1]Jacob S. Zelko 3]Fareeda Abdelazeez 1, 2]Malina Hy 1, 2]Varshini Chinta 3]Jay Sanjay [1]Georgia Tech Research Institute [2]Georgia Institute of Technology [3]Foobar

Developing Observational Health Research Capacity in the Julia Ecosystem

[

ABSTRACT

Observational health continues to be a growing field in health informatics research as electronic health records (EHR), patient medical claims, and other ancilliary patient data source become more readily computable and accessible to researchers. JuliaHealth is poised as an ecosystem to innovate within this area of research by bringing highly performant analytics approaches, composable solutions, and interoperable software that leverages prior state of the art. This paper will discuss the state of the art observational health research tools within the JuliaHealth ecosystem and how JuliaHealth is prepared to further research goals within this domain.

Keywords

Observational Health, OMOP Common Data Model, Database Management, Characterization

1. Introduction

Over the past 10 years, there have been several innovations within the observational health research community. Arguably, the most crucial innovation that has emerged is the Observational Medical Outcomes Partnership Common Data Model (OMOP CDM) which was designed specifically to standardize observational health data to enable transferable analyses. The importance of this model has become apparent within open science organizations like the OHDSI (Observational Health Data Science and Informatics) network where members across 100+ countries with access to various health systems have been able to successfully map their patient data to the OMOP CDM. This has led to successful collaboration relationships across countries where one research group can develop a network study and then disseminate the study across collaborators leading to the idea that you can "write once, run everywhere."
The OMOP CDM itself does not require a specific technology to work with the data stored in this standard. It features a person-centric design where each domain records personal identity while prioritizing data protection through the limiting of information that could endanger patient anonymity. As the OMOP CDM has become a cornerstone in running these widely distributed network studies, special attention to developing tools around handling OMOP CDM data is necessary to meaningfully contribute within the observational health research space.

1.1 What Is JuliaHealth?

JuliaHealth is an open-source collective dedicated to bringing the Julia programming language to various avenues of health research domains such as observational health, medical imaging, standards interoperability, etc. It leverages the strengths of the Julia language to introduce new approaches to these spaces, augment and improve existing workflows, and provide a central community structure for health researchers both part of and entering into the Julia ecosystem. Eventually, JuliaHealth will host a variety of subecosystems and associated resources that will enable novel research to be conducted within Julia.

1.2 Status of Conducting Observational Health Research in Julia

JuliaHealth itself is a loose collective of various research software engineers, health researchers, and hobbyist programmers that had its origins in $\approx$ 2020. Being that JuliaHealth is a federated network of contributors, varying aspects of the ecosystem have developed at differing rates as a function of multiple different factors such as developer time, workplace incentives, and more. One area of the subecosystems within JuliaHealth that has matured significantly over the years has been the observational health subecosystem (referred to as OHS for the remainder of this paper).
Although difficult to date, it can said that OHS had its inception in work done by Evans and Simonov with efforts such as *FunOHDSI.jl*, *OHDSICohortExpressions*, and *FunSQL.jl* around 2021 [5] Since then, Zelko et. al. have spearheaded most recent developments within OHS via the creation of a variety of tools and demonstrated their effectiveness in using these tools for novel observational health research endeavors [7] Through such efforts, OHS is becoming further capable of executing novel observational health research although, understandably, not as mature as similar ecosystems.
In particular, the strongest influence and inspiration to OHS is the OHDSI open-science network. OHDSI (Observational Health Data Sciences and Informatics) is a global, collaborative, interdisciplinary, and open network of researchers, healthcare professionals, students, patients, and other entities committed to large-scale health data analytics and generating reliable evidence, with the goal of promoting better health decisions and better care. Since current observational health data sources are aggregated at different levels and cannot be generalized to the overall population, the data is often incomplete, inaccurate, and prone to issues of bias, error handling, and multidimensional complexity. The goal of OHDSI is to produce reproducible, robust, and reliable evidence by developing standardized data formats and research protocol designs to enable large-scale collaborative research and improve the transparency and scalability of observational health research.
OHDSI had its inception in the development of the Observational Medical Outcomes Partnership (OMOP) that was created in 2009 by the US Food and Drug Administration to address concerns related to data type, study design, and data privacy concerns of "Real World Data" [2] [6] What could be considered the magnum opus of the OMOP group was the development of what is called the OMOP

