frontiers | Frontiers in Medicine

TYPE Technology and Code
PUBLISHED 18 December 2024
DOI 10.3389/fmed.2024.1430676

OPEN ACCESS

EDITED BY
Gokce Banu Laleci Erturkmen,
Software Research and Development Consulting, Türkiye

REVIEWED BY
Andreas K. Triantafyllidis,
Centre for Research and Technology Hellas (CERTH), Greece
Ozgur Kilic,
Muğla University, Türkiye

*CORRESPONDENCE
Philipp Antczak
✉ philipp.antczak@uk-koeln.de

RECEIVED 10 May 2024
ACCEPTED 21 October 2024
PUBLISHED 18 December 2024

CITATION
Schmidt J, Arjune S, Boehm V, Grundmann F, Müller R-U and Antczak P (2024) Bridging health registry data acquisition and real-time data analytics.
Front. Med. 11:1430676.
doi: 10.3389/fmed.2024.1430676

Bridging health registry data acquisition and real-time data analytics

Johannes Schmidt[1], Sita Arjune[2,3,4], Volker Boehm[5], Franziska Grundmann[2], Roman-Ulrich Müller[2,3,4] and Philipp Antczak[2,4]*

[1]Bonacci GmbH, Cologne, Germany, [2]Department II of Internal Medicine and Center for Molecular Medicine Cologne, University of Cologne, Faculty of Medicine and University Hospital Cologne, Cologne, Germany, [3]Center for Rare Diseases Cologne, Faculty of Medicine and University Hospital Cologne, University of Cologne, Cologne, Germany, [4]Cologne Excellence Cluster on Cellular Stress Responses in Aging-Associated Diseases (CECAD), Cologne, Germany, [5]Institute for Genetics, University of Cologne, Cologne, Germany

The number of clinical studies and associated research has increased significantly in the last few years. Particularly in rare diseases, an increased effort has been made to integrate, analyse, and develop new knowledge to improve patient stratification and wellbeing. Clinical databases, including digital medical records, hold significant amount of information that can help understand the impact and progression of diseases. Combining and integrating this data however, has provided a challenge for data scientists due to the complex structures of digital medical records and the lack of site wide standardization of data entry. To address these challenges we present a python backed tool, Meda, which aims to collect data from different sources and combines these in a unified database structure for near real-time monitoring of clinical data. Together with an R shiny interface we can provide a near complete platform for real-time analysis and visualization.

KEYWORDS

visualisation, shiny, database, healthcare, cohort

Introduction

The medical world has seen a paradigm shift in recent years, acknowledging that data collection and analysis is key to understand the most pressing challenges in human health. Particularly with rare diseases, where the low number of patients impact the statistical analysis of these, must ensure that high quality and systematic collection of data is optimized (1). Often retrospectively collected data is available within the hospitals medical record systems but are plagued by numerous free-text fields, simple collection of laboratory values where the measurement units are not standardized across the fields, and the sheer amount of variables that that have been accumulated into these databases over years of use (2, 3). Transitioning such database entities to more standardized and usable structures for clinical research or even simple oversight of departments within a healthcare organization can prove to be challenging and associated with a very high cost of implementation and transition.

Furthermore, quality control of such data is often performed only when data is extracted for clinical research and entry failures only noticed when compared to other individuals. This proves one of the major headaches for data scientists who aim to integrate and analyse such data in various contexts (4). Within the medical field, and especially for laboratory values, thresholds are known that describe compatibility with life, giving a first indication whether the values entered are reasonable. Given the broad spectrum of diseases and health states in humans it is not reasonable to assume that each medical professional knows and applies these thresholds, particularly when

they are early in their career. Written laboratory reports often include the range and thresholds to consider, but once provided within the database these are lost or stored in such a way that they are not directly accessible by the user (5). A more direct, and disease tailored, approach on the level of medical record oversight and data entry could lead to improved data quality and medical understanding.

In the last decade in Germany, there has been significant progress in the development of standardized interfaces to allow interoperability of data between health care institutions. The FHIR interface aims to provide a solution to transport data from one location to another and allow the sharing of patient data. While these developments are of great importance in the medical field, they do not fully address the internal and integrative use in clinical research. To this end, we have developed a small highly flexible and dynamic tool to collate, aggregate, integrate, and visualize clinical data. We opted to develop a centralized database structure, that pulls in data from multiple sources, formats, aligns, and tests them to ensure highest possible data quality. This database can then be connected to a visualization framework such as R shiny, Grafana, or Tableau to present the data in an aggregated fashion to healthcare professionals.

