Survey statistics in a database

Charco Hui

Supervisor: Professor Thomas Lumley
The University of Auckland
Department of Statistics

June 26, 2018

Problem/Goal

- Problem: Large multistage survey data sets.

Problem/Goal

- Problem: Large multistage survey data sets.
- Goal:
  - Implement R functions and testing some survey computations using the **dplyr** (Wickham et al., 2017) and **dbplyr** (Wickham and Ruiz, 2018) R packages as a database interface, **svydb**.
  - Design the functions to do as much computations in the database as possible.
  - Find out the feasibility of this approach on large survey data sets.

Large survey data sets

- Behavioral Risk Factor Surveillance System (BRFSS) - half a millions interviews per year.
- American Community Survey (ACS) - 3 million records per year.
- Nationwide Emergency Department Database (NEDS) - 25 million hospital visit records per year.

Existing software in R

- **survey** (Lumley, 2004) - Needs to read data into memory.
- **sqlsurvey** (Lumley, 2014) - Hand written SQL, portability issues.

Common formulas

Survey statistics in SQL

- Horvitz-Thompson estimator:

$$\sum_{h=1}^{L} \sum_{i=1}^{m_h} z_{hi}$$

, where

$$z_{hi} = \sum_{j \in PSU} w_{hij} x_{hij}$$

Common formulas

Survey statistics in SQL

- Horvitz-Thompson estimator:

$$\sum_{h=1}^{L}\sum_{i=1}^{m_h} z_{hi}$$

, where

$$z_{hi} = \sum_{j \in PSU} w_{hij} x_{hij}$$

- Variance estimation:

$$\sum_{h=1}^{L} \frac{m_h}{m_h - 1} \sum_{i=1}^{m_h} (z_{hi} - \bar{z}_h)^T (z_{hi} - \bar{z}_h)$$

- Population Total using the Horvitz-Thompson estimator:

```
SELECT SUM("x" * "wt") AS "Total" FROM "nh"
```

- Population Total using the Horvitz-Thompson estimator:

```
SELECT SUM("x" * "wt") AS "Total" FROM "nh"
```

- Variance scaling constant for complex surveys:

```
SELECT "strata", COUNT(DISTINCT "cluster")
    AS "m_h" FROM "nh"
```

- Population Total using the Horvitz-Thompson estimator:

```
SELECT SUM("x" * "wt") AS "Total" FROM "nh"
```

- Variance scaling constant for complex surveys:

```
SELECT "strata", COUNT(DISTINCT "cluster")
    AS "m_h" FROM "nh"
```

- Advantages/Disadvantages:
  - Efficiency vs flexibility.
  - Powerful databases.

dplyr

- Functions:

dplyr Function	**Description**	**Equivalent SQL**
select()	Selecting columns (variables)	SELECT
filter()	Filter (subset) rows.	WHERE
group_by()	Group the data	GROUP BY
arrange()	Sort the data	ORDER BY
join()	Joining tables	JOIN
mutate()	Creating New Variables (Columns)	COLUMN ALIAS

dplyr

- Functions:

dplyr Function	Description	Equivalent SQL
select()	Selecting columns (variables)	SELECT
filter()	Filter (subset) rows.	WHERE
group_by()	Group the data	GROUP BY
arrange()	Sort the data	ORDER BY
join()	Joining tables	JOIN
mutate()	Creating New Variables (Columns)	COLUMN ALIAS

- Pipes:

```
> x = sample(10)
> summary(diff(exp(floor(cos(x)))))
> x %>% cos() %>% floor() %>% exp() %>% diff()
    %>% summary()
> x %>% matrix(data = ., nrow = 1)
```

dbplyr

- Compatibility:

```
> mtdb %>% select(mpg, disp) %>% show_query()
<SQL>
SELECT "mpg", "disp" FROM "mt"
```

dbplyr

- Compatibility:

```
> mtdb %>% select(mpg, disp) %>% show_query()
<SQL>
SELECT "mpg", "disp" FROM "mt"
```