Common Data Model (OMOP CDM) to house and ingest this sort of data in a consistent format that has been optimized to assist in extracting meaningful population health information. In brief, the OMOP Common Data Model (CDM) is a "person-centered" model that has standardized common events in a given patient encounter with database tables that represent the unique structures found in patient events. This creates a longitudinal view of all healthcare-related events per person recorded in a singular health system. OHDSI was developed to focus on leveraging observational health data stored within the OMOP CDM.

At this time, OHS prioritizes creation and maintenance of tools that support analysis of "OMOP-ified"[1]data (that is, data which is formatted in the OMOP CDM), in much the same manner that OHDSI does.

OHDSI supports the development of open-source software tools in a centralized ecosystem called HADES (Health Analytics Data-to-Evidence Suite) for particularly managing and working with OMOP CDM formatted data. The HADES ecosystem has been in existence for ≈ 10 years and has given rise to improvements in the quality and reliability of observational health research as well as motivating novel collaboration mechanisms.

In particular, one collaboration mechanism that OHDSI has been able to operationalize meaningfully while working with observational health data is the notion of "network studies". These studies, are designed to be transparent and encompasses research endeavors being explored by members of the OHDSI comprehensive of relevant documentation, analysis code, results, and study protocols describing the study's scope and objectives. One core innovation brought about by OHDSI in the space of network science is how, given the HADES tools and procedures established by OHDSI, a research group can develop their analysis on their own OMOP CDM instance and then bundle together their study. Once this packaging is done, they can pass along the study package to another site to collaborate on the same analysis but on different data assets. Data confidentiality is preserved in this approach as data from individual participating sites remains exclusive to each site, with only aggregate results shared across a given collaborator network network. As such, no risk for re-identification comes about through this network collaboration and has ushered in the possibility for similar research approaches to be taken across the globe. [2]

One of the premier tools within HADES is the ATLAS Shiny application which was developed to "support the design and execution of observational analyses to generate real world evidence from patient level observational data."

2. Describing Common Observational Health Research Tasks within the Julia Ecosystem

Achieving the notion of a network study in an OHDSI-sense is a highly valuable goal to multiple communities. JuliaHealth now has capacity to do just the same and can harness the best of OHDSI and the best of Julia to achieve novel observational health workflows. The JuliaHealth observational health subecosystem (referred to as OHS for the remainder of this paper) supports a variety of use cases designed to support and enable investigations into observational health data. This paper summarizes some of the current capacity OHS possesses to conduct such endeavors. In particular, the following supported areas will be discussed:

[1]I am unsure of where this notion came from. I believe it may have originally been coined by my former mentor and OHDSI founder, Dr. Jon Duke. The provenance of related terms like "OMOP-ification" (the process of converting a database to the OMOP CDM) is even murkier.

(1) **Using Phenotype Definitions** - brief overview of phenotype definitions and how to work with them

(2) **Rapid Cohort Iteration and Creation** - how to use JuliaHealth tools to iteratively develop patient cohorts to explore

(3) **Working with Patient Databases** - current approaches to working with different patient databases

(4) **Ecosystem Interoperability** - how one can bridge between JuliaHealth's ecosystem to other research ecosystems

2.1 Using Phenotype Definitions

Broadly, ATLAS has become the de facto tool to develop patient phenotype definitions within observational health research contexts. These phenotype definitions have varying levels of complexity where they can be very simple (finding all patients with one health condition or diagnosis) or grow to be extremely complex (defining controlled, uncontrolled, and indeterminate patient cohorts based on medications, lab results, etc.) The computable phenotype definitions that ATLAS generates based on phenotype definitions are in a variety of serialization formats such as JSON (see A) or prepared SQL expressions available in multiple SQL flavors (see B).

