5]Jacob S. Zelko 1, 2]Malina Hy 1, 2]Varshini Chinta 3]Fareeda Abdelazeez 4]Jay Sanjay [1]Georgia Tech Research Institute [2]Georgia Institute of Technology [3]Zewail University [4]Indian Institute of Technology, Hyderabad [5]Northeastern University

juliacon

The JuliaHealth Observational Health Research Subecosystem

[

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

Observational health research is the study and analysis of historical real world data to explore various cohorts found within such datasets to generate real world evidence to support applications within health-related domains. Research within this space can investigate questions such as population health trends for a particular disease, health disparities within patient populations, and cost costs of care related to particular treatment procedures. Due to the untapped potential of this space, over the past 20 years, there have been several developments within the observational health research community to enable more effective research to be conducted within this field (such as the creation of interoperability standards and tool creation [**?**]).

Arguably, the most crucial innovation that has emerged in observational health research within the last decade is the development and adoption of the Observational Medical Outcomes Partnership Common Data Model (OMOP CDM) which was designed specifically to standardize real world health data. [**?**] 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 real world data have been able to successfully transform their datasets to the OMOP CDM. This has led to exciting new venues to enable successful collaboration relationships [**?**] across countries and institutions.

Data standards within observational health research. Although other standards exist within observational health, the OMOP CDM will be the particular standard focused on within this paper. To give further background on the OMOP CDM, OMOP was convened in 2009 by the US Food and Drug Administration to address concerns related to data type, study design, and data privacy concerns of utilizing "Real World Data". [**?**] [**?**] Their collective efforts resulted in the OMOP Common Data Model (OMOP CDM) to house and ingest real world data that is used in observational health research. The OMOP CDM is a "person-centered" model (i.e. any data found in a database formatted to the OMOP CDM can be traced back to an individual patient) that standardizes patient events to a plethora of database tables to represent the unique dynamics and structures found in individual patient events (for more details on these relationships, please refer to **??**). These events would be any recorded interaction a patient has with a healthcare system and they can consist of situations such as a patient visiting an inpatient emergency room, a drug prescription, a patient condition diagnosis, medical procedures, and more.

Collaborative networks and studies using observational health data. 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 OHDSI comprehensive of relevant documentation, analysis code, results, and study protocols describing the study's scope and objectives. 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" [**?**] [**?**] OHDSI was developed to focus on leveraging observational health data stored within the OMOP CDM. 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. 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