- Database backends:
  - MonetDB (MonetDB-B.V., 2008)
  - SQLite (Team, 2018)
  - Google BigQuery (Google, 2018)

Coding with dplyr

Usage

Differences:

```
x = 1; mean(x)
mtcars %>% select(mpg)
```

- Non-standard evaluation:

```
f1 = function(x, data){
  data %>% select(x)
}
f1(x = mpg, data = mtcars)
```

Coding with dplyr

Usage

Differences:

```
x = 1; mean(x)
mtcars %>% select(mpg)
```

- Non-standard evaluation:

```
f1 = function(x, data){
  data %>% select(x)
}
f1(x = mpg, data = mtcars)
```

- Quasi-quotation:

```
f2 = function(x, data){
  x = enquo(x)
  data %>% select(!!x)
}
f2(x = mpg, data = mtcars)
```

Difficulties

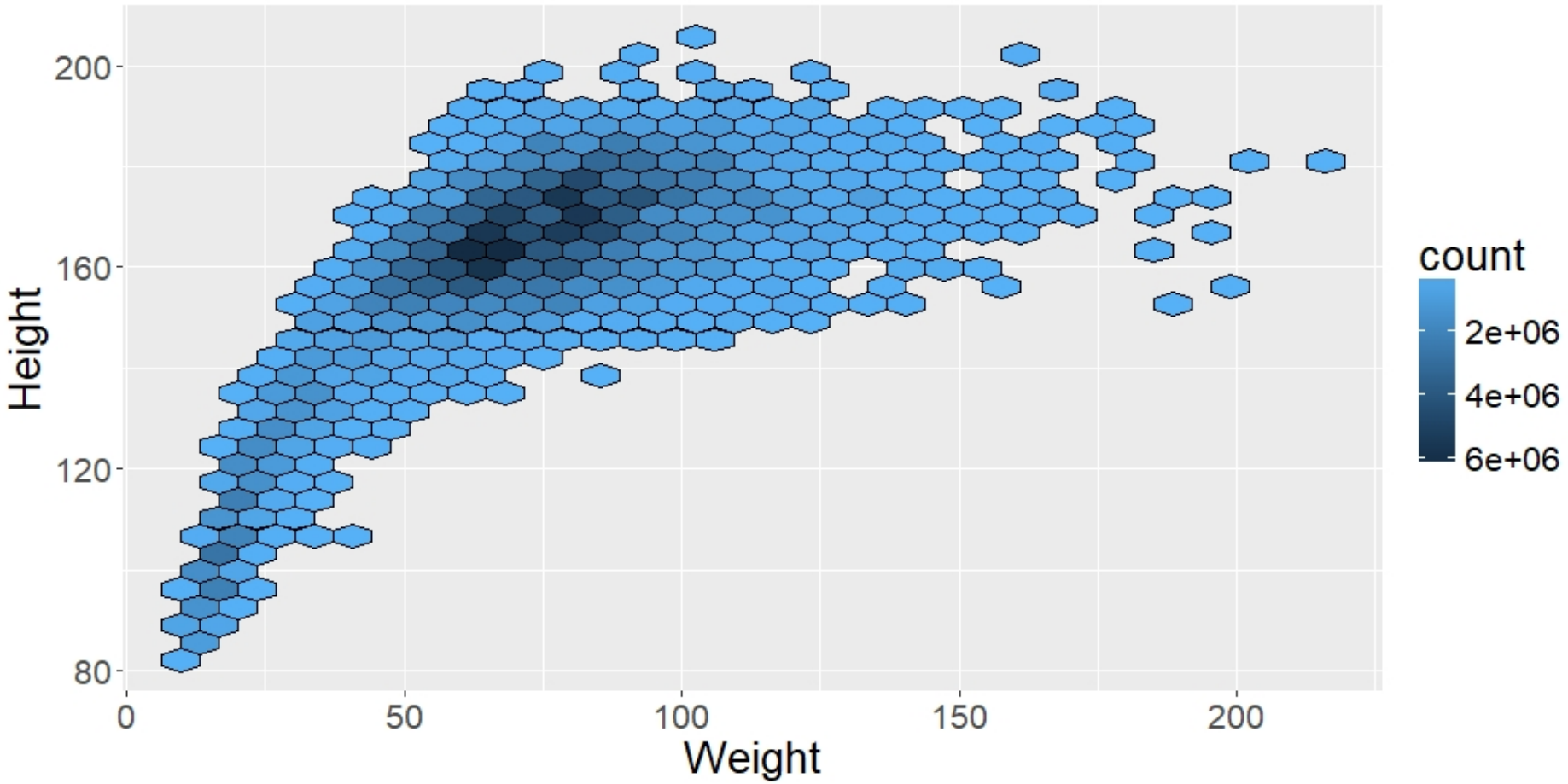
- No factor types in SQL.
- Difficult to code with quasi-quotation.
- Cannot do row-wise operations due to the lazy interface. That is, the data sets within a database in R will not be loaded into memory unless required.
- No matrix operations.
- No base R functions.
- No distributions.
- Inconsistent availability of functions between databases.