The Julia package, OHDSICohortExpressions.jl, exists specifically to ingest these serialized expressions of computable phenotype definitions and create patient cohorts that match the given criterion in the definition. How this tool works is to 1) read and verify the serialized computable phenotype definition 2) reinterpret the serialization into the FunSQL.jl Domain Specific Language 3) return a prepared SQL statement against a given SQL database that queries an OMOP CDM database for the desired patient population(s) of interest. Once these prepared statements are made, researchers can execute the generated SQL against their OMOP CDM instance to generate cohorts for future analyses.

2.2 Rapid Cohort Iteration and Creation

Although OHDSICohortExpressions.jl is sufficient to translate existing phenotype definitions generated in ATLAS to executable queries, oftentimes, there is a tricky problem of how to best iterate on phenotype definitions. [8] Often, one has to take an existing definition and either revise to a more narrow cohort or write manual code to build more niche cohorts after an initial definition is created. This refactoring of a phenotype definition can lead to conflating ideas within conversation between expert clinical reviewers, analysts, and health researchers. This broadly can be the notions of what we are exploring (the phenotype definition; often the responsibility of a clinical reviewer), what is possible to compute upon (the computable phenotype definition; often the responsibility of an analyst), and the goals of the overall study. For situations such as these, the tool OMOPCDMCohortCreator.jl was created.

OMOPCDMCohortCreator.jl allows one to iteratively build a patient population quickly and easily. One can use patient pools developed from pre-existing cohorts defined off of phenotype definitions as a starting point in exploring patients or create new patient populations quickly without the need for a well-defined phenotype definition. This can enable analysts to more quickly understand what is inside their data assets, do "quick" analysis of a particular smaller research question, and enable a more distinct separation of work between a clinical expert reviewer and analyst teams. It is highly recommended to still create phenotype definitions about particular populations one explores.

2.3 Working with Patient Databases

One of the biggest styming factors within not only the Julia ecosystem, but observational health research in general has been the difficulty of working with different databases. Although the OMOP CDM has been great for standardizing *how* real world data is handled, there is a lack of consensus on *what* is needed to handle these data assets. For a variety of reasons, different groups will choose different database architectures (perhaps PostgreSQL for improved permission support, DuckDB for performance, or even SQLite for ease of use) but all are subtly different enough to create headaches in seamlessly working across different research sites.
The way that the Julia ecosystem handles this is via the creation of a common database interface called the DBInterface. Specific Julia database packages can implement dispatches for this interace allowing users to have a much better experience working across databases. In spite of the such benefits, for some users (and especially within the context of observational health studies), it can beneficial to have a more prescriptive interface where one just needs to pass perhaps a password, username, database name, and schema. For that reason, we DBConnector.jl was created.
Although this is not an explicit JuliaHealth package, it was a pain point that was experienced throughout the process of observational health research across multiple partner sites. The promise of DBConnector.jl is that with one line command, users can connect simply, and in a common (albeit opinionated) manner, to databases. DBConnector unifies Julia database packages that implement the DBInterface to achieve this level of simplified connection making. Finally, if someone needs more fine-grained control over a particular database connection, they can choose to eschew this package. Instead, they can switch to the specific Julia database package that would give them the additional control. Although they'll lose the simplicity of this package, core required functionality is not lost when working with databases.
DBConnector.jl was inspired by the package *DatabaseConnector* (Link: https://github.com/OHDSI/DatabaseConnector)

2.4 Ecosystem Interoperability

In order to more effectively bridge the JuliaHealth ecosystem with ongoing efforts in other ecosystems, Julia provides methods to allow one to seamlessly interoperate with not only other ecosystems but entirely different languages as well. For example, through the work of packages like RCall.jl or PythonCall.jl, one can utilize R or Python packages and interfaces directly within Julia. [3] [1] This embedding allows one to patch one's workflow where proper tools are missing with the equivalent package in Python or R.
One particular ecosystem is the OHDSI Health Analytics Data-to-Evidence Suite (HADES) ecosystem where the majority of its tools are written in a combination of R and Java. HADES is a collection of approximately 20 packages, developed by the OHDSI community to interact directly with the OMOP CDM to support large-scale analytics. The goals of HADES is much the same as the JuliaHealth OHS such as supporting cohort construction, population-level estimation, patient-level prediction, and calculating other sorts of relevant research artifacts.
For HADES tools that expose APIs, rather than having to somehow wrap around this with a tool like RCall.jl, instead, HADES software can continue to run as a service and OHDSIAPI.jl can be used. OHDSIAPI.jl is a Julia wrapper around a variety of OHDSI API-based services that can directly interface with such services. This can be a great bridging mechanism so that observational health researchers can continue doing work and still interact with necessary research endpoints as needed without interrupting their workflow.