Implementation

Development of a universal translation service for medical data (MEDA)

Clinical registries are often based on data registration, management and storage designs which lack up-to-date database standards. These range from mere spreadsheets to specialized but non-standardized databases from various providers to collect and represent data (6). While these web-based tools often contain the ability to validate data entry, or limit the entry to specific datatypes, these features are often not used due to their complex configuration or lack of knowledge and experience by the initiating user. In addition, database structures are often inefficiently designed and variable names lack the descriptive nomenclature which allows other users to understand their values and implement these variables in their analyses. This then often requires the development of additional variable-dictionaries which provide extended definitions of the values. Particularly for the key aim of such datasets, i.e., downstream biostatistical analyses or real-time visualization, the initial data structure and simplicity of the database is an important aspect for implementation and use. Live data visualization for both data sharing and in-house observation of cohort development is hardly possible in this setting. This is especially important when the developer of the database has left the organization and the approaches and thoughts during the development process have not been documented accordingly. In most cases, the initial design allows questions posed by the developer and researchers associated with the project to be answered, however they can hinder the further use and analyses of these important data.

To address the challenges around clinical datasets described above, and to enable the utilization of existing resources, we have developed a Python and PostgreSQL application that is able to translate the existing information into a standardized database with a very well-defined data structure.[1] Specifically, we inherit the individual centric view fundamental to medical science and attach additional information as separated tables that can be brought together to analyse various questions. These tables separate cross-sectional and longitudinal data and are grouped based on their clinical relevance. The typical database structure we have utilized is provided in Figure 1 and highlights the components that are required to be configured within our tool.

To test this simple structure, we used a large patient cohort with chronic kidney disease available at the University Hospital Cologne and translated the currently utilized ClinicalSurveys.net (7) database using our tool. ClinicalSurveys is a web-based tool to design and collect patient relevant data through a simple survey based tool. It allows collaboration across multiple sites in a secure manner and enables a systematic data collection. This cohort data contains numerous, meticulously collected patient information ranging from different levels of laboratory values, questionnaires, family history, tomography, or historic clinical information. All in all, over 2000 variables were represented within this ClinicalSurveys database structure. The design of this database followed a fully patient centric approach where longitudinal data was encoded as repeated variables within its single database. While this can be a reasonable approach to collect prospective data on individuals over a longer period of time, it can be quite error prone as, particularly, longitudinal data may be entered in the wrong section of the database skewing downstream analysis. In addition, the long list of variables can lead to an increase in human errors during entry where misplaced punctuation marks or swapping of variables may occur. Downstream analyses and visualization may then be skewed by these data structures. Furthermore, quality assurance is more difficult to achieve since the large number of variables within a single database is challenging to evaluate for human individuals. Meda addresses some of these challenges in a semi-automated fashion. Most importantly, the tool automatically generates the database structure based on the configured data slots required. In essence, Meda follows a simple 5 steps approach:

Step 1: Reading Source Data:

> The pipeline begins by reading raw source data in a flat structure, where each value occupies its own column.

Step 2: Data Class Organization:

> The flat data is organized into nested data classes, which correspond to SQL-tables. When defining the data classes. At this point transformations or other computed variables can be generated through the provision of additional python functions.

Step 3: Data Class Factory:

> The data class factory populates the nested data classes from the flat data structure.

Step 4: DTO (Data-Transfer-Object) Factory:

> The DTO factory translates the nested data classes into DTOs that mirror the SQL structure.

Step 5: DTO Registry:

> The DTO registry manages the DTO factory and database connection. It generates a DTO from a data class and writes it to the database.

