Bioinformatics Advances, 2023, **00**, vbad170
https://doi.org/10.1093/bioadv/vbad170
Advance Access Publication Date: 29 November 2023
Application Note

OXFORD

Cheminformatics

MedicaidJS: a FAIR approach to real-time drug analytics

Kunaal Agarwal[1], Hae Rin Kim[1], Jonas S. Almeida [1], Lorena Sandoval [1,*]

[1]Division of Cancer Epidemiology and Genetics (DCEG), National Cancer Institute, Rockville, MD 20850, United States

*Corresponding author. Division of Cancer Epidemiology and Genetics (DCEG), National Cancer Institute, 9609 Medical Center Dr, Rockville, MD 20850, United States. E-mail: sandovall2@nci.nih.gov

Associate Editor: Cecilia Arighi

Abstract

Motivation: As prescription drug prices have drastically risen over the past decade, so has the need for real-time drug tracking resources. In spite of increased public availability to raw data sources, individual drug metrics remain concealed behind intricate nomenclature and complex data models. Some web applications, such as GoodRX, provide insight into real-time drug prices but offer limited interoperability. To overcome both obstacles we pursued the direct programmatic operation of the stateless Application Programming interfaces (HTTP REST APIs) maintained by the Food and Drug Administration (FDA), Medicaid, and National Library of Medicine. These data-intensive resources represent an opportunity to develop Software Development Kits (SDK) to streamline drug metrics without downloads or installations, in a manner that addresses the FAIR principles for stewardship in scientific data—Findability, Accessibility, Interoperability, and Reusability. These principles provide a guideline for continual stewardship of scientific data.

Results: MedicaidJS SDK was developed to orchestrate API calls to three complementary data resources: Medicaid (data.medicaid.gov), Food and Drug Administration (open.fda.gov), and the National Library of Medicine RxNorm (lhncbc.nlm.nih.gov/RxNav). MedicaidJS synthesizes response data from each platform into a zero-footprint JavaScript modular library that provides data wrangling, analysis, and generation of embeddable interactive visualizations. The SDK is served on github with live examples on observableHQ notebooks. It is freely available and can be embedded into web applications as modules returning structured JSON data with standardized identifiers.

Availability and implementation: Open source code publicly available at https://github.com/episphere/medicaid, live at episphere.github.io/medicaid, supplementary interactive Observable Notebooks at observablehq.com/@medicaidsdk/medicaidsdk.

1 Introduction

Rising prescription drug prices call for in-depth, real-time analyses of shifting trends. From 2007 to 2018, net drug prices increased 4.5% annually, outpacing price growth of other health care services (Rome *et al.* 2022). This increase has created significant affordability concerns. Furthermore, since average prices of brand-name drugs have increased, the demand for generic drug prescriptions with the same active ingredients has doubled (Hayford et al. 2022). In such a dynamic cost landscape, increasing transparency of drug metric data in real-time would allow users to effectively gauge drug price affordability.

Tracking drug metric data involves examining drug cost, utilization, toxicity, and contextual patient demographics. This process is facilitated by National Drug Codes (NDCs) but is challenged by the convoluted indexing of NDC storage. NDCs are issued by the FDA, and represent universal product identifiers for human drugs in the USA. They are composed of unique 10 or 11-digit, three-segment numbers representing a label, product, and package code. These indexes are constantly updated and factor in minor changes in manufacturing, nomenclature, dosage. Handling this continual process of regeneration is a core challenge for the development of open-source tools engaging drug metric data.

A rare example of an open-source tool programmatically accessing U.S prescription drug prices is the web application, GoodRx (goodrx.com). GoodRx is a real-time drug price comparison tool that aggregates drug prices and coupons from contracted pharmacies. The design of GoodRx restricts programmatic interoperation by limiting the number of API calls and, at the moment of this writing, not offering new API keys for development of external applications.

