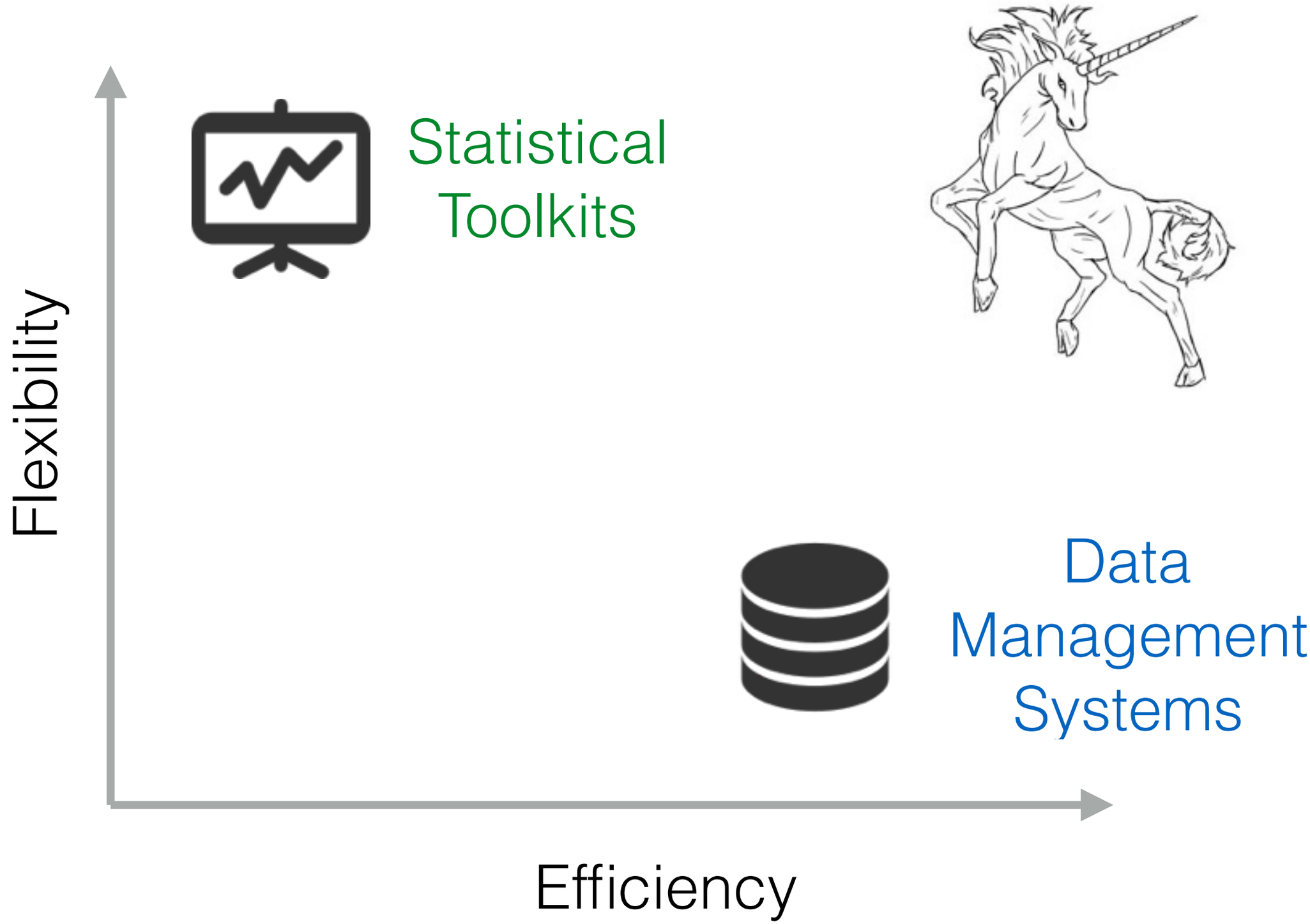
CWI
Centrum Wiskunde & Informatica

MonetDBLite – Bringing Column Stores to the Masses

Hannes Mühleisen*

DBGDBD 2015-12-16

Integrate not Reinvent



Running a DB is hard*

- Installation / Automatic startup difficult
- Configuration for workload often crucial
  - `checkpoint_completion_target`?
  - `effective_cache_size`?
- Maintenance, updates, … workload increase

⟶ Most people don’t bother if not forced!

What about  ?