1 https://github.com/bonacci-johannes/meda

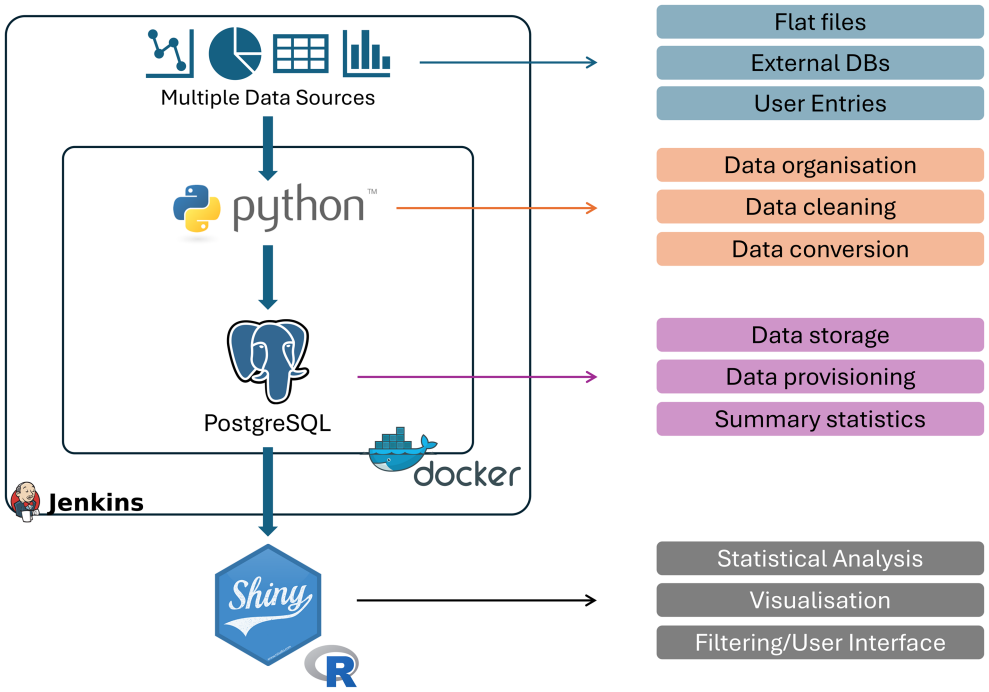

FIGURE 1
Schematic overview of Meda. The developed pipeline and containers used to transform clinical data to a real-time visualisation platform as well as their specific functions within the pipeline. Python and PostgreSQL are encased within a docker to ensure robust responses and compatibility as well as a Jenkins Butler to update, maintain and reset the database when necessary.

In addition, Meda aims to generate subsets of manageable chunks of data, following clinically relevant chunks of information. Import classes are defined that ensure errors can be caught and that data is imported in the right format. In prospective studies, where data is continuously collected over long periods of time, we are therefore able to import data on a regular basis. Next, based on the result of the class import an additional error table is generated that allows users to visualize these import errors and address them accordingly. We found that the visual representation had a significant influence on the motivation of our staff to fix and remedy the shown errors. Lastly, we implemented a threshold-based value verification system which aims to identify values that we deem to be "not compatible with life" and which are sent back to the users for verification.

Application

Setting up Meda for semi-automated data entry

As in our example, ClinicalSurveys was used to house and collect the data from various collaborators, we used Dockers surrounding our database and Meda tool to simplify the setting up and destruction of the database. Simply put, the PostgreSQL database is fully refreshed upon each update that is being made. This ensured that only one true data source was available and reduced the need for verification of data entries within our database. To manage the automated setting up and destruction of the database Jenkins was used. The Jenkins Butler (8) monitors changes in the source code, scripts, and classes that are required and updates the database as soon as changes are observed. The total workflow using this approach takes less than 3 min and can therefore be performed as often as daily if new data are expected on a regular basis. The aforementioned classes need to be implemented to ensure that the right data is entered into the database. A simple Patient centric import of individual characteristics is shown in Code Section 1. The utilized Feature keyword here is an included separate class which provides the information on how to construct a dataclass from a data dictionary and how to import it into the SQL table. It enables the use of transformers, specifications of the input key, specifications of target table type (error, crossectional, or longitudinal), and the potential defaults to consider.

```
class HeadADPKD(FeatureDataclass):
    patient_id: str = Feature(input_key='u_name', unique_index=True)

class Patient(HeadADPKD):
    # Columns
    clinical_survey_id: str = Feature(input_key='uid')
    clinical_survey_user: str = Feature(input_key='u_firstname')
    birthdate: Optional[date] = Feature(input_key='u_birth', null_defaults=NullDefaults.date)
    gender: str = Feature(input_key='u_gender')
    age_at_adpkd_diagnosis: Optional[int] = Feature(input_key='v_3726',
        null_defaults=NullDefaults.text, comment='years')

    # Sub tables
    race: Optional[Race] = Feature(input_key=('v_3381', 'v_3431'),
        transformer=transformer_race)
    health_status: Optional[HealthStatus]
    mutation: Optional[Mutation]
    meta: Optional[Meta]
    examinations: FrozenSet[MedicalExamination]
    family: Optional[Family]
    tolvaptan_dosing: Optional[TolvaptanDosing]
    tolvaptan_reaction: Optional[TolvaptanReaction]
    updosing: FrozenSet[Updosing]

    # errors
    import_errors: Optional[str] = Feature(is_error_field=True)
```