While effective for a one-off inspection of the current cost landscape, single purpose applications like GoodRx are not capable of serving as general purpose platforms for developing consumer-facing tools. Moreover, these tools are not configured to systematically track drug prices over time. Such functionality is provided by a modular Software Development Kit (SDK). MedicaidJS was implemented as a public, open-source modular SDK with a design guided by the FAIR principles of stewardship of scientific data (Wilkinson *et al.* 2016, García-Closas *et al.* 2023). The FAIR principles: Findable, Accessible, Interoperable, and Reusable outline a framework for the developing reusable computational resources.

The accessibility principle in FAIRness stresses the importance of standardized protocols for data retrieval. Accordingly, the portability of orchestrating HTTP REST API calls to multiple data sources becomes the central engine accompanying the MedicaidJS SDK. The FAIR principles (Findable, Accessible, Interoperable, and Reusable) define a framework for approaching Data Commons as API ecosystems (Grossman 2019). Moreover, the logistics and privacy

Received: September 8, 2023; **Revised:** November 9, 2023; **Editorial Decision:** November 18, 2023; **Accepted:** November 28, 2023
Published by Oxford University Press 2023.

expectations of user-centric biomedical data processing are central to the development of applications and workflows entirely in the web stack. The modern web stack ranges from the browser JavaScript interpretation runtime to the browser Web Assembly (WASM) compilation target, with ample support for efficient modularization by ES6 modules. MedicaidJS operates under similar architectural guidelines: all SDK methods execute in the same web stack as the data streams it orchestrates.

Approaching the development of FAIR computational applications requires the usage of data commons or cloud-computing infrastructures built as interoperable resources (Grossman *et al.* 2016). The resulting data commons often utilize cloud-hosted data services, programmatically operable through stateless Application Programming Interfaces (HTTP REST APIs). All data sources orchestrated by MedicaidJS fall into this data commons template. SDKs exist as serverless cloud applications and are abstractions to REST API methods. MedicaidJS leverages the architectural advantages of a standard SDK and incorporates novel computational workflows.

A common approach to evaluate the FAIRness of an application is a comparison to the FAIR maturity model (Bahim *et al.* 2020). This model provides essential guidelines required for an application to be yielded as FAIR. Such guidelines are divided by each of the core FAIR principles and have an associated priority. To prove the FAIRness of MedicaidJS requires the proof that the data APIs it uses are FAIR. Each of these APIs complies with the findability requirement as both the metadata and data contain persistent and unique identifiers. An example of persistent metadata can be seen through the data JSON objects returned by the Medicaid API. All data objects contain specific unique identifiers. The accessibility requirements are also followed through the ability to use HTTP GET and POST requests to access the data. Similarly, interoperability is shown through the standardization of the identifiers and the protocols to retrieve them. Finally, reusability is indicated because community standards for metadata and the data itself are followed (Grossman 2019).

2 Methods

MedicaidJS began as a minimal abstraction to handle data fetching from the Medicaid API (data.medicaid.gov). The library then organically evolved to orchestrate complementary API calls to RxNorm (lhncbc.nlm.nih.gov/RxNa) and the Food and Drug Administration (open.fda.gov). The development process followed the principles of Lean Development. As a result of this process where minimal viable products seed modular composition (Fig. 1), MedicaidJS was developed entirely in-browser as a Javascript SDK. The resulting software product includes documentation, code samples, and guides that developers can integrate into their own web applications. Its stateless service architecture is defined as a serverless execution model (Almeida *et al.* 2019, Grzesik *et al.* 2022). In compliance with modern SDK guidelines, the modularization adhered to ECMAScript (ES6) standards.

MedicaidJS data workflows exist in four modular components based on the following Medicaid datasets: National Average Drug Acquisition Cost (NADAC 2013–2023), State Drug Utilization (2013–2023), Healthcare Quality Measures, and miscellaneous datasets. The operation of these modules is demonstrated by live interactive, collaborative documents known as Observable Notebooks (see Availability).