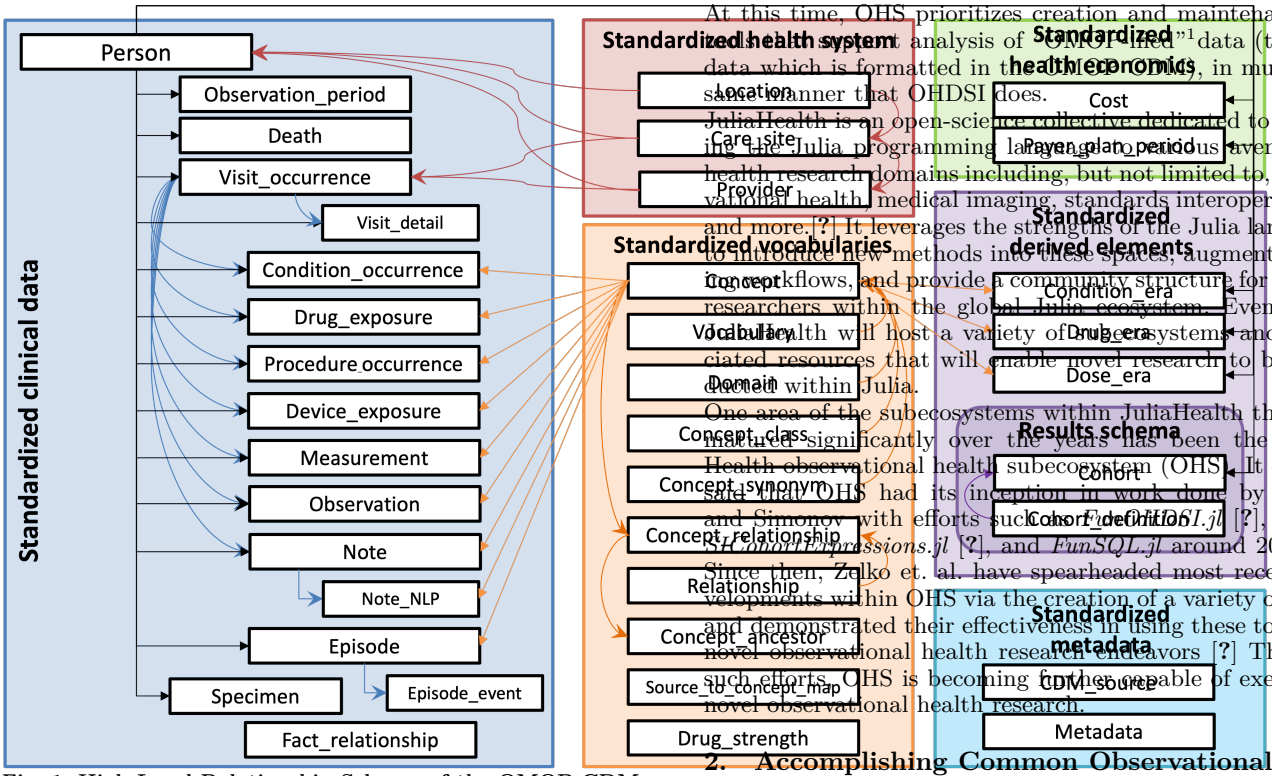

Fig. 1: High Level Relationship Schema of the OMOP CDM Version 5.4. This is the schema that is prescribed by the OMOP CDM. It should be noted OMOP CDM itself does not require a specific technology to work with the data stored in this standard.

execute the study package on their unique OMOP CDM formatted dataset. 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. 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. [**?**]

Research software to support the analysis of observational health data. As the OMOP CDM has become ubiquitous within observational health research, developing research software to handle and manage OMOP CDM data is necessary to meaningfully contribute within the observational health research space. 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.

1.1 JuliaHealth and Observational Health Research

As JuliaHealth is an ad hoc network of contributors, varying aspects of the ecosystem have developed at differing rates as a function of factors such as developer time or workplace incentives.

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

JuliaHealth is an open-science collective dedicated to bringing the Julia programming language to various avenues of health research domains including, but not limited to, observational health, medical imaging, standards interoperability and more.[**?**] It leverages the strengths of the Julia language to introduce new methods into these spaces, augment existing workflows, and provide a community structure for health researchers within the global Julia ecosystem. Eventually, JuliaHealth will host a variety of subecosystems and associated resources that will enable novel research to be conducted within Julia.

One area of the subecosystems within JuliaHealth that has matured significantly over the years has been the JuliaHealth observational health subecosystem (OHS). It can be said that OHS had its inception in work done by Evans and Simonov with efforts such as *CohortDefinitionAPI.jl* [**?**], *OHDSICohortExpressions.jl* [**?**], and *FunSQL.jl* around 2021 [**?**] 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 [**?**] Through such efforts, OHS is becoming further capable of executing novel observational health research.

2. Accomplishing Common Observational Health Research Tasks within Julia

JuliaHealth has capacity through the OHS to perform observational health workflows via a variety of tools designed to support and enable novel investigations. This paper summarizes and highlights some of the current capacity OHS provides such as:

(1) **Using Phenotype Definitions** - brief overview of phenotype definitions and how to work with them
(2) **Rapid Cohort Iteration and Creation** - how to iteratively develop patient cohorts to explore
(3) **Working with Patient Databases** - approaches to working with different patient databases
(4) **Ecosystem Interoperability** - extending JuliaHealth's ecosystem to other research ecosystems

2.1 Using Phenotype Definitions

One of the premier tools within the OHDSI HADES ecosystem is the ATLAS Shiny application which was developed to develop patient phenotype definitions within observational health research contexts (amongst other use cases). These phenotype definitions have varying levels of complexity where they can be very simple (such as 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 va-

[1] It is unclear where terms like "OMOP-ify" or "OMOP-ification" (the process of converting a data to the OMOP CDM) originated; it may have been coined by Dr. Jon Duke.

riety of serialization formats such as JSON schemas or prepared SQL expressions available in multiple SQL flavors.[2]
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 serialized computable phenotype definitions.
(2) Reinterpret serialization into the FunSQL.jl Domain Specific Language.
(3) Return a SQL statement for an OMOP CDM database to generate a given cohort.