- In-process SQL database, data either in memory or in a file, rock-solid, used on every smartphone, browser, OS, ….
- People also use it for large-ish dataset analysis
- Bad idea, SQLite was never built for this
  - e.g. row-based storage model

Enter MonetDBLite

MonetDBLite for R

Released Nov. 2015

What is MonetDBLite

- Embedded & streamlined MonetDB
  - No installation
  - In-Process operation
  - Query results are pointers to C arrays
  - Data append from C arrays
- Wrappers for R, Python*, Java*, …

Engineering Challenges

- MonetDB was never designed to run embedded
  - Fatal errors `exit()`
  - Relative paths relying on `setwd()`
  - Symbol clashes `error()` etc.
  - Global variables galore (restartability?)
  - `stdout`/`stderr` used for error reporting
- Build/runtime dependencies (esp. Windows)
- R's windows compilation toolchain

Installation

- Installs like any other R package

```
install.packages(c("MonetDB.R", "MonetDBLite"),
    repos=c("http://dev.monetdb.org/Assets/R/",
            "http://cran.rstudio.com/"))
```

- Linux: Source install
- Windows/OSX: Binary packages

DBI Usage

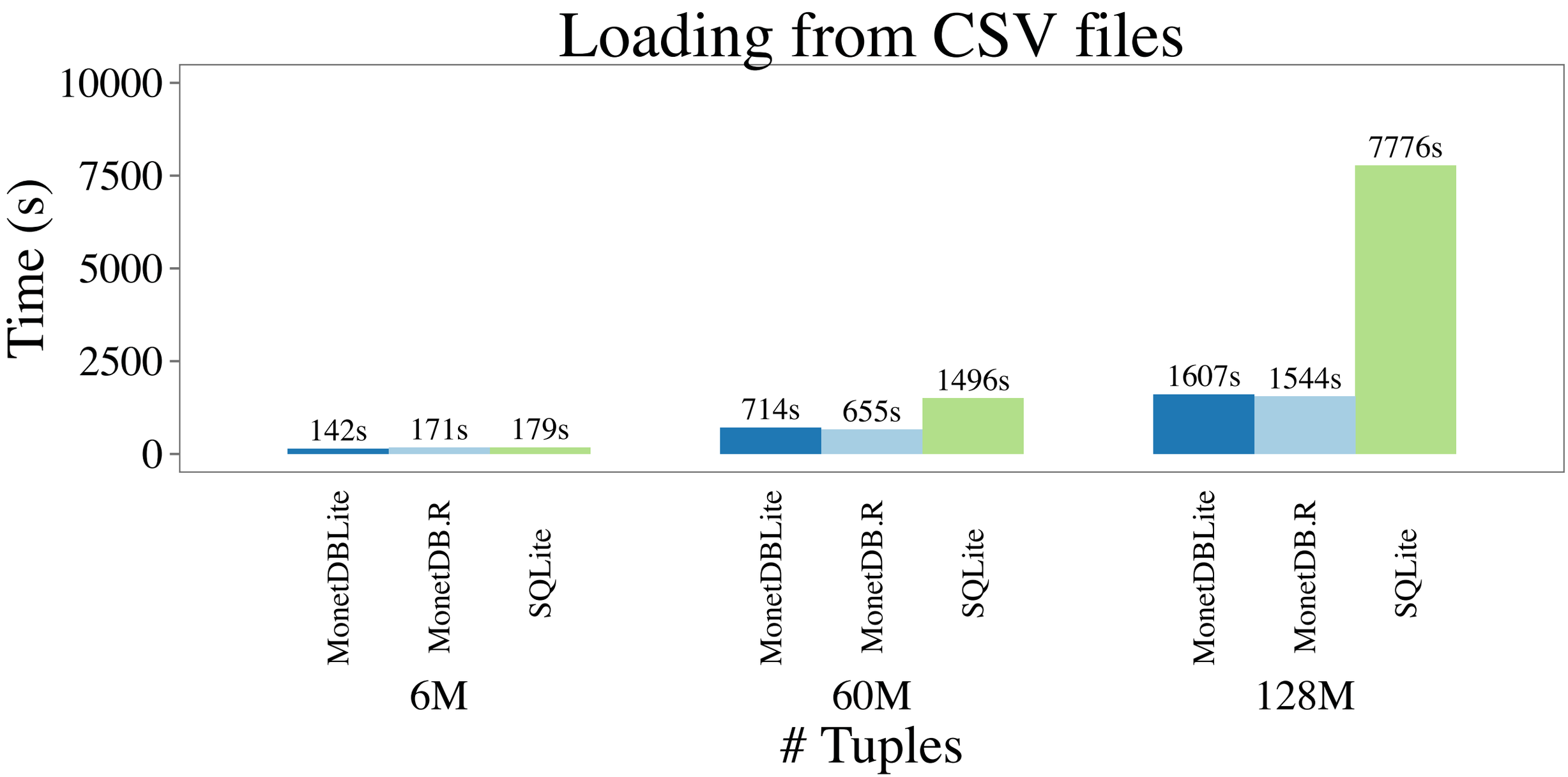
```
library(MonetDB.R)
dbdir <- tempdir()
con <- dbConnect(MonetDB.R(), embedded=dbdir)
dbWriteTable(con, "mtcars", mtcars)
dbGetQuery(con, "SELECT MAX(mpg) FROM mtcars WHERE cyl=8")
```

