtis A, Touloupou M, et al. Internet of medical things (IoMT): acquiring and transforming data into HL7 FHIR through 5G network slicing. Emerg Sci J 2019;3:64–77
14 Semler SC, Wissing F, Heyder R. German Medical Informatics Initiative. Methods Inf Med 2018;57(S 01):e50–e56
15 Bender D, Sartipi K. HL7 FHIR: An Agile and RESTful approach to healthcare information exchange. In: Proceedings of the 26th IEEE International Symposium on Computer-Based Medical Systems. IEEE; 2013:326–331
16 Duda SN, Kennedy N, Conway D, et al. HL7 FHIR-based tools and initiatives to support clinical research: a scoping review. J Am Med Inform Assoc 2022;29(09):1642–1653
17 World Wide Web Consortium. XML Path Language (XPath) 1.0. W3C Recommendation. 2016. Accessed August 28, 2022, at: https://www.w3.org/TR/1999/REC-xpath-19991116
18 R Core Team. R: A Language and Environment for Statistical Computing. Vienna, Austria: R Foundation for Statistical Computing; 2021
19 Peschel T, Palm J, Przybilla J, Meineke F. Handling HL7 FHIR resources in R with fhircrackr. R package version 2.0.1; 2022
20 The Comprehensive R Archive Network. Accessed August 29, 2022 at: https://cran.r-project.org/
21 Hardt D RFC6749 - The OAuth 2.0 Authorization Framework. Internet Engeneering Task Force. 2012. Accessed August 29, 2022 at: https://www.rfc-editor.org/rfc/rfc6749
22 Wickham H httr: Tools for Working with URLs and HTTP. R package version 1.4.4. 2022. Accessed October 27, 2022 at: https://CRAN.R-project.org/package=httr
23 Wickham H S4. In: Advanced R. 2nd ed. London, United Kingdom: Chapman & Hall/CRC; 2019
24 Wickham H, Hester J, Ooms J xml2: Parse XML. R package version 1.3.3. 2021. Accessed August 29, 2022 at: https://CRAN.R-project.org/package=xml2
25 HL7 FHIR Resource Patient Accessed August 29, 2022 at: https://www.hl7.org/fhir/patient.html
26 Codd EF. A relational model of data for large shared data banks. Commun ACM 1970;13:377–387
27 Yu D, Wang X, Wu H EHR Data Pre-Processing and Preparation. In: Statistics and Machine Learning Methods for EHR Data. 1st ed. New York: Chapman and Hall/CRC; 2020:37
28 Leipzig Health Atlas FHIR Testdata. Accessed August 29, 2022 at: https://www.health-atlas.de/studies/43
29 Zautke A, Essenwanger A, Wulff A, et al. Kerndatensatz Modul Fall. Accessed August 29, 2022 at: https://www.medizininformatik-initiative.de/Kerndatensatz/Modul_Fall/IGMIIKDSModulFall.html
30 Scherag A, Andrikyan W, Dreischulte T, et al. POLAR–„POLypharmazie, Arzneimittelwechselwirkungen und Risiken "–wie können Daten aus der stationären Krankenversorgung zur Beurteilung beitragen? Prävent Gesundhförd 2022:1–10
31 Hagel S, Gantner J, Spreckelsen C, et al. Hospital-wide ELectronic medical record evaluated computerised decision support system to improve outcomes of Patients with staphylococcal bloodstream infection (HELP): study protocol for a multicentre stepped-wedge cluster randomised trial. BMJ Open 2020;10 (02):e033391
32 WEather-based STroke event and Outcome Risk Modeling (WE-STORM) project description. Accessed August 29, 2022 at: https://forschen-fuer-gesundheit.de/project.php?fdpgid=20
33 NT-proBNP als Marker bei Vorhofflimmern (MII-VHF) project description. Accessed August 29, 2022 at: https://forschen-fuer-gesundheit.de/project.php?fdpgid=19
34 Palm J Code for MII Projectathon #6: MII-VHF. Accessed August 29, 2022 at: https://github.com/medizininformatik-initiative/Projectathon6-smith2
35 Santhanam N, Maros M Code for MII Projectathon #6: WE-STORM. Accessed August 29, 2022 at: https://github.com/medizininformatik-initiative/Projectathon6-miracum1
36 Gruendner J, Deppenwiese N, Folz M, et al. The architecture of a Feasibility Query Portal for Distributed COVID-19 Fast Healthcare Interoperability Resources (FHIR) patient data repositories: design and implementation study. JMIR Med Inform 2022;10(05): e36709
37 Marcon Y, Bishop T, Avraam D, et al. Orchestrating privacy-protected big data analyses of data from different resources with R and DataSHIELD. PLOS Comput Biol 2021;17(03): e1008880
38 Kahn MG, Callahan TJ, Barnard J, et al. A harmonized data quality assessment terminology and framework for the secondary use of electronic health record data. EGEMS (Wash DC) 2016;4(01):1244
39 Kapsner LA, Mang JM, Mate S, et al. Linking a consortium-wide data quality assessment tool with the MIRACUM metadata repository. Appl Clin Inform 2021;12(04):826–835
40 Kohane IS, Aronow BJ, Avillach P, et al; Consortium For Clinical Characterization Of COVID-19 By EHR (4CE) What every reader should know about studies using electronic health record data but may be afraid to ask. J Med Internet Res 2021;23(03):e22219
41 Cerner. Bunsen: FHIR Data with Apache Spark. Accessed October 27, 2022 at: https://engineering.cerner.com/bunsen/0.5.10-SNAPSHOT/
42 Hosch R, Baldini G, Parmar V, et al. UMEssen/FHIR-PYrate: v0.2.0-beta.1 (v0.2.0-beta.1). Zenodo; 2022. Accessed December 27, 2022 at: https://doi.org/10.5281/zenodo.7025227