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. [**?**] 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 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.
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.

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 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, 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. DBConnector.jl was inspired by the package *DatabaseConnector* (Link: https://github.com/OHDSI/DatabaseConnector)
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.

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. [**?**] [**?**] 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.

[2] Although the JSON schema is not made formally available within OHDSI, a community made copy of the schema can be found here: [**?**]

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

Using the JuliaHealth tools developed, we enumerate an example observational health study workflow that would be used in practice.[3]
For this paper, we used two datasets:

(1) **Eunomia** - this is a synthetic patient dataset created from the open source synthetic patient generator, *Synthea*, and hosted within the HADES and JuliaHealth ecosystems. [**?**]
(2) **MIMIC III** - a deidentified datasets consisting of inpatient visits from Beth Israel Deaconness Hospital in Boston, Massachussetts. The particular version of the dataset that we used was transformed into the OMOP CDM using tools from MIT LCP and PhysioNet. [**?**, **?**, **?**, **?**]

Reproducing the workflow using the *Eunomia* dataset is made readily available at the repository here: [**?**]
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

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

occ.GenerateDatabaseDetails(
    :sqlite,
    "main"
)

occ.GenerateTables(conn)
```

Listing 1: Configuring OMOP CDM Database Connection. Using *DBConnector.jl*, connection to the *Eunomia* SQLite database can be made. Then, *OMOPCDMCohortCreator.jl* (occ) is used to generate connection details used internally by the package for the rest of the session.

[3]The code snippets presented in the body of the paper may omit certain intermediate steps. Please review the associated code repository for a full working tutorial located here: https://github.com/TheCedarPrince/2022-JuliaCon-Proceedings-Submission. As a further note, the code to fully reproduce a sample analysis with *Eunomia* is available within the code repository. As MIMIC III is not publicly available, a sample analysis using MIMIC III is available with some information stripped for privacy considerations.

3.2 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.

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

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

Listing 2: Creating a Cohort Using a Computable Phenotype Definition. *OHDSICohortExpressions.jl* (oce) ingests a computable phenotype definition to generate a cohort of patients. The JSON is translated into an intermediate SQL Syntax Tree, instantiated in SQL (SQLite) targeting an OMOP CDM v5.3.1 database, and then populates the database's cohort table.

3.3 Characterizing Patient Populations

```
patients = occ.GetCohortSubjects(1, conn)

patients_race = occ.GetPatientRace(C.person_id, conn)
patients_gender = occ.GetPatientGender(C.person_id, conn)

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

Listing 3: Find Demographic Characteristics of Cohort. Demographic properties of a cohort can be quickly queried from a database.

3.4 Calculating Crude Prevalence Rate

```
strep_df = df.combine(patient_groups, df.nrow ⇒ :count)

# Creator code to process data and generate counts...

strep_df.prev = strep_df.count ./ strep_df.total_count
```

Listing 4: Calculating crude prevalence rates. Once demographic information is extracted, processing can be performed using tools such as *DataFrames.jl* (df) to calculate metrics such as prevalence rates in a very straightforward manner.

Using this analytical approach, tabulation can be quickly conducted across patients.

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

Table 1. : Example of a Strep Throat Prevalence Calculation. This illustrates the sort of results one can quickly generate using the OHS. Note, that although the prevalence values in this table are based on synthetic data and are not realistic, the output looks very similar to what one could expect to generate.