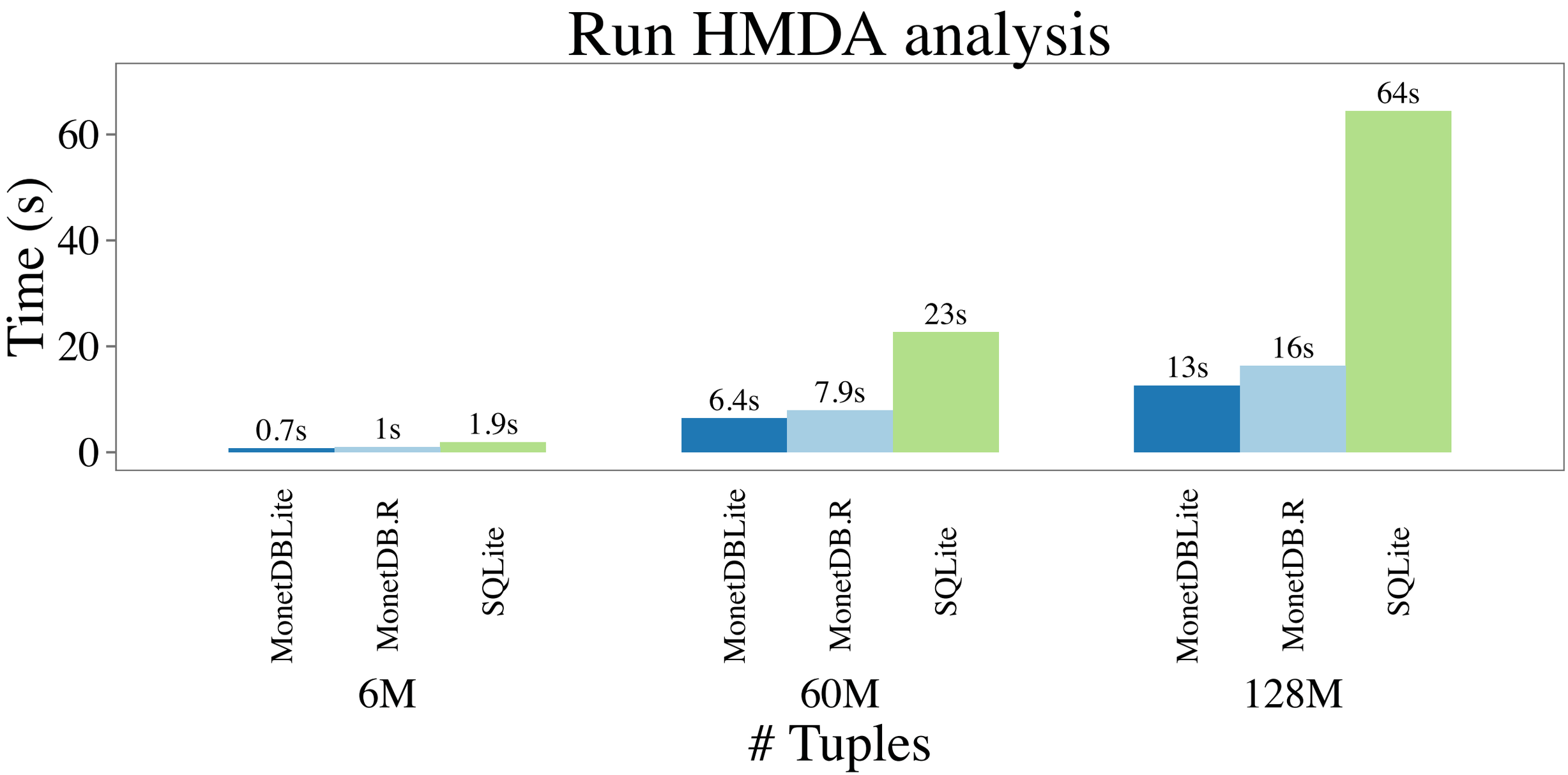
dplyr Usage

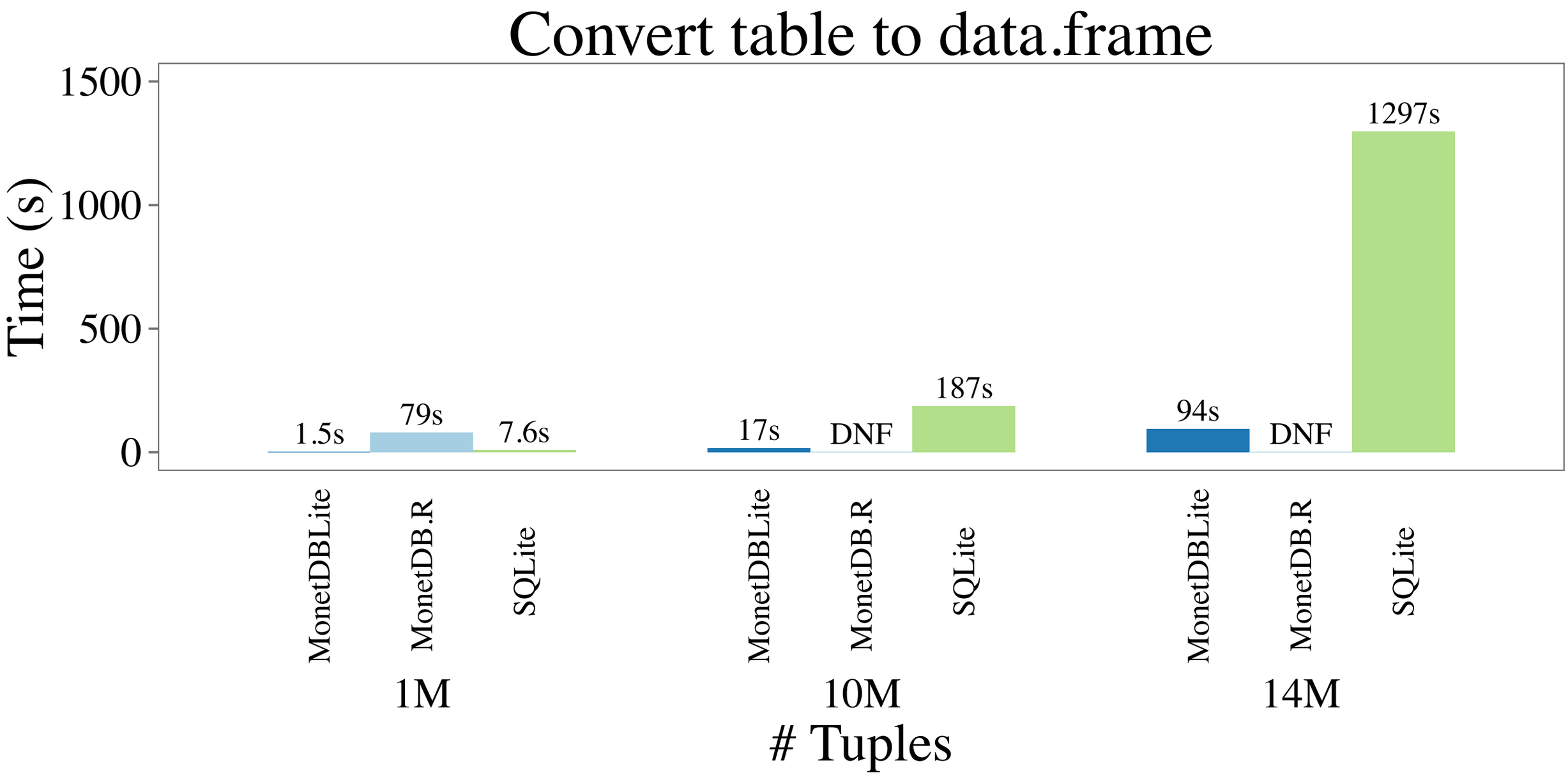
```
library(dplyr)
ms <- src_monetdb(embedded=dbdir)
mt <- tbl(ms, "mtcars")
mt %>% filter(cyl == 8) %>% summarise(max(mpg))
```

Experiments

- Home Mortgage Disclosure Act dataset & queries
  - 128M records, 71 fields each, 56 GB CSV
  - Sampled down to 6M and 60M for testing
- Contenders: MonetDB, MonetDBLite & SQLite
- Experiments:
  - CSV loading
  - HMDA queries
  - Table transfer