Hexagon Binning

svydb

Hexagon Binning

svydb



Hexagon Binning

Method

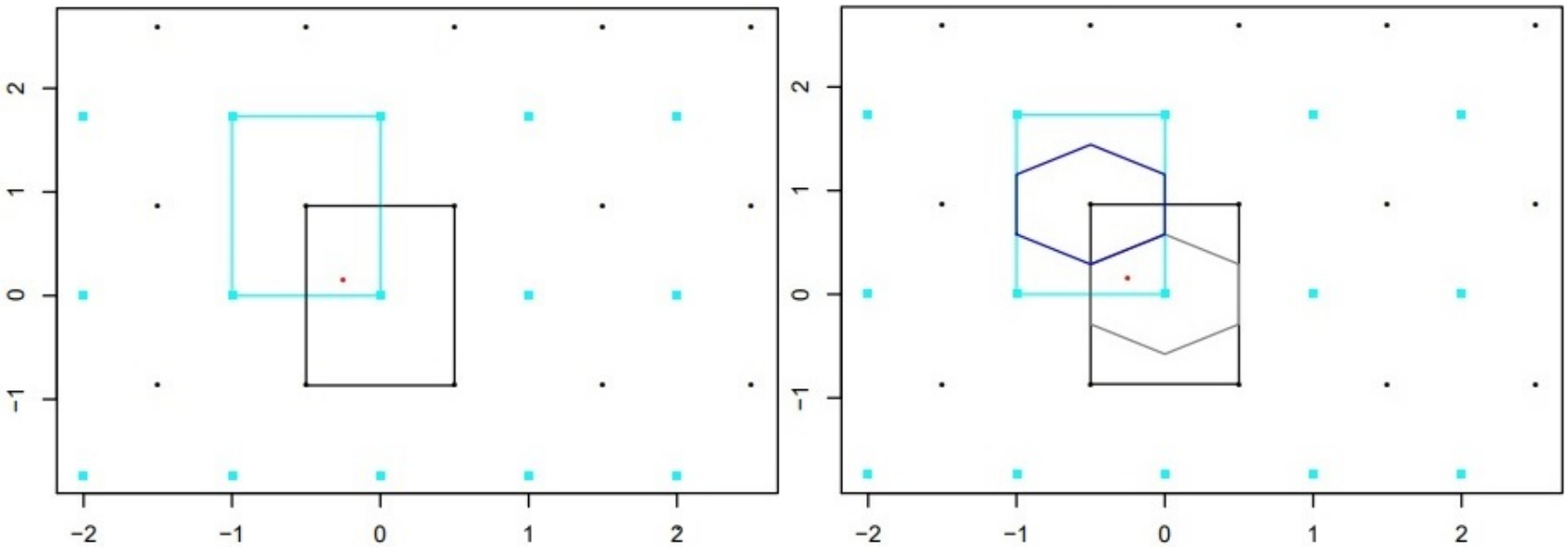
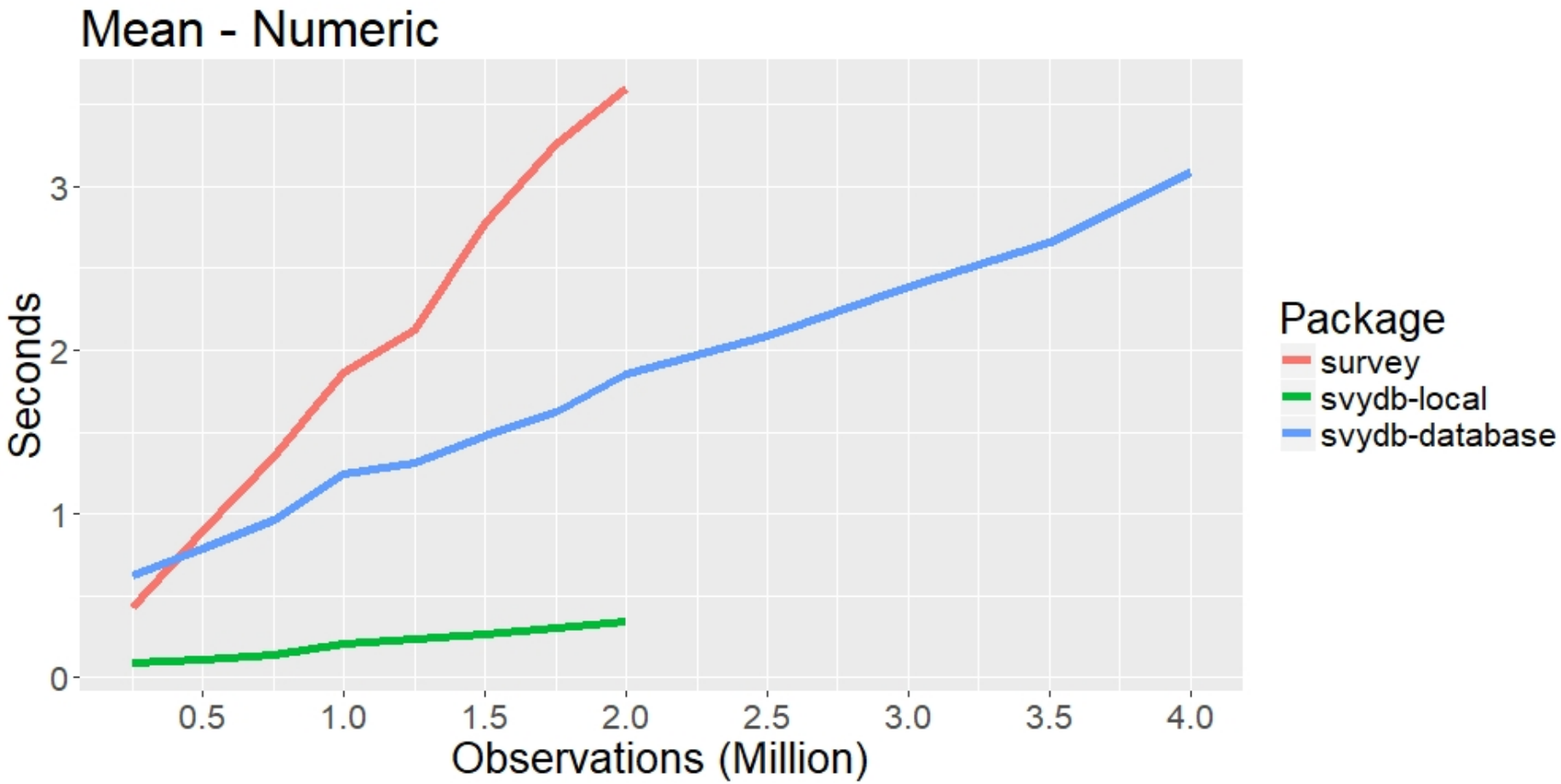


