Building Bridges with Julia

Using OHDSI R Packages in Julia

PRESENTER: Jacob S. Zelko

INTRODUCTION:

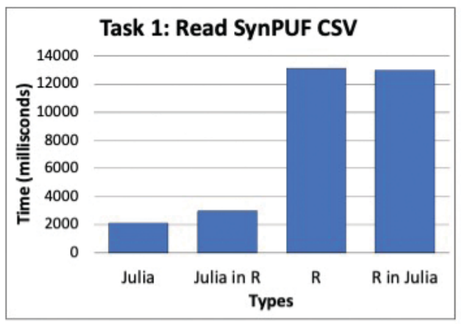
The amount of healthcare data is skyrocketing. Waiting on answers from this data just one day, can mean thousands of lives lost – as seen in the COVID pandemic. Beyond the cost of human lives, the financial costs of needed hardware for delivering these crucial answers are rising. Between the issues of financial costs and the urgency for rapid insights, there is clear need in the OHDSI community to encourage tooling that can bridge this problem.

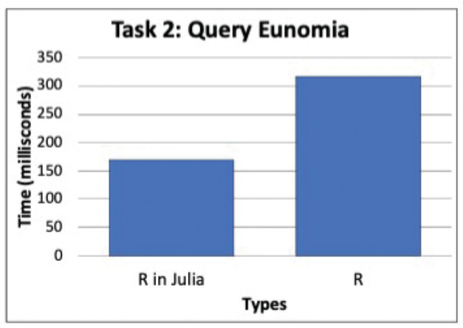
METHODS:

Demonstrated here is a possible solution, using a dynamic and high performance programming language, Julia[1], to interoperate with R to utilize OHDSI packages and perform basic procedures easily to produce performance gains. For this approach, I used the R package JuliaConnectoR[2] to use Julia tools from within R and the Julia package RCall[3] to call R tools from within Julia. Two basic exercises for benchmarking were done:

- Task 1: Read SynPUF[4] CSV
- Task 2: Query Eunomia[5]

RESULTS:





For Task 1, a ≈5x's speedup is seen from the Julia in R example over R's CSV reader.

For Task 2, only using R and R embedded in Julia was considered. Leveraging Julia, the R in Julia example is ≈2x's fast as the R implementation.

Task 1 Code: Reading SynPUF CSV

Figure A: Julia

```
# Using Julia's CSV reader
using CSV

# Reading in SynPUF data
data = CSV.File("synpuf.csv")
```

Figure B: R

```
# Reading raw SynPUF data
data <- read.csv("synpuf.csv")
```

Figure C: R in Julia

```
# Load RCall package
using RCall

# Read SynPUF data using R
data = R"read.csv(\"synpuf.csv\")"
```

Figure D: Julia in R

```
# Load the JuliaConnectoR package
library("JuliaConnectoR")

# Import Julia's CSV reader
jcsv <- juliaImport("CSV")

# Read in example SynPUF data
data <- jcsv$File("synpuf.csv")
```

Julia in R (Fig 1D), can provide a nearly 5x's speedup over the R CSV reader (Fig 1B).

Task 2 Code: Query Eunomia

Figure A: Querying in R

```
# Open connection to Eunomia
library('DatabaseConnector')
connectionDetails <- Eunomia::getEunomiaConnectionDetails()
connection <- connect(connectionDetails)

# Create SQL Query
sql <-   "
SELECT *
FROM @cdm.person
"

# Return people from SQL query
result <- renderTranslateQuerySql(connection, sql, cdm ="main")

# Make R data frame
data.frame(t(sapply(result,c)))
```

Figure B: Querying in Julia Using R

```
# Load RCall package
using RCall

# Get patients from Eunomia PERSON table
people = R"""

library('DatabaseConnector')
connectionDetails <- Eunomia::getEunomiaConnectionDetails()
connection <- connect(connectionDetails)

sql <-   "
SELECT *
FROM @cdm.person
"
result <- renderTranslateQuerySql(connection, sql, cdm ="main")

"""

# Convert R 'list' to Julia DataFrame
people_data = rcopy(people)
```

Querying OHDSI's Eunomia package in Julia using R gives a 2x's speedup over the base R implementation

For more information, scan the QR code here!

Georgia Tech Research Institute

Problem. Solved.

SELECTED DISCUSSION TOPICS:

How were benchmarks made?

Table 1: Task Benchmarks

Task	Type	Time (ms)
Read SynPUF CSV	Julia	2114
Read SynPUF CSV	Julia in R	2960
Read SynPUF CSV	R	13100
Read SynPUF CSV	R in Julia	12980
Query Eunomia	R in Julia	169
Query Eunomia	R	317

In Table 1, the minimum time from 10 evaluations of the code created for each task was recorded. For the "Read SynPUF CSV" task, 150 MBs of SynPUF data were read. The benchmarking tools, BenchmarkTools.jl[6] for Julia and bench[7] for R was used to generate times.

Why Julia instead of language X?

- Interoperability with other languages
- High performance computing
- Understandable syntax.
- Emerging resources for OHDSI tasks (e.g. database interfaces, OMOP CDM, etc.).

What are future research directions?

Further directions for this research will be to

- Leverage existing OHDSI tools with Julia
- Identify improvements with Julia via language interoperability (i.e. R & Julia)
- Develop tooling for actual study to test how feasible it is to leverage Julia in an OHDSI network study design.

ACKNOWLEDGEMENTS

Thank you so much to the following people for their support in this endeavor!

- Charity Hilton and Jon Duke (Georgia Tech Research Institute).
- Dilum Aluthge and Clark C. Evans (JuliaHealth)
- Kristin Kostka (OHDSI Community).

REFERENCES:

1. Bezanson, Jeff, Alan Edelman, Stefan Karpinski, and Viral B Shah. 2017. "Julia: A Fresh Approach to Numerical Computing." *SIAM review* 59(1): 65–98.
2. Lenz, Stefan, Maren Hackenberg, and Harald Binder. 2021. "The JuliaConnectoR: A Functionally Oriented Interface for Integrating Julia in R." *arXiv:2005.06334 [cs, stat]*. http://arxiv.org/abs/2005.06334 (August 13, 2021).
3. RCall.Jl. 2021. Julia Interop. Julia. https://github.com/JuliaInterop/RCall.jl (August 13, 2021).
4. "Medicare Claims Synthetic Public Use Files (SynPUFs) | CMS." https://www.cms.gov/Research-Statistics-Data-and-Systems/Downloadable-Public-Use-Files/SynPUFs (August 13, 2021).
5. *Eunomia*. 2021. Observational Health Data Sciences and Informatics. R. https://github.com/OHDSI/Eunomia (August 13, 2021). Environments." *arXiv e-prints*.
6. Chen, Jiahao, and Jarrett Revels. 2016. "Robust Benchmarking in Noisy
7. High Precision Timing of R Expressions • Bench." https://bench.r-lib.org/ (August 13, 2021).

OHDSI