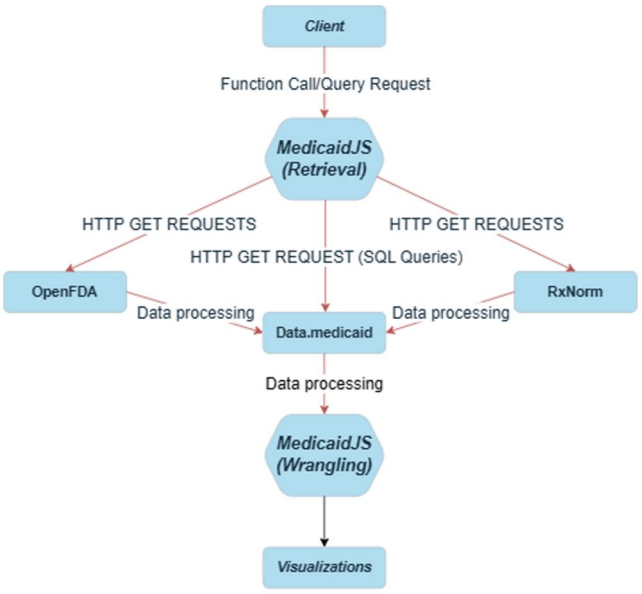

Figure 1. UML diagram representing data flow between MedicaidJS modules (hexagons) and API data resources (rectangles).

[SELECT data_variables FROM distribution_id][WHERE data_variables = "value"][LIMIT x OFFSET y]

Equation 1. Composition of a SQL-like query.

Data retrieval marks the initial step of every workflow in the library. Users can selectively fetch data through HTTP requests and SQL-like queries. In the case of the Medicaid API, every built-in drug metric workflow defaults to SQL-like queries (Equation 1). Moreover, data retrieval facilitated by SQL-like queries configures the low level API calls to optimize performance and enacts data aggregation within the query response. This retrieval strategy allows for further algorithmic efficiency through parallelization, multithreading, and data caching. The scalability of this solution is particularly clear for the NADAC datasets (up to 1.5 million rows).

The need for harmonization across data sources was a significant preoccupation in SDK development. Fortunately, both NADAC and Drug Utilization data objects include a unique NDC (National Drug Code) and a generic name. The caveat is that generic names can have multiple NDCs, due to differences in manufacturing practices and dosages. To handle this dual indexing structure, each workflow allows users to simultaneously analyze individual NDCs, custom compilations of NDCs, or generic names where all NDCs are combined.

NADAC data undergoes an additional level of wrangling by distinguishing between two fields: "as_of_date" and "effective_date." As_of_date refers to the date of data retrieval but does not necessarily represent when prices take effect. In contrast, effective_date marks when drug pricing information becomes applicable for reimbursement and other pricing purposes. Either can be toggled, with the default as effective_date.

3 Results

Some MedicaidJS methods were developed to respond to user-requested real-time visual analytics (Fig. 2). The modular design goes beyond data wrangling and standardization

by supporting extensible visual analytics and assisting with the interactive display of the analysis results. This tactile verification also plays a role in quality assurance and quality control checks. For example, the State Drug Utilization workflow includes capabilities for outlier filtration through an interquartile range analysis. This filtration significantly improves visualization quality and allows for deeper statistical analysis as it accentuates data variance (Fig. 2A and B).

The MedicaidJS visualization functions rely on two dependencies dynamically imported through a Content Delivery Network (CDN). The first dependency is PlotlyJs, a cross-language, declarative charting library. The second dependency is the LocalForage API which mediates the use of IndexedDb, the NoSQL database embedded in the modern web browser. For older browsers, localForage defaults to WebSQL, the in-browser implementation of the relational database SQLite. In-browser persistent storage provides the critical resource for data caching and asynchronous querying of data. The implementation of the suite of asynchronous MedicaidJS data processing methods was approached with a functional coding style taking full advantage of the in-browser computational environment, such as order-free map-reduce operations.