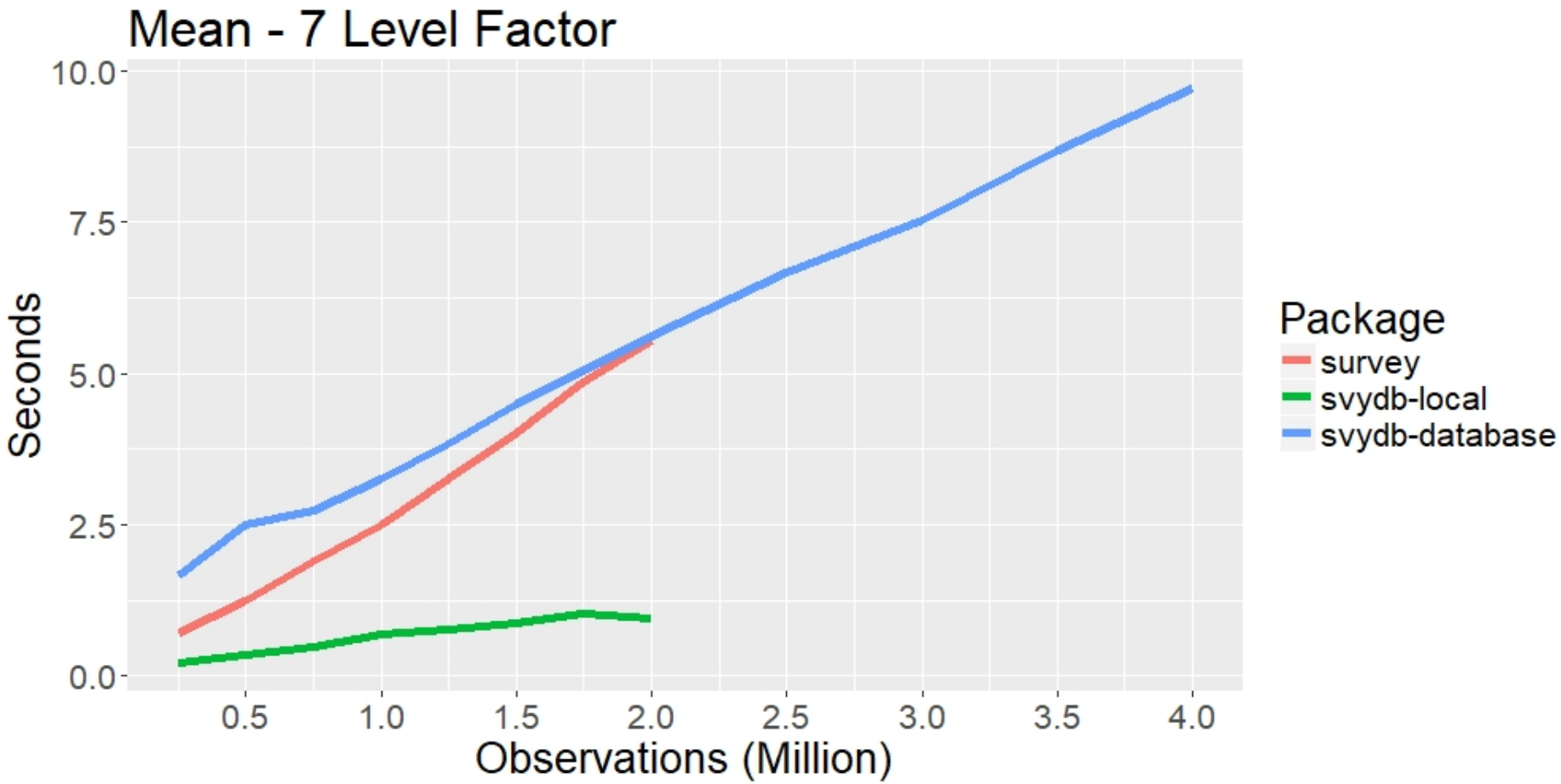
Figure: Hexagon binning explanation, (Lewin-Koh, 2016)

