Benchmarking Log Normalization of Sparse Matrix

AUTHOR
Ed Ruiz

PUBLISHED
August 21, 2023

1 Description

This is a draft test script to showcase speed for performing a matrix operation (library normalization) on different formats of a sparse gene expression matrix with three different number of cells (1M, 10M, and 100M). This data was generated from binned data of a mouse embryo at embryonic day 16.5 (Stereo-seq data).

See sample processing file for data preprocessing steps.

2 Load deps

```
library(duckdb) # version 0.8.1.1
library(dbplyr)
library(dplyr)
# TODO: replace above with just below package load
# library(Duckling)
library(magrittr)
library(data.table)
library(microbenchmark)
library(DelayedArray)
```

```
Warning: package 'DelayedArray' was built under R version 4.3.1

Warning: package 'MatrixGenerics' was built under R version 4.3.1

Warning: package 'IRanges' was built under R version 4.3.1

Warning: package 'S4Arrays' was built under R version 4.3.1
```

```
library(BPCells)
reloadGiotto() #library(Giotto)
library(Seurat) # remotes::install_github("satijalab/seurat", "seurat5", quiet = TRUE)
library(SeuratObject) # remotes::install_github("mojaveazure/seurat-object", "seurat5", q
# library(SeuratDisk)
options(Seurat.object.assay.version = "v5")
```

3 Setup data

```
setwd('~/Downloads/duckdb-benchmarking/')

# Load saved database
db_filename_1M = "e16_e2s6-1M-sparseijx-duckdbv8-1-1.db"
db_filename_10M = "e16_e2s6-10M-sparseijx-duckdbv8-1-1.db"
db_filename_100M = "e16_e2s6-100M-sparseijx-duckdbv8-1-1.db"
# TODO: replace above with duckling implementation

# Load input test files #####
con_1M = dbConnect(duckdb(), dbdir = db_filename_1M)
con_10M = dbConnect(duckdb(), dbdir = db_filename_10M)
con_100M = dbConnect(duckdb(), dbdir = db_filename_100M)
```

4 Load prepared data

```
# Load saved dgc data
# See sample processing file for preprocessing steps
file_name_1M = "~/Downloads/duckdb-benchmarking/E16.5_E2S6_1M.RDS"
file_name_10M = "~/Downloads/duckdb-benchmarking/E16.5_E2S6_10M.RDS"
file_name_100M = "~/Downloads/duckdb-benchmarking/E16.5_E2S6_100M.RDS"

gxp_1M = readRDS(file_name_1M)
gxp_10M = readRDS(file_name_10M)
gxp_100M = readRDS(file_name_100M)

bpcells_1M = open_matrix_dir("~/Downloads/duckdb-benchmarking/seurat_bpcells_1M")
bpcells_10M = open_matrix_dir("~/Downloads/duckdb-benchmarking/seurat_bpcells_10M")
bpcells_100M = open_matrix_dir("~/Downloads/duckdb-benchmarking/seurat_bpcells_100M")

# create dt test inputs
ijx_1M = setDT(Matrix::summary(gxp_1M))
ijx_10M = setDT(Matrix::summary(gxp_10M))
ijx_100M = setDT(Matrix::summary(gxp_100M))

# create delayedMatrix test inputs
# TODO: confirm this setup, need to write to hdf5 first?
delayed_matrix_1M <- DelayedArray(gxp_1M)
delayed_matrix_10M <- DelayedArray(gxp_10M)
delayed_matrix_100M <- DelayedArray(gxp_100M)

# Create seurat objs with BPCells
seurat_1M = CreateSeuratObject(counts = bpcells_1M)
seurat_10M = CreateSeuratObject(counts = bpcells_10M)
seurat_100M = CreateSeuratObject(counts = bpcells_100M)
```

5 Benchmark functions

```
scalefactor=1001

runLibraryNormTestDT <- function(ijx){
   #  test_ijx = copy(ijx)
    ijx[, x := (as.numeric(x) / sum(x, na.rm = TRUE)) * scalefactor, by = j]
    return(ijx)
}

runLibraryNormTestDB <- function(con){
    res <- tbl(con, "ijx") %>%
        dplyr::group_by(j) %>%
        dplyr::mutate(x = (as.numeric(x) / sum(x, na.rm = TRUE)) * scalefactor) %>%
        dplyr::ungroup() %>%
        dplyr::collapse() %>%
        dplyr::collect()

    return(res)
}

runLibraryNormTestDB_sql <- function(con){
    sql_query = paste0(
                'SELECT a.i, a.j, a.x/b.sum_x * ', scalefactor,' AS x
                FROM (
                    SELECT *
                    FROM (\n',
                dbplyr::sql_render(con = con, tbl(con, 'ijx')),
                '\n)
                ) a
                INNER JOIN (
                    SELECT j, SUM(x) AS sum_x
                    FROM (\n',
                dbplyr::sql_render(con = con, tbl(con, 'ijx')),
                '\n) GROUP BY j
                ) b
                ON a.j = b.j'
                )

    res = dplyr::tbl(src = con, dbplyr::sql(sql_query)) %>% dplyr::collect()

    return(res)
}
```