3.5 Summary of Results

As seen in **??**, 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

(2) Distributed Computing

(3) Automatic SQL Generation

4.1.1 Modularization. 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. 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. 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:
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:
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:
In this way, future researchers using tools such as OMOPCDMCohortCreator.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.

```
FGetPatientRace = Fix2(occ.GetPatientRace, conn)
FGetPatientGender = Fix2(occ.GetPatientGender, conn)
FGetPatientAgeGroup = Fix2(occ.GetPatientAgeGroup, conn)

CofactorPatients(patients) = @chain patients begin
    FGetPatientRace
    FGetPatientGender
    FGetPatientAgeGroup
end

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

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

Listing 5: Modularizing Study Code. Using Julia's partial application abilities, connection objects can be fixed allowing study code to be readily modularized. Using the package, Chain.jl, occ and processing functions can be composed together to create modular study code.

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
```

Listing 6: Distributed Computation of a Study. With occ, study code can be easily abstracted and modularized. Using Julia's native Distributed package, each module of a study can then be parallelized over patient cohorts.

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 Automatic SQL Generation. 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.

```
# Julia Input
occ.GetPatientAgeGroup([1]; age_groupings = [
                                              [0, 40],
                                              [41, 80]
                                            ])

# SQL Output
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"
```

Listing 7: Producing SQL from Julia Expression. When occ functions are not passed a connection object, they can produce SQL representing the underyling query the Julia expression is executing.

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

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.

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.

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) The JuliaHealth community and contributors for fearlessly trying to push boundaries on what observational health informatics can truly be

(4) Dilum Aluthge for helping this work find a home in JuliaHealth

(5) Adam Black for thinking through potential approaches

(6) Clark Evans and Kirill Simonov for being the pioneers of this work in Julia
(7) The Roux Institute's OHDSI Center for their encouragement

7. References

[1] RCall.jl.
[2] Dilum Aluthge. Announcing the JuliaHealth Organization.
[3] Clark C. Evans and Kirill Simonov. FunOHDSI.jl.
[4] Clark C. Evans and Kirill Simonov. OHDSICohortExpressions.jl.
[5] Ary L Goldberger, Luis AN Amaral, Leon Glass, Jeffrey M Hausdorff, Plamen Ch Ivanov, Roger G Mark, Joseph E Mietus, George B Moody, Chung-Kang Peng, and H Eugene Stanley. PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals. 101(23):e215–e220.
[6] George Hripcsak, Patrick B. Ryan, Jon D. Duke, Nigam H. Shah, Rae Woong Park, Vojtech Huser, Marc A. Suchard, Martijn J. Schuemie, Frank J. DeFalco, Adler Perotte, Juan M. Banda, Christian G. Reich, Lisa M. Schilling, Michael E. Matheny, Daniella Meeker, Nicole Pratt, and David Madigan. Characterizing treatment pathways at scale using the OHDSI network. 113(27):7329–7336. doi:10.1073/pnas.1510502113.
[7] Alistair E W Johnson, Tom J Pollard, Lu Shen, Li-Wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. MIMIC-III, a freely accessible critical care database. 3(1):1–9.
[8] Alistair EW Johnson, David J Stone, Leo A Celi, and Tom J Pollard. The MIMIC Code Repository: Enabling reproducibility in critical care research. 25(1):32–39.
[9] Brad Kelechava. The SQL Standard - ISO/IEC 9075:2023 (ANSI X3.135).
[10] OHDSI. *The Book of OHDSI: Observational Health Data Sciences and Informatics*. OHDSI.
[11] J Marc Overhage, Patrick B Ryan, Christian G Reich, Abraham G Hartzema, and Paul E Stang. Validation of a common data model for active safety surveillance research. 19(1):54–60.
[12] Christopher Rowley. PythonCall.jl: Python and Julia in harmony.
[13] Craig Sachson. *Our Journey*.
[14] Martijn Schuemie, Frank DeFalco, and Vojtech Huser. Eunomia.
[15] Kirill Simonov, Clark C. Evans, and Jacob S. Zelko. FunSQL : Julia library for compositional construction of SQL queries. doi:10.5281/zenodo.7705325.
[16] U.S. Food and Drug Administration. Real-world evidence.
[17] 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.
[18] Jacob S. Zelko. Julia in health and medicine. In *JuliaCon 2023*. Birds of a Feather.
[19] 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].