Times

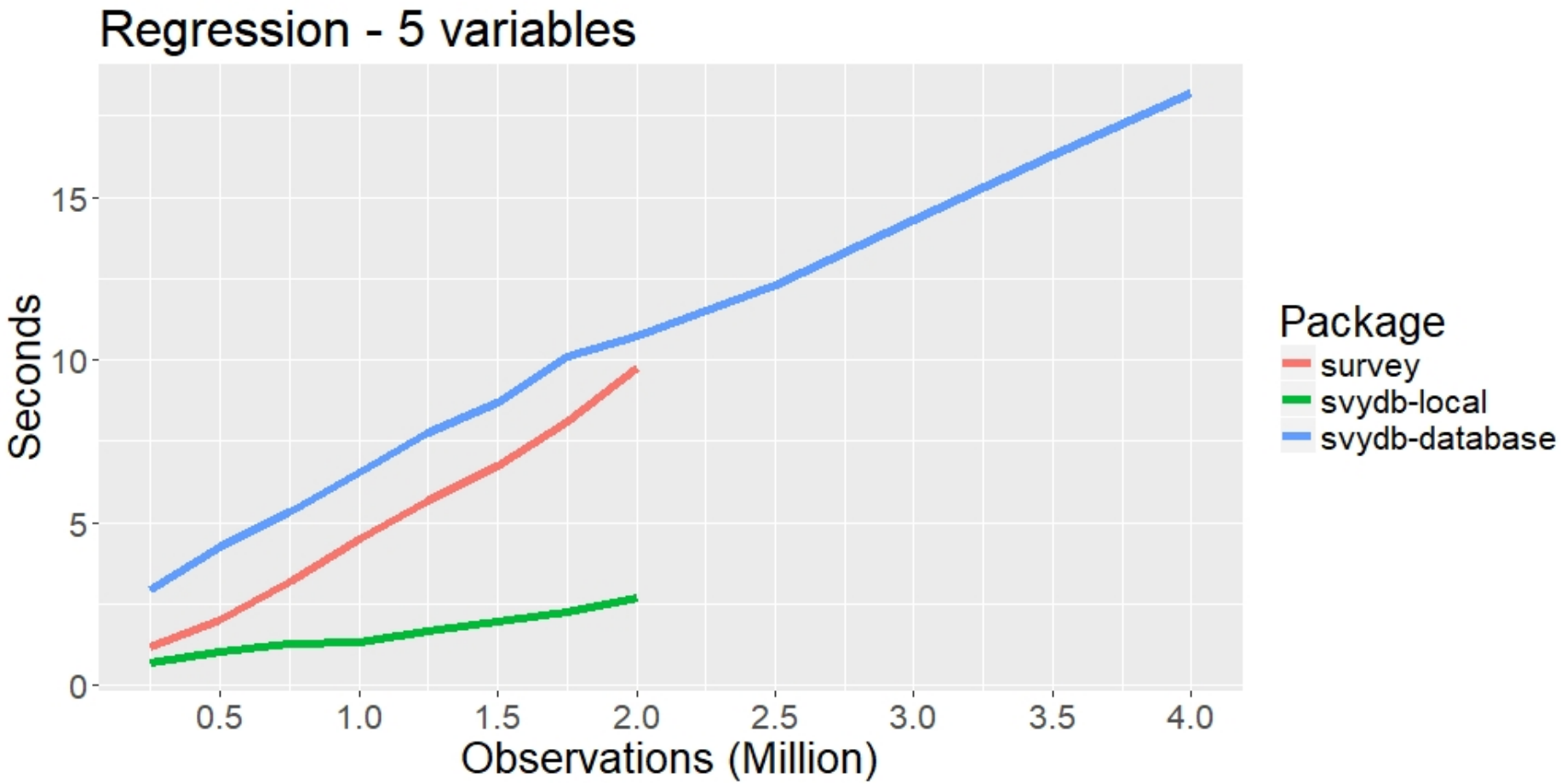
Times


Mean - Numeric
Seconds
0
1
2
3
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Observations (Million)
Package
survey
svydb-local
svydb-database