Code Section 1: Example Code for extracting data into the proposed database schema.

Automated evaluation and identification of missing and non-reliable data

During the data import, several additional steps are performed before adding the data to the database. First values are converted to a common reference unit. The unit conversion is a simple step but requires extensive configuration that covers all possible units. So far we have focused on the possible units within our CKD use case example and provide our conversion tables within the code. Code section 2 shows an example of such a configuration. This ensures that we do not need to store the unit information and that all data are converted to the relevant reference unit. Next, data are reviewed for known thresholds that are not compatible with life. Here a simple table (Table 1), which can be adjusted by user dynamically through a web-based interface, is evaluated and any values exceeding these thresholds collected within their own separate table. The results are presented to the user who can then adjust, if necessary, the value within the original table used as input. This also applies to any missing data encountered during the data import.

This workflow can easily be integrated into daily clinical routines and allows for direct evaluation and visualization of the data. In addition, the near instant visual response to the fixing of missing or non-reliable data results in a significantly increased data quality. Furthermore, enabling auditing within PostgreSQL can provide a continuous log of changes that have been performed and ensure that data consistency is preserved.

```
##############################
'density':
  ref_unit: 'g/L'
  conversion:
    'mg/L': 1000
    'mg/dL': 100
##############################
'particle_density':
  ref_unit: 'x 1E9/l'
  conversion:
    '/µl': 1000
    '/µL': 1000
    'x 1E3/µl': 1
    'x 1E12/l': 0.001
    'x 1E6/µl': 0.001
##############################
'molar_density':
  ref_unit: 'mol/L'
  conversion:
    'mol/l': 1
    'mmol/L': 1.e+3
    'µmol/L': 1.e+6
    'pmol/L': 1.e+12
    'pmol/l': 1.e+12
```

Code Section 2: Automated conversion of units during import and plausibility check.

Visualization and continuous evaluation of data provides new insights into patient health

The last step in our pipeline is the development of a visual representation of the data imported by Meda. Here we decided to develop an R (9) shiny application. While other types of frameworks exist to provide real-time views of such data, they are limited in their

TABLE 1 Example threshold table used during data import.

Column	Review_high	Invalid_low	Invalid_high
Natrium	160	115	160
Kalium	7	2	7
Lipase	1000	0	3000
Osmolarity	330	240	350
Hematocrit	50	20	70
Mcv	105	50	120
Calcium	3	1	4
Phosphat	2.5	0.2	6
Creatinine	3	0.2	20
Urea	200	10	500
Uric_acid	12	0.2	25
Albumin	60	5	100

statistical analyses that can provide useful information in a clinical setting (Figure 2).

Integration and functionality of shiny application in R

The development of the Shiny application (10) represents a significant progression in the implementation of the ClinicalSurveys database. The Shiny framework in R facilitates the creation of dynamic web applications that offer the ability to visualize and analyze data in real time. Through the utilization of this technology, the application converts unprocessed clinical data into user-friendly, interactive dashboards and reports. As a result, in the future healthcare providers are provided with instantaneous access to patient information and trends. The Shiny application has been carefully designed to accommodate the particular requirements of healthcare professionals. The platform provides a collection of interactive tools that enable users to analyze demographic patient data along multiple axes, including time, disease advancement, and treatment results. At this degree of engagement, a more profound comprehension of patient health patterns is fostered, which empowers the development of individualized patient care plans and the discovery of effective treatment protocols.

