Survey analysis

Thomas Lumley
Biostatistics
U. Washington
tlumley@u.washington.edu

Issues

* Survey data sets are often big
  - 10^4-10^5 people, 10^6 businesses
  - 10^3 variables
* Survey analysts often have conservative hardware/software preferences
  - not 32Gb Opteron boxes

Non-toy example

* National Health Interview Survey:
  - ➡ 25000 people, 500 variables
  - ➡ c.100Mb data frame in R
  - ➡ 240Mb SQLite database

* possible, but painful, in R on 1Gb laptop
  - ➡ 357Mb Vcells after linear model
* easy in R on 32Gb server
* easy, but slower, in R<->SQL on laptop.
  - ➡ 7Mb Vcells after linear model

SQL

* Data processing language
* Standardized (?)
* Math (as far as sums and products).
* Big money in making SQL fast....

Survey statistics

* Mostly simple summaries, occasional regression model
* Based on Horvitz-Thompson estimator for population totals
* HT estimator is just sums and products

➡ How much survey statistics can we write as SQL code?

Computing near the data

* Chen & Ripley (DSC 2003):
  - ➡ outsource large-data computation to databases, avoiding data transfer bottleneck
  - ➡ R writes SQL queries to control computation
  - ➡ hack into Postgres and the R evaluator to provide transparent interface for user.

* R evaluator hacks are too hard to maintain, but rest of concept can be stolen.

Computing near the data

* Data stored in SQLite database
* R creates SQL queries
* Large results go to new SQL tables
* Small results returned to R
  - ➡ often just a few kb

SQL queries

A typical query form:

```
SELECT SUM(_x), SUM(_x*_x) FROM
  (SELECT SUM(x*wt) AS _x, stratum
  FROM data GROUP BY cluster)
GROUP BY stratum
```

SQL queries

R function to help

```
sqlsubst("SELECT %%vars%% FROM
%%table%% GROUP BY %%strat%%",
list(vars= varnames,
table=tablequery, strat=strata)
)
```

User interface

R code with formulas and objects, essentially same syntax as survey package.

```
sqclus1 <- sqlsurvey(id = "dnum", fpc = "fpc",
   weights = "pw", data = "apiclus1.db",
   table.name = "clus1",
   key = "snum")

svymean(~api99, design = sqclus1, byvar = ~stype)

svytotal(~enroll+stype,
          design = subset(sqclus1, api99 > 500))
svylm(api00~api99+stype*comp_imp, design = sqclus1)

close(sqclus1)
```

Object structure

* Survey design object contains:
  * database connection
  * table name, and subset table name if a subset
  * name of unique identifier variable
  * zero-row data frame specifying types and factor levels
  * character vectors giving stratum, cluster, weight variable names.

Model matrix

* Basic model matrix constructions do not depend on the data (except through types, factor levels).
* Model matrix columns are simple arithmetical functions of model frame columns

➡ Use R functions on zero-row subset to lay out model matrix, then write SQL to create it.

➡ Store temporary table names in an environment, use a finalizer to destroy them on garbage collection.

Estimating functions

* Estimating functions are also stored in a temporary table, but deleted by `on.exit()`.
* Linked to survey meta-data by `INNER JOIN` on unique identifier variable
* Don't need to modify tables, only `CREATE TABLE`.
* Table names passed to function for HT estimator, so standard error estimation is generic.