3. Conducting a Small-Scale Observational Health Research Study Using JuliaHealth Tools

With the JuliaHealth tools that have been created, we can now demonstrate an observational health study workflow that would be used in practice. For this paper, we used two datasets: a synthetic patient dataset called *Eunomia* and the MIMIC III patient dataset which had been converted into the OMOP CDM to illustrate the workflow provided by the Observational Health sub-ecosystem. Reproducing the workflow using the *Eunomia* dataset is made readily available at the repository here: [4]
Eunomia will be used to drive the workflow example looking at patients who have ever had a history of strep throat. Results generated by the *Eunomia* dataset will be shown for pedagogical purposes although the actual results of that analysis will be nonsensical given the synthetic nature of the database. However, to complement this, a similar study will be done on the MIMIC III dataset characterizing individuals with a history of Type 2 Diabetes Mellitus to encourage discussion on what is practically possible with these tools.

3.1 Connecting to a Database

```
import DBConnector:
  DBConnection
import HealthSampleData:
  Eunomia
import OMOPCDMCohortCreator as OCC

conn = DBConnection("sqlite", Eunomia())

occ.GenerateDatabaseDetails(
    :sqlite,
    "main"
)

occ.GenerateTables(conn)
```

3.2 Bridging between OHDSI and JuliaHealth Communities

Briding between tools used in the OHDSI and JuliaHealth communities is now exceptionally straightforward.

3.3 Using a Cohort Definition To Build an Initial Cohort

After having a strep throat phenotype definition defined using ATLAS, we can then execute the resulting computable phenotype definition against the *Eunomia* database.

```
using JSON3
using OHDSICohortExpressions

import DBInterface as DBI

cohort = JSON3.read("strep_throat.json")
cohort_expression = cohort[:items][1][:expression]

model = Model(cdm_version = v"5.3.1",
              cdm_schema = "main",
              vocabulary_schema = "main",
              results_schema = "main",
                target_schema = "main",
                target_table = "cohort");

sql = translate(cohort_expression,
                dialect = :sqlite,
                model = model,
                cohort_definition_id = 1);

for query in split(sql, ";")[1:end-1]
  DBI.execute(conn, sub_query)
end
```

A quick query of the cohort table (where the resulting subjects were stored) shows ≈ 1600 from the dataset match this criteria.

3.4 Characterizing Patient Populations

```
C = GetCohortSubjects(1, conn)
rename!(C, :subject_id => :person_id)

C_race = occ.GetPatientRace(C.person_id, conn)
C_gender = occ.GetPatientGender(C.person_id, conn)

C_age_group = occ.GetPatientAgeGroup(
  C.person_id,
  conn;
)
```

3.5 Calculating Crude Prevalence Rate

```
import DataFrames as DF

C_features = DF.outerjoin(C_race,
                          C_gender,
                          C_age_group;
                          on = :person_id,
                          matchmissing = :equal)

C_features = C_features[:, DF.Not(:person_id)]

C_groups = DF.groupby(C_features,
                      [
                        :race_concept_id,
                        :gender_concept_id,
                        :age_group
                      ])

C_groups = DF.combine(C_groups, DF.nrow => :count)
```

Using this analytical approach, tabulation can be quickly conducted across patients. Additionally, throughout the process, one can perform relevant analysis methods. Here, we'll conduct a crude prevalence calculation across the patient cohort.