Continuous evaluation for proactive healthcare

One of the most significant features of the Shiny application is its capability for continuous data evaluation. As the PostGreSQL database is refreshed with each update (once daily), the application automatically incorporates the latest data, ensuring that healthcare providers have access to the most current patient information. The insights garnered from the continuous evaluation of patient data have profound implications for both patient care and clinical research. For patient care, it enables a shift toward more proactive and personalized

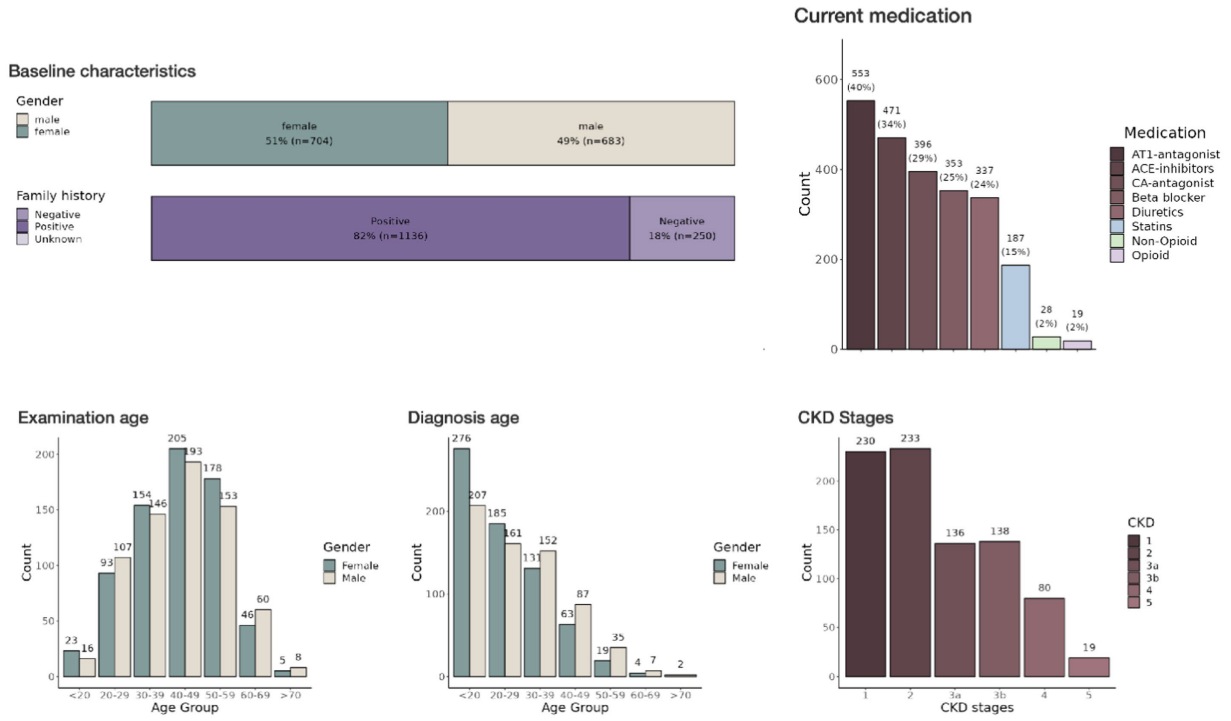

FIGURE 2
Interactive dashboard of the ADPKD registry in a shiny application—This image displays part of the interactive dashboard from the shiny application used to visualize data from the large patient cohort with chronic kidney disease. Several pages and filter options are available. The dashboard itself is segmented into tabs including 'Patient Characteristics', 'Disease Stage', 'Medication', and others, allowing for a comprehensive overview of the data categories. The 'Baseline characteristics' section provides histograms detailing demographic information.

healthcare strategies, significantly improving patient outcomes. In the realm of research, the application provides a rich dataset for analyzing treatment efficacy, patient responses, and disease patterns, thereby contributing to the advancement of medical knowledge and the development of new treatment modalities.

Discussion

The Meda pipeline was developed to bridge health registry data and real-time analysis of the available data. Our key approach was to develop a system where any type of clinical information could be imported, through the provision of simple configuration files, and where data could be displayed in near real-time to the user. Meda restructures and standardizes such information and provides programmable access to this data. While we developed this in the context of clinical registries, its approach can be used for whole clinical databases that over the years have increased in complexity.