Subsets

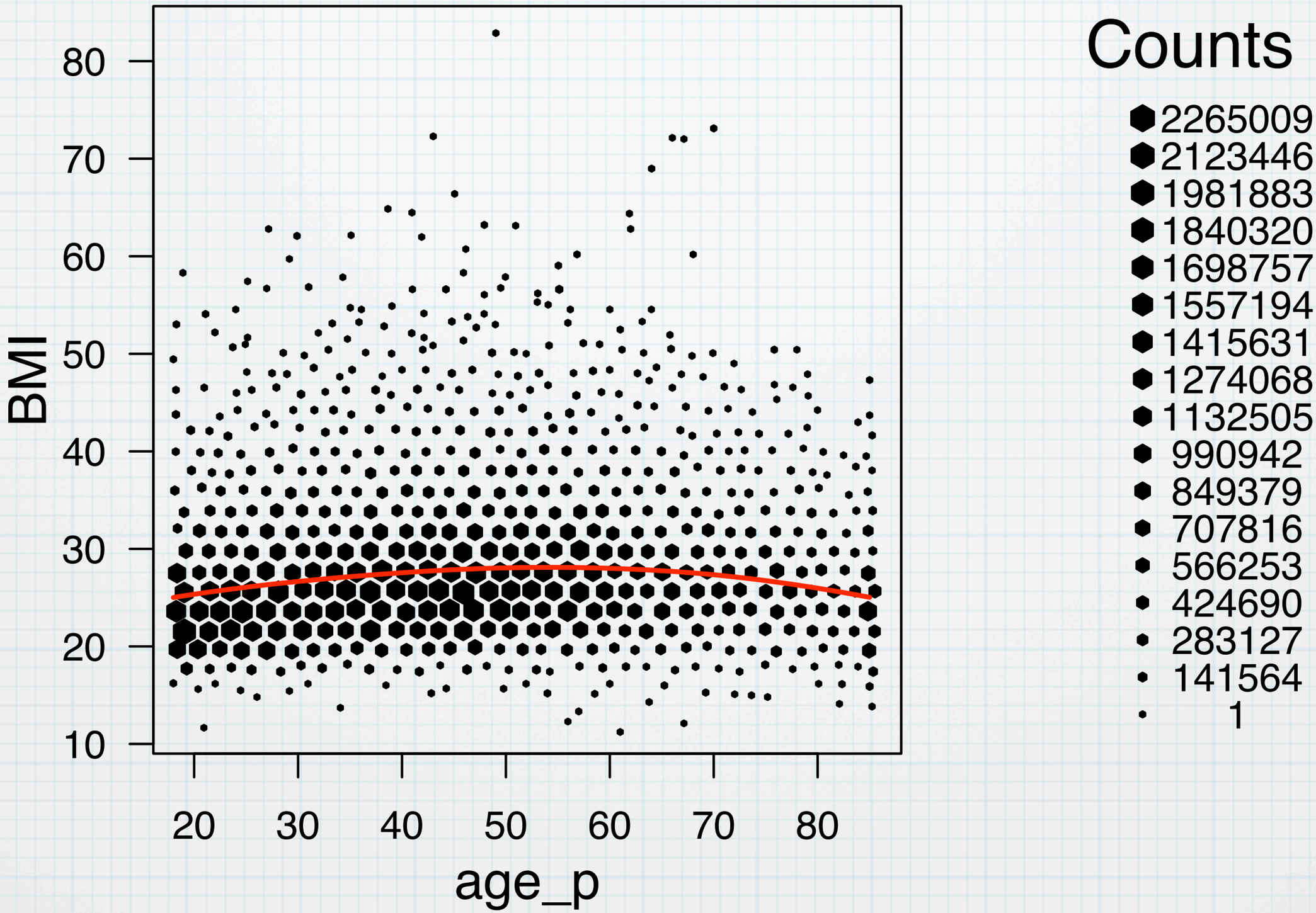
* Subset is a new design with some weights set to zero
* Don't copy all the variables, just weights and identifier
* R and SQL have different expression syntax, so we translate the parsed expressions

```
R:   (age < 65) & (state %in% c("IA", "WA")

SQL: (age < 65) AND (state IN ("IA", "WA"))
```

Graphics

* Kernel smoothing
  - ➡ binning in SQL, then KernSmooth
* hexagonal binning
  - ➡ seems to require full data transfer (in chunks)
  - ➡ code to merge two hexbins


BMI
80
70
60
50
40
30
20
10
20
30
40
50
60
70
80
age_p
Counts
2265009
2123446
1981883
1840320
1698757
1557194
1415631
1274068
1132505
990942
849379
707816
566253
424690
283127
141564
1

State of play

* SQLite:
  - ➡ other SQL engines should be easy
* Multistage stratified random samples:
  - ➡ Calibration, two-phase designs probably feasible
  - ➡ Replicate weights should be straightforward.
  - ➡ PPS is hard
* Means, totals, quantiles, linear regression
  - ➡ Poisson, logistic, Cox require `exp` and `log`, which are not standard SQL but are common extensions

Where can I get some?

surveyNG package is on CRAN.

Currently has SQL-backed facilities as described here, plan to add sparse-matrix methods for moderate-size designs.

Home page for surveyNG (and survey):

http://faculty.washington.edu/tlumley/survey/