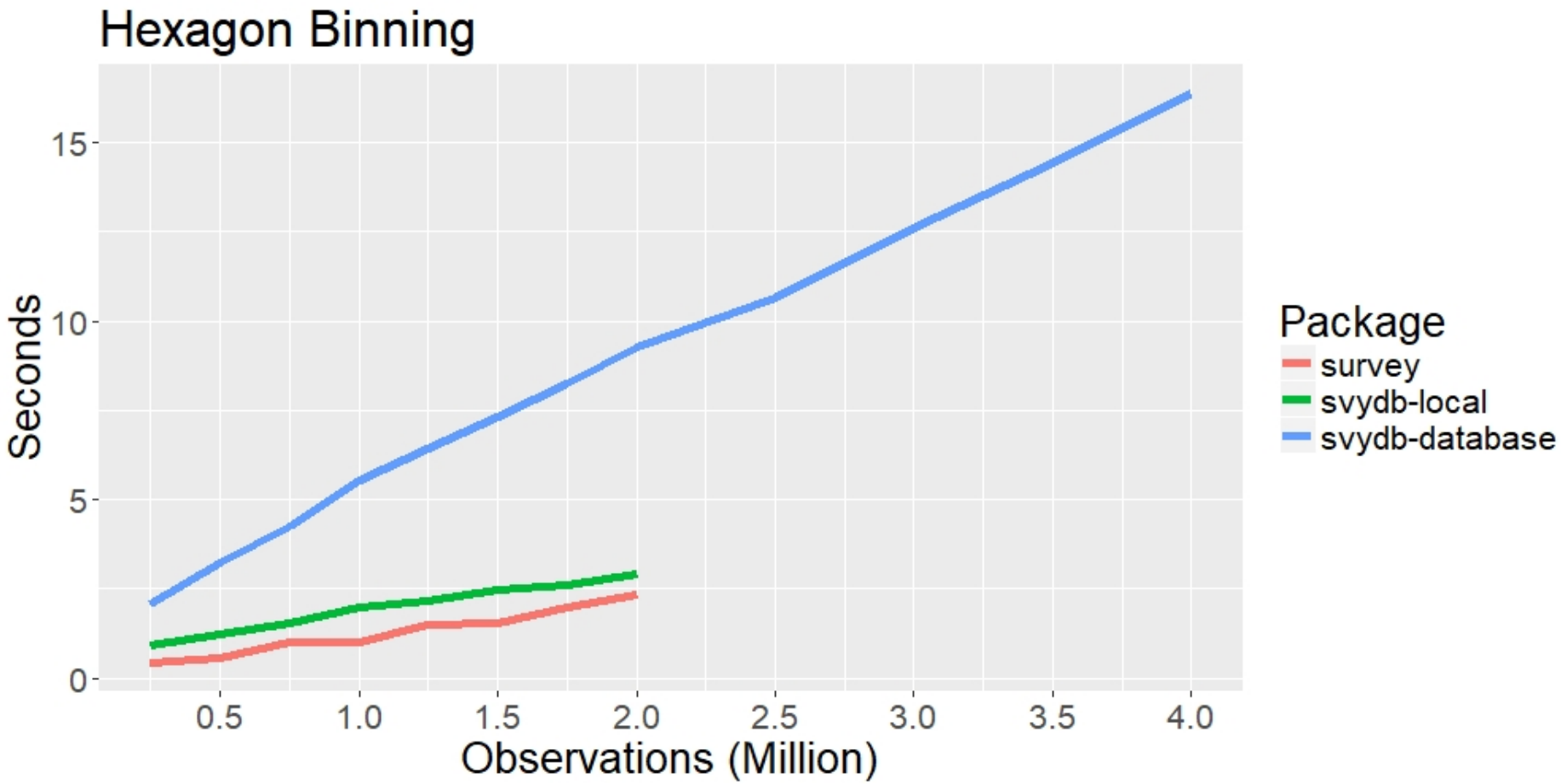
Times


Mean - 7 Level Factor
Seconds
10.0
7.5
5.0
2.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Observations (Million)
Package
survey
svydb-local
svydb-database

Times


Regression - 5 variables
Seconds
0
5
10
15
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Observations (Million)
Package
survey
svydb-local
svydb-database

Times


Hexagon Binning
Seconds
0
5
10
15
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Observations (Million)
Package
survey
svydb-local
svydb-database

List of functions in svydb

- Statistics:
  - `svydbdesign()`
  - `svydbtotal()`
    - `coef()`
    - `SE()`
  - `svydbmean()`
  - `svydblm()`
    - `summary()`
    - `predict()`
  - `svydbquantile()`
  - `svydbtable()`
  - `svydbby()`
  - `svydbrepdesign()`
  - `svydbreptotal()`
  - `svydbrepmean()`
- Graphics
  - `svydbhist()`
  - `svydbboxplot()`
  - `svydbhexbin()`
  - `svydbhexplot()`
  - `svydbcoplot()`

Conclusion

- It is feasible to compute survey statistics in SQL as long as:
  - No heavy iterations.
  - Not dependent on mathematical or statistical operations.

Conclusion

- It is feasible to compute survey statistics in SQL as long as:
  - No heavy iterations.
  - Not dependent on mathematical or statistical operations.
- Faster on large data sets, for most statistics.

Conclusion

- It is feasible to compute survey statistics in SQL as long as:
  - No heavy iterations.
  - Not dependent on mathematical or statistical operations.
- Faster on large data sets, for most statistics.
- It can give accurate results, checked with the **survey** package.