Race	Gender	Age Group	Cohort	Total	Prev. (%)
B/AA	F	60 - 64	13	21	61.90
W	F	50 - 54	59	111	53.15
W	F	55 - 59	62	97	63.91
A	M	80 - 84	6	10	60.00
UNK	F	60 - 64	29	39	74.35
B/AA	M	65 - 69	12	18	66.66

3.6 Summary of Results

As seen in Table 1 and Table 2, informative results can be rapidly generated by this workflow relevant to observational health research. Although the values shown in Table 2 are synthetic and the crude prevalence values themselves are meaningless, we can take a this approach and apply it directly to the MIMIC III dataset. In Table 3, we can calculate across this dataset meaningful statistics that could later be further analyzed and drive further questioning (such as ...).

4. Discussion

4.1 Advanced Features of the JuliaHealth Ecosystem

1. Modularization of code and chaining 2. Distributed computing

4.1.1 Modularization of Code and Chaining. An exciting emergent aspect of the JuliaHealth ecosystem that came about as a result of developing OMOPCDMCohortCreator.jl was the prospect of radical modularization and chaining of package functionalities together. For example, taking our workflow example from the earlier section where we characterized patients by race, gender, and age group, we can do a few "tricks" to modularize these characterization functions as follows:
We can leverage a unique property of Julia in that Julia supports partial evaluation of functions (also known as currying) to simplify repeated function calls to the same function over and over again. First, we'll take our original functions and fix a connection object to each of our cofactor functions. This simplifies having to continuously pass forward a connection object and opens up the ability to chain together these functions simply:

```
import Base:
  Fix2

import OMOPCDMCohortCreator:
  GetPatientRace,
  GetPatientGender,
  GetPatientAgeGroup

CGetPatientRace = Fix2(GetPatientRace, c)
CGetPatientGender = Fix2(GetPatientGender, c)
CGetPatientAgeGroup = Fix2(GetPatientAgeGroup, c)
```

NOTE: The "C" prefix for each of our curried functions denotes that it is curried.
Second, as OMOPCDMCohortCreator.jl is built on top of DataFrames.jl and one has the option of creating a DataFrame with the result from every OCC function, we can use the package, Chain.jl. This package abstracts away some of the explicit functionalities of DataFrames.jl but instead gives much easier ways to define a flow of functions:

```
import Chain:
  @chain

CofactorPatients(patients) = @chain patients begin
    CGetPatientRace_fixed
    CGetPatientGender_fixed
    CGetPatientAgeGroup_fixed
end
```

```
CharacterizePatients(patients) =
@chain patients begin
    _[:, DF.Not(:person_id)]
    DF.groupby(_, names(_))
    DF.combine(_, DF.nrow => :count)
    occ.ExecuteAudit
end
```

This gives one the ability to simply reason about an analysis in a very straightforward and readable manner:

```
RunStudy(patients) = @chain patients begin
    CofactorPatients
    CharacterizePatients
end
```

In this way, future researchers using tools such as OMOPCDM-CohortCreator.jl can create these more convenient abstractions. In fact, it could be a point of exploration in the future to build on top of OMOPCDMCohortCreator.jl even more abstracted and reusable functionalities to simplify ease of use while also enabling more readily auditable functionality. Finally, as these functions have now been fully modularized, these can now be readily distributed using Julia's distributed computing methods:

```
import Distributed:
  @distributed

result = @distributed (append!) for i in 1:10
   vals =
      RunStudy(strep_patients[i*100 +
1:(i+1)*100, :])
   [vals]
end

vcat(result...) |> audit_patient_groups
```

Distributed computing here then allows a researcher to run multiple investigations at once as well as optimize performance to a given database.

