dbverse: composable database libraries for larger-than-memory scientific analytics

Edward C. Ruiz

ecruiz@bu.edu

Ph.D. Candidate, Dries Lab, Boston University

August 15, 2024

Link to slides: https://rpubs.com/Ed2uiz/dbverse-duckCon2024

Motivation

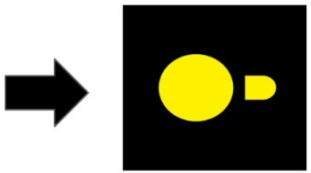
Current challenges with scientific data analysis

- Scientific data is often messy and complex.
  - → Not your standard dataframe
  - → Heterogeneous, multi-modal (e.g. “multi-omics”)
  - → Larger-than-memory (e.g. spatial multi-omics)
- A variety of tools and languages are used to analyze scientific data.
  - → Interoperability is often limited
  - → Fragmentation by data type
- **How can we develop better approaches for scientific data analysis?**

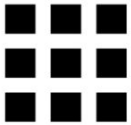
dbverse overview

Scientific Data

- Matrices
- Spatial geometries
- Genomic Sequences



{dbMatrix}



- .csv, .mtx
- Sparse Matrices (i.e. dgCMatrix)
- Dense matrices
- Matrix statistics, arithmetic

{dbSpatial}

- Points, polygons
- 50+ geospatial file formats (.shp, .geojson, more)
- 60+ ST_*() functions
- {terra}, {sf} interoperability

{dbSequence}

- 10+ sequence file formats (.bam, .fastq, .vcf, more)

Link to slides: https://rpubs.com/Ed2uiz/dbverse-duckCon2024

How does it work? *dbMatrix* example

dbMatrix adopts familiar `{Matrix}` syntax...

```
scaled <- dbMatrix[,"cell_1"] * 10
```

with underlying methods implemented with `{dplyr}` ...

```
scaled <- dplyr::tbl(con, "dbMatrix") |>
  dplyr::select(cell_id = "cell_1") |>
  dplyr::mutate(scaled = expression * 10)
```

How does it work? *dbMatrix* example

dbMatrix adopts familiar `{Matrix}` syntax...

```
scaled <- dbMatrix[,"cell_1"] * 10
```

with underlying methods implemented with `{dplyr}` ...

```
scaled <- dplyr::tbl(con, "dbMatrix") |>
  dplyr::select(cell_id = "cell_1") |>
  dplyr::mutate(scaled = expression * 10)
```

which are transpiled to SQL via `{dbplyr}`...

```
SELECT cell_id, expression * 10 AS scaled
FROM my_cells.db WEHRE cell_id = 'cell_1';
```

How does it work? *dbMatrix* example

dbMatrix adopts familiar `{Matrix}` syntax…

```
scaled <- dbMatrix[,"cell_1"] * 10
```

with underlying methods implemented with `{dplyr}` …

```
scaled <- dplyr::tbl(con, "dbMatrix") |>
  dplyr::select(cell_id = "cell_1") |>
  dplyr::mutate(scaled = expression * 10)
```

which are transpiled to SQL via `{dbplyr}`…

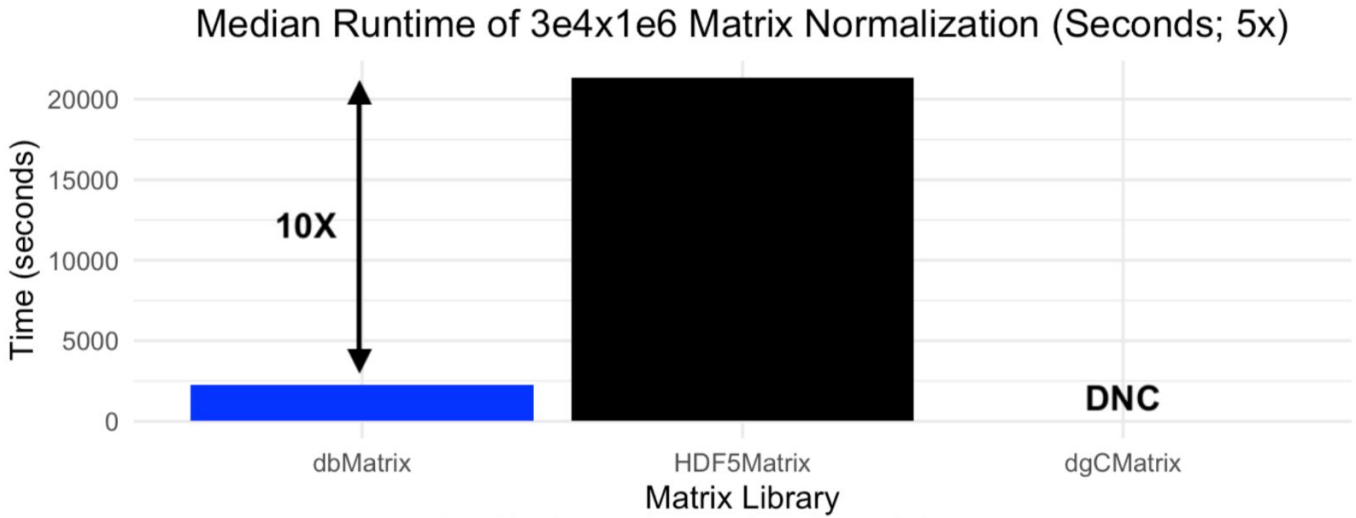
```
SELECT cell_id, expression * 10 AS scaled
FROM my_cells.db WEHRE cell_id = 'cell_1';
```