The choice of webfront was driven by the requirements within our statistical analyses. While there are a number of real-time visualization frameworks available, such as Grafana (11), Metabase, or Tableau (12), they are not designed to handle clinical information and the underlying statistics within the biomedical domain. The shiny front, in combination with the many R packages available, allows us to generate and display any type of statistical analysis based on the data. These have been widely used in clinical data visualization and several packages have been generated to fulfil the requirements by the relevant health professionals (13–15). Shiny, and therefore R, bring additional obstacles into this development as R is generally slow in utilizing database queries, has a complex memory management, and can be inefficient in the use of data structures. To address these shortcomings we have opted to preprocess the database data every morning, and on demand, which generates the objects required for visualization and statistical analysis and are loaded through serialized R object storage. This results in a much faster visualization but limits the real-time application of our approach. Given that our registry data does not change on a daily basis and that data entry can be delayed based on clinical workload we struck a balance between functionality and overall speed in our approach. Further development of existing, faster, frameworks for visualization would remedy this.

While our tool is not the first visualization platform available (16, 17), our tool expands on the purely visual aspects of healthcare data. As databases across the healthcare sector are growing and are often based on grandfathered implementations developed in the last decades, access to this data is often complex and convoluted. In addition, the interpretation of this huge amount of data is challenging and requires a more computational and visual approach (18). Particularly, the growing number of complex cohorts, with both retrospective and prospective data collection, has proved to be challenging due to the heterogeneity in collection systems, the lack of standardization across healthcare institutions, and differences in ethical considerations. Our tool aims to address a number of these issues by enabling the integration and near real-time representation of data. By interfacing directly with a hospitals clinical data

repository our tool could show important statistics and analyses in near real-time to clinical staff, ensuring an efficient and effective oversight of data entry in various settings as well as allow for AI based decision support systems to be made available (19). While raw data is the preferred data-type, the tool would also be able to collect already computed statistics and integrate data from multiple institutions to visualize the state of healthcare institutions over a larger geographical area while not exceeding the ethical considerations of each institution.

New approaches to sharing data between institutions using the FHIR (Fast Healthcare Interoperability Resource), provides means of interfacing and exchanging data in a save and standardized environment. While our tool does not currently contain a plugin for including FHIR resources, these are often best placed at the database to database interface (20) where our tool performs best. FHIR has been used extensively for data capture, standardization, recruitment, and consent management (20). Our tool can utilize such information directly from the associated database and provide a suitable visualization and update for healthcare professionals. However, direct implementation of such plugins is possible within the framework of MEDA. Our open approach via data class factories and classes can enable any type of direct interoperability with the standards utilized at any given institution.

Overall, we have established a tool that addresses the current scientific and clinical challenges in working with larger cohorts and provides a standardized structure for use within data science groups. We hope to enable a faster and simpler pipeline for clinical questions from data to results and drive the knowledge generation within medicine.

Data availability statement

The data analyzed in this study is subject to the following licenses/restrictions: patient data can only be published using summary statistics as described by the ethical agreements of this cohort. As much of the publication is code to use such data, we do not intend to publish the additional data. Requests to access these datasets should be directed to roman-ulrich.mueller@uk-koeln.de.

Author contributions

JS: Conceptualization, Data curation, Methodology, Software, Writing – review & editing. SA: Data curation, Investigation, Project administration, Validation, Visualization, Writing – original draft, Writing – review & editing. VB: Methodology, Software, Visualization, Writing – original draft. FG: Data curation, Project administration, Writing – review & editing. R-UM: Conceptualization, Data curation, Funding acquisition, Investigation, Supervision, Writing – review & editing. PA: Conceptualization, Methodology, Project administration, Resources, Software, Supervision, Visualization, Writing – original draft, Writing – review & editing.

Funding

The author(s) declare that financial support was received for the research, authorship, and/or publication of this article. The research at hand was supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – CECAD, EXC 2030 – 390661388. R-UM was supported by Marga and Walter Boll-Stiftung.

Conflict of interest

JS was employed by Bonacci GmbH.