4.1.2 Auto-Generating SQL from JuliaHealth Commands. One crucial aspect of the OHS is that it is unavoidable to work with databases that contain patient health information. For better or worse, the tool that is used to best interact with these databases are SQL dialects that loosely follow consistently the ANSI SQL guidance. To accomodate for this, several of the core tools developed in this sub-ecosystem generate SQL that can be used in the case where Julia may not be available in a given environment housing patient health data. As an example, *OMOPCDMCohortCreator.jl* functions build upon *FunSQL.jl* allowing one to use a function such 'GetPatientAgeGroup' without a connection object which would result in SQL.

```
  GetPatientAgeGroup([1]; age_groupings =
[[0, 40], [41, 80]])
```

```
  SELECT
    "PERSON_2"."person_id",
    (CASE
      WHEN ("PERSON_2"."age" < 41)
        THEN '0 - 40'
      WHEN ("PERSON_2"."age" < 81)
        THEN '41 - 80' END) AS "age_group"
  FROM (
    SELECT
      "PERSON_1"."person_id",
      (2023 - "PERSON_1"."year_of_birth") AS "age"
    FROM "PERSON" AS "PERSON_1"
    WHERE ("PERSON_1"."person_id" IN (1))
  ) AS "PERSON_2"
```

4.2 Strong Composition within JuliaHealth

Within Julia, one aspect of the language that is often viewed as a technical problem within other languages emerges more as a social problem within Julia. That problem is the problem of strong composition within Julia. What composition within Julia is thought to be generally is that a user of Julia packages A and B can combine these packages oftentimes seamlessly together in such a way as to solve a problem that the original designers of A and B did not think about. This has the virtue of Julia packages being strongly flexible to suit problems at hand but a discoverability problem where users developing these novel compositions may not share about these combinations. As a result, there arise a paradox wherein strongly compositional systems, like Julia, simultaneously give rise to obfuscational logic.
Within JuliaHealth, this problem is an active area of exploration. During the summer of 2023, this problem was one of the focuses of Google Summer of Code mentee, Fareeda Abdelazeez. Originally, the project dealt with determining if another package in the JuliaHealth ecosystem was necessary to support prediction for patient outcomes. Instead, what was realized instead is that what should be further investigated is novel compositions that can be viewed through the lens of JuliaHealth. As shown in this workflow, although there were some packages that were made specifically for JuliaHealth, other packages that were not JuliaHealth specific were used alongside of it that composed flawlessly with the rest of the JuliaHealth OHS.

4.3 Potential Future Directions

As shown in this work, there is a rich opportunity for the JuliaHealth organization to become a strong presence within observational health research. JuliaHealth is now sufficiently at a stage of maturity where potential future researchers can leverage JuliaHealth's tools to conduct further future research or build upon existing architecture to target specific needs. Some potential future directions that can be taken are as follows.
Due to the strong composition that exists within the JuliaHealth and broader Julia ecosystem, solutions may already exist for solving various health informatics related research. However, what has consistently been a problem within the Julia ecosystem is the lack of sufficient documentation dedicated to describing these novel compositions and their application to various endeavors. Future efforts can be dedicated to taking existing tools and reframing them in a JuliaHealth context to assist newcomers to Julia and JuliaHealth in determining how to explore and address problems that they are interested in.
Additionally, due to the composable and modular nature of the OHS, research analyses and visualizations can be decoupled and interchanged with one another. Whereas in other ecosystems that make technologies for tools such as dashboards tightly coupled to analytics regimens, the subecosystem is mature enough to enable new users and developers to create analytics tools that can be separately woven into more graphical user interfaces. For example,

treatment pathways is a novel observational health approach to understand and visualize the care a patient receives within a care setting. Only a few tools exist to visualize such hidden patient narratives but with the JuliaHealth ecosystem, the possibility of making tools that can interrogate patient care pathways more deeply could be fully realized.
Furthermore, again exploiting the composition within the JuliaHealth ecosystem, as mentioned in the context of JuliaHealth and machine learning applications, future efforts could focus on developing fairness and bias auditing tools in patient datasets. These tools could consider factors commonly explored in health equity literature like data completeness, demographic variables, and fairness algorithms.
Moreover, there's potential to advance clinical informatics by developing a flexible clinical type system, supporting computable phenotype definitions and medical ontologies. The ecosystem can also benefit from tools to audit the fairness of phenotype definitions. Finally, as the landscape of data expands, integrating non-traditional sources like social media and environmental data into standardized patient records will be a critical frontier. In these ways, JuliaHealth is primed to drive innovation in observational health informatics, pushing the boundaries of data-driven healthcare research.