and lazy evaluated in a DuckDB database 🐥🚀!

Illustrative {dbMatrix} benchmark

{dbMatrix} performs larger-than-memory sparse matrix operations and outperforms HDF5Matrix

```
norm_mat <- t(t(dbMatrix) / libsizes) * scalefactor
lib_norm_mat <- log(norm_mat + offset) / log(base)
log_norm_mat <- t(norm_mat) - colMeans(lib_norm_mat)
# ...additional matrix operations
```



Link to slides: https://rpubs.com/Ed2uiz/dbverse-duckCon2024

Illustrative {dbSpatial} benchmark

{dbSptaial} outperforms existing in-memory methods for spatial intersections

Task: find the intersection between cell polygons in tissue region of interests (ROIs)

Median Runtime (seconds; 5X queries)

	No. Polygons	dbSpatial	sf (memory)	Δ Performance
ROI 1	1564	0.05	0.480	9X
ROI 2	92498	1.56	39.024	25X
ROI 3	143245	2.55	65.398	25X

Illustrative {dbSequence} benchmark

{dbSequence} outperforms competing methods

Task: filter reads in a genomic `*.bam` file (28GB, 285e6 reads)

- `samtools` v1.20 (Li *et al.* 2009)
- q01: chromosome region
- q02: q01 + flag
- q03: q02 + CIGAR string (`samtools` + `awk`)

Median Runtime (seconds; 5X queries)

Query	dbSequence	samtools	Δ Speed
q01	0.03400	0.08	2X
q02	0.00622	0.02	3X
q03	19.26000	DNC	∞

Link to slides: https://rpubs.com/Ed2uiz/dbverse-duckCon2024

Conclusions

Advantages of using DuckDB for scientific data analysis

- **Runs on modern laptops**: All previous benchmarks were performed on a Macbook Pro M2, 16GB RAM, 512GB SSD
- **Open Source**: MIT license
- **Platform-independent**: Runs on all major OS
- **Portable**: Share results in a single `*.db` file
- **Affordable**: Free to use, pay for more local storage as needed or 'hybrid execution' with *MotherDuck*
- **First release**: `08/15/2024` (today 🎉)

Limitations and future directions

- `dbverse` is currently only compatible with R
  - → Plan to support other languages (e.g. Python)
- Limited visualization/plotting functionality
  - → `uwdata/mosaic` integration, see discussion #354
- Limited support for large images
  - → DuckDB Spatial Extension Raster support
- Plans to adopt `duckplyr`
  - → See `duckdblabs/duckplyr` issue #86
- … and much more!

Acknowledgements

Ruben Dries Lab

- **Jiaji George Chen**
- Iqra Amin
- Wonyl Choi
- Junxiang Xu
- Yibing Michelle Wei
- Jeffrey Sheridan
- Quynh Sun
- Veronica Jarzabek

Funding

BU Rafik B. Hariri Institute for Computing and Computational Science & Engineering

CZI

GEM THE NATIONAL GEM CONSORTIUM

CENTER BOSTON MEDICAL EXCEPTIONAL CARE. WITHOUT EXCEPTION.

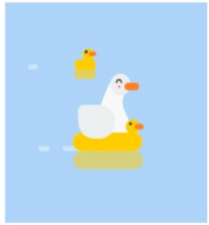
Hematology and Oncology
Carter Trainee Fund

Link to slides: https://rpubs.com/Ed2uiz/dbverse-duckCon2024

Questions?

To learn more please visit:
https://drieslab.github.io/dbverse/

🦤👨🏽‍💻@Ed2uiz | ✉️ ecruiz@bu.edu



Link to slides: https://rpubs.com/Ed2uiz/dbverse-duckCon2024