The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

References

1. Paganelli AI, Mondéjar AG, da Silva AC, Silva-Calpa G, Teixeira MF, Carvalho F, et al. Real-time data analysis in health monitoring systems: a comprehensive systematic literature review. *J Biomed Inform*. (2022) 127:104009. doi: 10.1016/j.jbi.2022.104009

2. Kannampallil TG, Schauer GF, Cohen T, Patel VL. Considering complexity in healthcare systems. *J Biomed Inform*. (2011) 44:943–7. doi: 10.1016/j.jbi.2011.06.006

3. Lehmann CU, Kim GR, Johnson KB. Pediatric Informatics: Computer Applications in Child Health Springer (2009). doi: 10.1007/978-0-387-76446-7

4. Galitsky B, Goldberg S. Artificial intelligence for healthcare applications and management. Cambridge: Academic Press (2022).

5. Clot-Silla E, Argudo-Ramirez A, Fuentes-Arderiu X. Letter to the editor: measured values incompatible with human life. *EJIFCC*. (2011) 22:52–4.

6. Schweinar A, Wagner F, Klingner C, Festag S, Spreckelsen C, Brodoehl S. Simplifying multimodal clinical research data management: introducing an integrated and user-friendly database concept. *Appl Clin Inform*. (2024) 15:234–49. doi: 10.1055/a-2259-0008

7. Vehreschild JJ, Rüping MJ, Cornely OA. A web-based research portal for rare infectious diseases [text/html]. 10. Kongress Für Infektionskrankheiten und Tropenmedizin (KIT 2010) (2010) 10. doi: 10.3205/10KIT127,

8. Jenkins. (2011). Jenkins. Available at:https://www.jenkins.io/

9. R Core Team. (2021). R: A language and environment for statistical computing. R Foundation for Statistical Computing. Available at:https://www.R-project.org/

10. Chang W., Cheng J., Allaire J. J., Sievert C., Schloerke B., Xie Y., et al. (2024). Shiny: web application framework for R. Available at:https://shiny.posit.co/

11. Abbasian M, Khatibi E, Azimi I, Rahmani AM. PHAS: an end-to-end, open-source, and portable healthcare analytics stack. *Procedia Comp Sci*. (2023) 220:511–8. doi: 10.1016/j.procs.2023.03.065

12. Ko I, Chang H. Interactive visualization of healthcare data using tableau. *Healthcare Infor Res*. (2017) 23:349–54. doi: 10.4258/hir.2017.23.4.349

13. Heinsberg LW, Koleck TA, Ray M, Weeks DE, Conley YP. Advancing nursing research through interactive data visualization with R shiny. *Biol Res Nurs*. (2023) 25:107–16. doi: 10.1177/10998004221121109

14. Miller DM, Shalhout SZ. StoryboardR: an R package and shiny application designed to visualize real-world data from clinical patient registries. *JAMIA Open*. (2023) 6:ooac109. doi: 10.1093/jamiaopen/ooac109

15. Owen RK, Bradbury N, Xin Y, Cooper N, Sutton A. MetaInsight: an interactive web-based tool for analyzing, interrogating, and visualizing network meta-analyses using R-shiny and netmeta. *Res Synth Methods*. (2019) 10:569–81. doi: 10.1002/jrsm.1373

16. Abudiyab NA, Alanazi AT. Visualization techniques in healthcare applications: a narrative review. *Cureus*. (2022) 14:e31355. doi: 10.7759/cureus.31355

17. Elshehaly M, Randell R, Brehmer M, McVey L, Alvarado N, Gale CP, et al. QualDash: adaptable generation of visualisation dashboards for healthcare quality improvement. *IEEE Trans Vis Comput Graph*. (2021) 27:689–99. doi: 10.1109/TVCG.2020.3030424

18. Menon A., Aishwarya M. S, Joykutty A. Maria, Av A. Y., Av A. Y.,. (2021). Data visualization and predictive analysis for smart healthcare: tool for a hospital. 2021 IEEE Region 10 Symposium (TENSYMP), 1–8

19. Alowais SA, Alghamdi SS, Alsuhebany N, Alqahtani T, Alshaya AI, Almohareb SN, et al. Revolutionizing healthcare: the role of artificial intelligence in clinical practice. *BMC Med Educ*. (2023) 23:689. doi: 10.1186/s12909-023-04698-z

20. Vorisek CN, Lehne M, Klopfenstein SAI, Mayer PJ, Bartschke A, Haese T, et al. Fast healthcare interoperability resources (FHIR) for interoperability in Health Research: systematic review. *JMIR Med Inform*. (2022) 10:e35724. doi: 10.2196/35724