5. Conclusion

In conclusion, this paper has demonstrated the capabilities of the JuliaHealth OHS and its potential to further investigations in observational health research. Although it is an ecosystem that is relatively new, its robustness allows for novel investigations to be pursued within this domain of research. Additionally, due to the strongly compositional nature of the Julia ecosystem, these JuliaHealth tools can readily compose with tooling to support in-depth statistical analyses, mathematical modeling, and visualization capacities.
The deep dive into the JuliaHealth ecosystem unravels its arsenal of tools tailored for observational health research, meticulously designed to navigate the complexities of handling OMOP CDM data. Notably, the sub-ecosystem within JuliaHealth presents mature, internationally federated research capabilities, building upon existing innovations like OHDSI's HADES, while offering novel tools such as OMOPCDMCohortCreator.jl. In considering future directions, the paper posits the promising avenues for JuliaHealth, envisioning expanded documentation, the development of visualization tools, and integration of machine learning applications for fairness auditing. The ecosystem's potential to revolutionize clinical informatics by integrating non-traditional data sources, auditing phenotype definitions, and fostering innovation in data-driven healthcare research is paramount. In essence, the JuliaHealth ecosystem stands as a catalyst for transformative observational health research, poised to push the boundaries of innovation, collaboration, and standardization in the pursuit of better healthcare outcomes. As it continues to evolve, JuliaHealth emerges as a beacon of progress in the dynamic landscape of health informatics.

6. Acknowledgements

For this work, we would like to thank and recognize the following individuals and groups:

(1) Dr. Jon Duke for his early support
(2) Kristin Kostka for being an early champion in trying novel ideas
(3) Dilum Aluthge for helping this work find a home in JuliaHealth
(4) Adam Black for thinking through potential approaches
(5) The Roux Institute's OHDSI Center for their encouragement
(6) The JuliaHealth community and contributors for fearlessly trying to push boundaries on what observational health informatics can truly be

7. References

APPENDIX

A. JSON Serialization of Computable Phenotype Definition

An example of the JSON serialization of computable phenotype definitions can be found within the paper repository here:
In particular, this example seeks to identify individuals with Type 2 Diabetes Mellitus within a given OMOP CDM.
This was generated using ATLAS Version 2.12.1 as well as WebAPI Version 2.12.1. A generic definition of the schema that was created for ATLAS 2.7.1 can be explored here: https://github.com/OHDSIBr/ATLAS-JSON-Schema/tree/2.7.4

B. PostgreSQL Serialization of Computable Phenotype Definition

An example of a PostgreSQL SQL serialization of computable phenotype definitions can be found within the paper repository here:
In particular, this example seeks to identify individuals with Type 2 Diabetes Mellitus within a given OMOP CDM.
This was generated using ATLAS Version 2.12.1 as well as WebAPI Version 2.12.1.

C. References

[1] RCall.jl.
[2] OHDSI. *The Book of OHDSI: Observational Health Data Sciences and Informatics*. OHDSI.
[3] Christopher Rowley. PythonCall.jl: Python and Julia in harmony.
[4] Martijn Schuemie, Frank DeFalco, and Vojtech Huser. Eunomia.
[5] Kirill Simonov, Clark C. Evans, and Jacob S. Zelko. FunSQL : Julia library for compositional construction of SQL queries. doi:10.5281/zenodo.7705325.
[6] U.S. Food and Drug Administration. Real-world evidence.
[7] Jacob Zelko, Malina Hy, Varshini Chinta, Emily Liau, and Morgan Knowlton. A pilot characterization study assessing health equity in mental healthcare delivery within the state of georgia.
[8] Jacob S. Zelko, Sarah Gasman, Shenita R. Freeman, Dong Yun Lee, Jaan Altosaar, Azza Shoaibi, and Gowtham Rao. Developing a Robust Computable Phenotype Definition Workflow to Describe Health and Disease in Observational Health Research. doi:10.48550/arXiv.2304.06504. arxiv:2304.06504 [cs].