The same source code, versioned by GitHub pages, is used by the MedicaidJS library and by the live application (i.e. dependencies, distinct between data processing and interactive analytics, are resolved dynamically). This modular reusability is illustrated in the figures below, rendered as a compilation of plots independently generated by MedicaidJS methods (Fig. 2).

4 Conclusion

Prescription drug prices have increased substantially in the past few years. This makes it imperative to improve drug prescription transparency by utilizing CMS and FDA real-time data resources. The accompanying extensible MedicaidJS Software Development Kit (SDK) reported here was developed to address this goal. MedicaidJS will orchestrate fetching data through HTTP GET attribute-pair requests, or SQL-like queries, from the following APIs: Medicaid, the Food and Drug Administration, and the National Library of Medicine RxNorm. MedicaidJS workflows are composed of four modular components that clean, process, and visualize data upon user request.

MedicaidJS is currently being tested by public health researchers, data scientists, and software developers at the National Cancer Institute. The software is being woven into existing projects to create real-time drug metric dashboards.

There are many future improvements and applications that would benefit from the MedicaidJS functionality. We anticipated the scenario by paying particular attention to the modularity of this open source library, relying on the ES6 mechanisms. We hope that MedicaidJS can act as a framework for future projects regarding drug data.

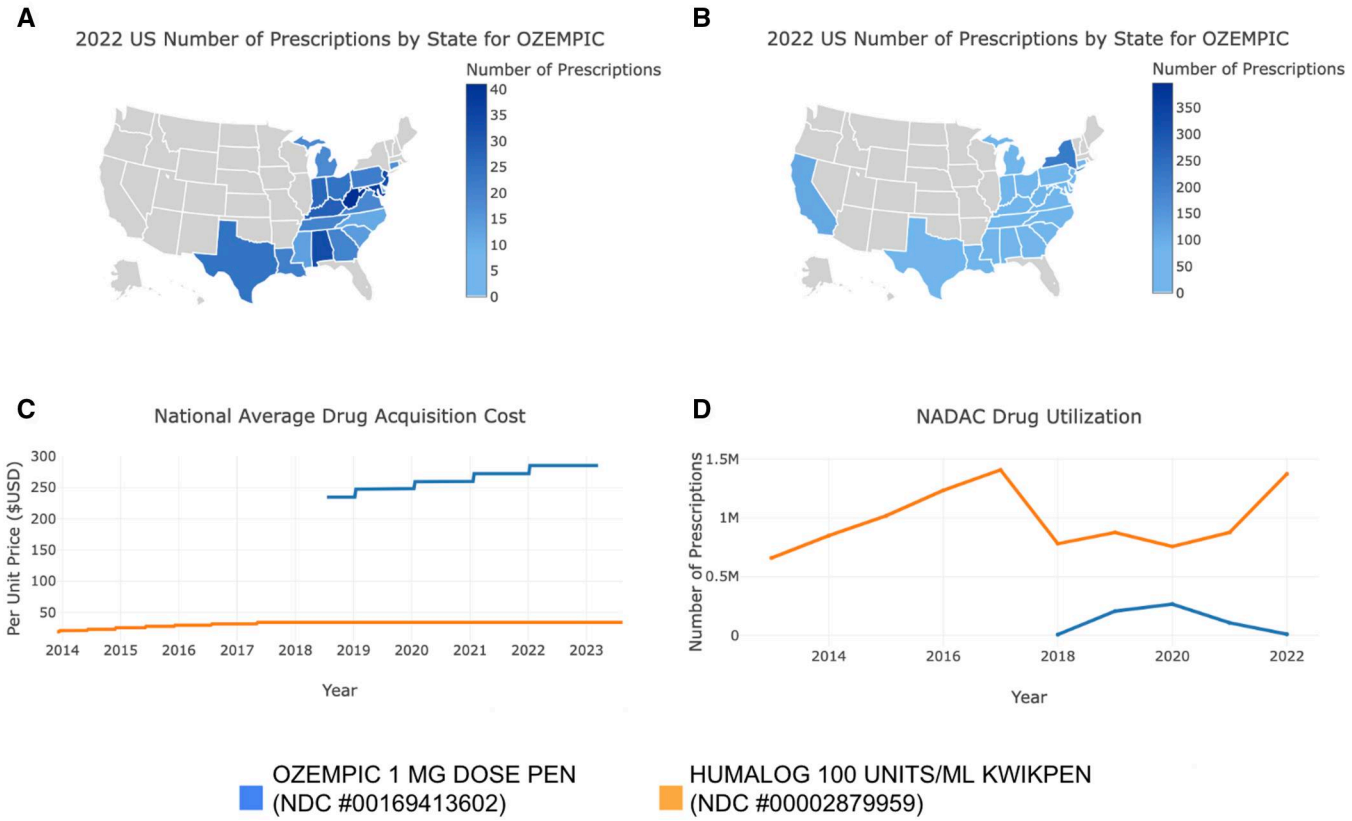

Figure 2. Compilation of MedicaidJS plots. (A and B) Comparison of two choropleth maps representing state drug utilization. Map A filters outliers, while B includes outliers. (C) Time series of per unit price for OZEMPIC 1 MG DOSE PEN (NDC 00169413602) and HUMALOG 100 UNITS/ML KWIKPEN (NDC 00002879959) (2014 to 2023). (D) Time series of number of prescriptions in the USA representing drug utilization for OZEMPIC 1 MG DOSE PEN (NDC 00169413602) and HUMALOG 100 UNITS/ML KWIKPEN (NDC 00002879959) (2013 to 2022). A webcast showcasing the use of MedicaidJS to generate these four plots is available with the live deployment (see Availability).