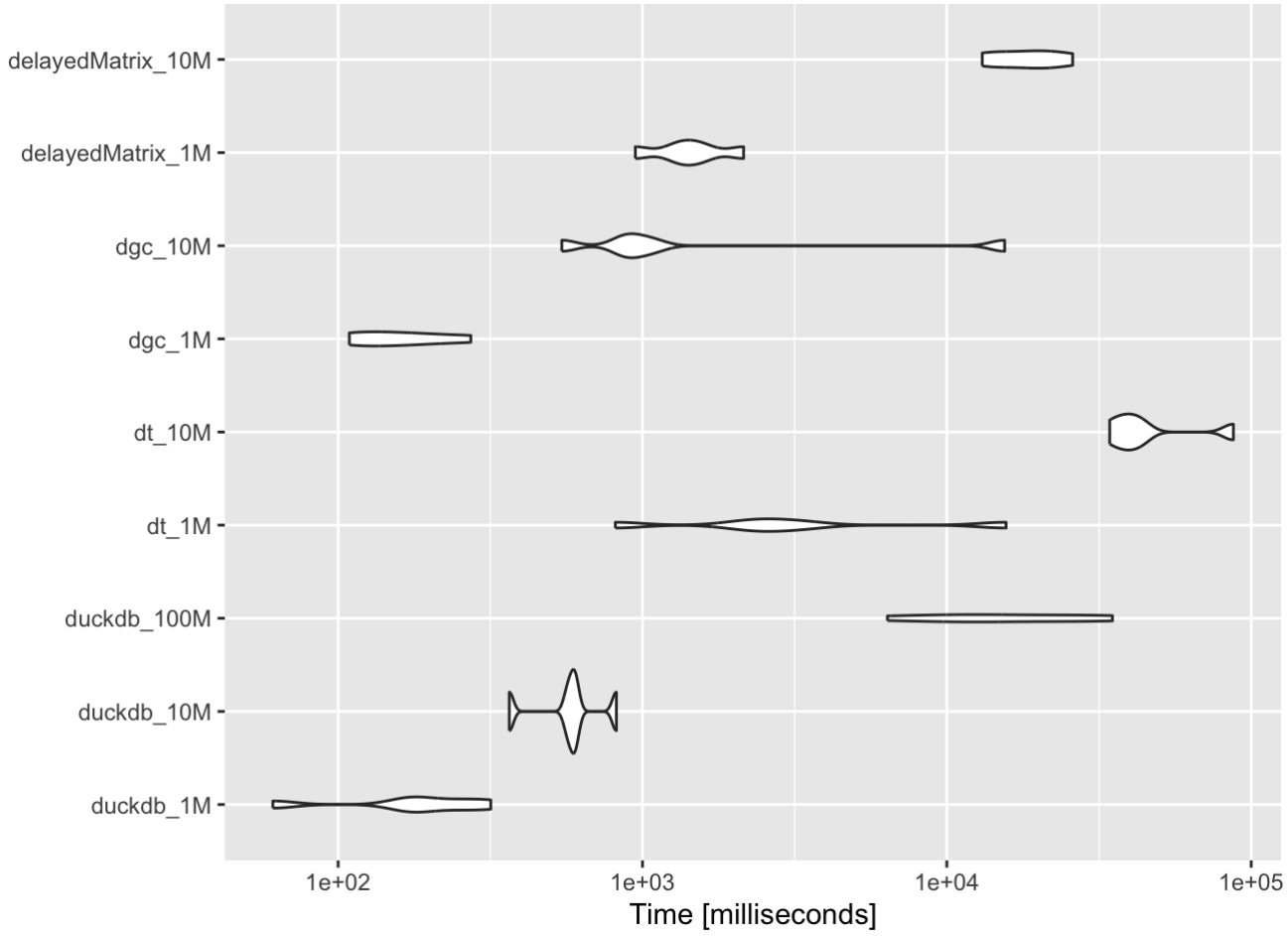
6 Run benchmark

```
res = microbenchmark::microbenchmark(
    duckdb_1M = runLibraryNormTestDB(con_1M),
    duckdb_10M = runLibraryNormTestDB(con_10M),
```

```
    duckdb_100M = runLibraryNormTestDB(con_100M),
    dt_1M = runLibraryNormTestDT(ijx_1M),
    dt_10M = runLibraryNormTestDT(ijx_10M),
    # dt_100M = runLibraryNormTestDT(ijx_100M), # does not run on MBP M2, 16GB RAM
    dgc_1M = Giotto:::libNorm_giotto(mymatrix = gxp_1M, scalefactor = scalefactor),
    dgc_10M = Giotto:::libNorm_giotto(mymatrix = gxp_10M, scalefactor = scalefactor),
    # dgc_100M = Giotto:::libNorm_giotto(mymatrix = gxp_100M, scalefactor = scalefactor),
    delayedMatrix_1M = Giotto:::libNorm_giotto(mymatrix = delayed_matrix_1M, scalefactor
    delayedMatrix_10M = Giotto:::libNorm_giotto(mymatrix = delayed_matrix_10M, scalefacto
    # delayedMatrix_100M = Giotto:::libNorm_giotto(mymatrix = delayed_matrix_100M, scalef
    # bpcells_1M = NormalizeData(seurat_1M, normalization.method = "LogNormalize"),
    # bpcells_10M = NormalizeData(seurat_10M, normalization.method = "LogNormalize"),
    # bpcells_100M = NormalizeData(seurat_100M, normalization.method = "LogNormalize"),
    times = 5
)
```

```
Warning in microbenchmark::microbenchmark(duckdb_1M =
runLibraryNormTestDB(con_1M), : less accurate nanosecond times to avoid
potential integer overflows
```

```
# View res
ggplot2::autoplot(res)
```



```r
# Close db connections
dbDisconnect(con_1M)
dbDisconnect(con_10M)
dbDisconnect(con_100M)
```