Loading from CSV files
Time (s)
10000
7500
5000
2500
0
142s
171s
179s
714s
655s
1496s
1607s
1544s
7776s
MonetDBLite
MonetDB.R
SQLite
MonetDBLite
MonetDB.R
SQLite
MonetDBLite
MonetDB.R
SQLite
6M
60M
128M
Tuples


Run HMDA analysis
Time (s)
60
40
20
0
0.7s
1s
1.9s
6.4s
7.9s
23s
13s
16s
64s
MonetDBLite
MonetDB.R
SQLite
MonetDBLite
MonetDB.R
SQLite
MonetDBLite
MonetDB.R
SQLite
6M
60M
128M
Tuples


Convert table to data.frame
Time (s)
1500
1000
500
0
1.5s
79s
7.6s
17s
DNF
187s
94s
DNF
1297s
MonetDBLite
MonetDB.R
SQLite
MonetDBLite
MonetDB.R
SQLite
MonetDBLite
MonetDB.R
SQLite
1M
10M
14M
Tuples

Live Demo

MonetDBLite Installation

hannes — R --no-save — R — R --no-save — 97×25

```
> install.packages(c("MonetDB.R", "MonetDBLite"),
+   repos=c("http://dev.monetdb.org/Assets/R/", "http://cran.rstudio.com/"))
Installing packages into '/Users/hannes/Library/R/3.2/library'
(as 'lib' is unspecified)
trying URL 'http://dev.monetdb.org/Assets/R/bin/macosx/mavericks/contrib/3.2/MonetDB.R_1.0.1.tgz'
Content type 'application/x-gzip' length 125556 bytes (122 KB)
==================================================
downloaded 122 KB

trying URL 'http://dev.monetdb.org/Assets/R/bin/macosx/mavericks/contrib/3.2/MonetDBLite_0.1.0.tg
z'
Content type 'application/x-gzip' length 2348737 bytes (2.2 MB)
==================================================
downloaded 2.2 MB


The downloaded binary packages are in
        /var/folders/3v/3sdy3h4j67q2crtfvh_5kqt80000gn/T//Rtmpvdx8Jk/downloaded_packages
>
```

[Reproduction of live demonstration]

MonetDBLite Startup

```
>
> library(MonetDB.R)
Loading required package: DBI
dbdir <- "/tmp/hmda_demo"
con <- dbConnect(MonetDBLite(), dbdir)
> dbdir <- "/tmp/hmda_demo"
> con <- dbConnect(MonetDBLite(), dbdir)
Loading required package: MonetDBLite
>
```

Read 110 MB CSV into R - ~13s

```
>
> system.time(dd <- read.csv("~/hmda_demo.csv", sep="|", na.strings="null", header=T, stringsAsFa
ctors=F))
   user  system elapsed
 12.798   0.177  12.981
>
```

Import table into MonetDBLite - ~1s

```
> system.time(dbWriteTable(con, "hmda_demo", dd))
   user  system elapsed
  0.708   0.141   1.111
>
```

[Reproduction of live demonstration]

Fast SQL querying

```
>
> system.time(print(dbGetQuery(con, "SELECT actiontype, propertytype, loanpurpose, COUNT(*) AS nu
m_records FROM hmda_demo GROUP BY actiontype, propertytype, loanpurpose ORDER BY actiontype, prop
ertytype, loanpurpose LIMIT 10")))

   actiontype propertytype loanpurpose num_records
1           1            1           1       41433
2           1            1           2        3271
3           1            1           3       74940
4           1            2           1         684
5           1            2           2          66
6           1            2           3         636
7           1            3           1         161
8           1            3           2          49
9           1            3           3         406
10          2            1           1        3446
   user  system elapsed
  0.017   0.014   0.082
>
```

Fast table export into R

```
>
> system.time(dd2 <- dbReadTable(con, "hmda_demo"))

   user  system elapsed
  0.281   0.026   0.315
>
```

Fast data availability after R restart

```
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> library(MonetDB.R)
Loading required package: DBI
> system.time(con <- dbConnect(MonetDBLite(), "/tmp/hmda_demo"))
Loading required package: MonetDBLite
   user  system elapsed
  0.307   0.018   0.336
> system.time(dd2 <- dbReadTable(con, "hmda_demo"))
   user  system elapsed
  0.540   0.043   0.591
>
```

Next Steps

- Single-file MonetDB (single DLL/so/dylib)
  - Need to inline startup MAL/SQL files
- Finish Python/Java wrappers
- “Restartability”
- Multiple MonetDB’s in a single process?

Centrum Wiskunde & Informatica

Questions?

https://www.monetdb.org/blog/monetdblite-r

@hfmuehleisen