Author contributions

Kunaal Agarwal (Data curation [equal], Methodology [equal], Project administration [equal], Software [lead], Writing—original draft [lead], Writing—review & editing [lead]), Hae Rin Kim (Conceptualization [equal], Data curation [equal], Methodology [equal], Writing—review & editing [supporting]), and Lorena Sandoval (Conceptualization [equal], Methodology [equal], Project administration [equal], Visualization [equal], Writing—review & editing [supporting])

Funding

None declared.

Conflict of interest

None declared.

References

Almeida JS, Hajagos J, Saltz J *et al.* Serverless OpenHealth at data commons Scale-Traversing the 20 million patient records of New York's SPARCS dataset in real-time. *PeerJ* 2019; **7**:e6230.

Bahim C, Casorrán-Amilburu C, Dekkers M *et al.* The FAIR data maturity model: an approach to harmonise FAIR assessments. *Data Sci J* 2020;**19**:41.

García-Closas M, Ahearn TU, Gaudet MM *et al.* Moving toward findable, accessible, interoperable, reusable practices in epidemiologic research. *Am J Epidemiol* 2023;**192**:995–1005.

Grossman RL. Data lakes, clouds, and commons: a review of platforms for analyzing and sharing genomic data. *Trends Genet* 2019; **35**:223–34.

Grossman RL, Heath A, Murphy M *et al.* A case for data commons: toward data science as a service. *Comput Sci Eng* 2016;**18**:10–20.

Grzesik P, Augustyn DR, Wyciślik Ł *et al.* Serverless computing in omics data analysis and integration. *Brief Bioinform* 2022; **23**:bbab349.

Hayford T *et al.* CBO Report. Prescription Drugs: Spending, Use, and Prices. Congressional Budget Office, 2022. https://www.cbo.gov/publication/57050.

Rome BN, Egilman AC, Kesselheim AS *et al.* Trends in prescription drug launch prices, 2008-2021. *JAMA* 2022;**327**:2145–7.

Wilkinson MD, Dumontier M, Aalbersberg IJ *et al.* The FAIR guiding principles for scientific data management and stewardship. *Sci Data* 2016;**3**:160018.