Thank You.

Carr, D. B. et al. (1987). "Scatterplot Matrix Techniques for Large N". In: *Journal of the American Statistical Association* 82.398, pp. 424–436. ISSN: 01621459. URL: http://www.jstor.org/stable/2289444.

Carr, Dan et al. (2018). *hexbin: Hexagonal Binning Routines*. R package version 1.27.2. URL: https://CRAN.R-project.org/package=hexbin.

Google (2018). *Analytics Data Warehouse — Google Cloud*. URL: https://cloud.google.com/bigquery/.

Hui, Charco (2018). *svydb: Survey analysis in a database.*. URL: https://github.com/chrk623/svydb.

Lewin-Koh, Nicholas (2016). *Hexagon Binning: an Overview*. URL: https://cran.r-project.org/web/packages/hexbin/vignettes/hexagon_binning.pdf.

Lumley, Thomas (2004). "Analysis of Complex Survey Samples". In: *Journal of Statistical Software* 9.1. R package verson 2.2, pp. 1–19.

— (2014). *sqlsurvey: analysis of very large complex survey samples (experimental)*. R package version 0.6-11/r41. URL: https://R-Forge.R-project.org/projects/sqlsurvey/.

MonetDB-B.V. (2008). *MonetDB Database System*. URL: https://www.monetdb.org/.

Team, SQLite Development (2018). *Self-contained, high-reliability, embedded, full-featured, public-domain, SQL database engine.*. URL: https://www.sqlite.org/.

Wickham, Hadley and Edgar Ruiz (2018). *dbplyr: A 'dplyr' Back End for Databases*. R package version 1.2.1. URL: https://CRAN.R-project.org/package=dbplyr.

Wickham, Hadley et al. (2017). *dplyr: A Grammar of Data Manipulation*. R package version 0.7.4. URL: https://CRAN.R-project.org/package=dplyr.