Estimating ALL THE THINGS

Thomas Lumley

8(9) July 2025

Why we care

Case-control studies work because of special properties of the odds ratio; they are limited to logistic regression.

Traditional survey statistics was largely about population counts and means.

With modern weighted estimation for complex sampling we can estimate almost anything, so anyone who can do ordinary data analysis can analyse complex surveys without much special knowledge.

Compared to maximum likelihood or imputation

Often possible to get more precise estimates with semiparametric maximum likelihood or multiple imputation – eg *case-control logistic regression*

Extra precision comes from additional assumptions: case-control $\hat{\beta}$ only estimates the census parameter if the model is exactly correct

The robustness:efficiency tradeoff is real; it's not clear which side you want to be on.

More Research Is Needed.

Han, Shaw, Lumley (2021) *Statistics in Medicine*. 40(30): 6777–6791.

Notation

- N population size
- n sample size
- π_i probability that unit i would be sampled
- π_{ij} probability that both units i and j would be sampled
- w_i weights, usually (adjusted versions of) π_i^{-1}
- R_i sampling indicator: $E[R_i] = \pi_i$

Estimating population totals

Population total T_X of X is

$$T_X = \sum_{i=1}^{N} X_i$$

Horvitz-Thompson estimator is

$$\hat{T} = \sum_{i=1}^{N} \frac{R_i}{\pi_i} X_i$$

Since $E[R_i/\pi_i] = 1$, $\hat{T}$ is unbiased as long as $\pi_i > 0$ for all units in the population.

How we usually estimate things

- Maximising or minimising some *objective function*
  - *Maximum likelihood*
  - *Least squares*
- Solving some *estimating equation*

Because of calculus, these are close to being equivalent:

- Maximise

$$\ell(\beta) = \sum_{i=1}^{N} \ell_i(\beta)$$

- Solve

$$U(\beta) = \sum_{i=1}^{N} U_i(\beta) = \sum_{i=1}^{N} \frac{\partial \ell_i}{\partial \beta} = 0$$

Example

Linear regression minimises the *residual sum of squares*, with $\ell_i = (y - x_i\beta)^2$

$$\sum_{i=1}^{N}(y_i - x_i\beta)^2 = 0$$

which is equivalent to solving the set of equations (the *normal equations*)

$$\sum_{i=1}^{N} x_i(y_i - x_i\beta) = 0$$

Basic estimation question

Using just the sample, how can I get an estimate of *the same quantity* that I would estimate if I had the whole population?

The whole-population answer (called the *census parameter*) is defined by the estimating equation or objective function applied to the population

Those are population totals. We understand population totals. We can *estimate* $\ell(\beta)$ or $U(\beta)$

Example

In linear regression, the census parameter minimises the population residual sum of squares:

$$\tilde{\ell}(\beta) = \sum_{i=1}^{N} (y_i - x_i\beta)^2$$

We can get an unbiased estimate of this

$$\widehat{\ell}(\beta) = \sum_{i=1}^{N} \frac{R_i}{\pi_i} (y_i - x_i\beta)^2$$

and minimise $\hat{\ell}(\beta)$ instead.

If $\hat{\ell}(\beta)$ is close to $\ell(\beta)$, the β that minimises one must be close to the β that minimises the other, so we get a good estimate of the census parameter.

Simpler example: mean

The population mean μ solves

$$\sum_{i=1}^{N}(X_i - \mu) = 0$$

The estimated mean solves

$$\sum_{i \in \text{sample}} w_i(X_i - \mu) = 0$$

Rearranging,

$$\hat{\mu_x} = \frac{\hat{T_x}}{\sum_i w_i} = \frac{\hat{T_x}}{\hat{N}}$$

Digression: why not divide by N?

- $\hat{N}$ is what the formula gives
- $\hat{N}$ often gives better performance, because $\hat{T}$ and $\hat{N}$ correlated
- Sometimes (eg, in cluster sampling) we don't know N
- When we do know N, the weights are often *calibrated* so $\hat{N} = N$

When does this work?

Treating the objective function or estimating equations as a population total works for

- means, totals, contingency tables
- linear and generalised-linear models
- survival analysis
- proportional odds and related ordinal models
- quantiles and quantile regression
- scatterplot smoothers
- and many more

How to get the computer to do it

In Stata, most built-in and some user-written estimation commands allow a `svy:` prefix that tells the computer to

- use weights in the objective function/estimating equation
- use the appropriate resampling or sandwich estimator for standard errors

In R, the `survey` package has

- `svymean`, `svyquantile`
- `svyglm`, `svyolr`, `svyloglin`, `svyivreg`
- `svysmooth`
- `svykm`, `svycoxph`

and `svyVGAM` has multinomial, negative binomial, zero inflated/truncated, etc

When doesn't it work?

Not easily for mixed models, where the likelihood isn't a sum over individual observations. Mixed models are hard: there's a new package `svylme` for **linear** mixed models, but even then you need probabilities at each stage.

Not straightforwardly for purely *predictive* models, where you care about prediction in a new sample rather than estimating the correct values of β. The next version of `survey` will have `withCrossval` for cross-validation

Not straightforwardly for rank-based statistics, because the appropriate definition of ranks is tricky (though we now have `svyranktest`, `svylogrank`)

Standard errors

The standard error estimation is based on the (model-agnostic, or model-robust) *sandwich* estimator, with the middle of the sandwich being a Horvitz-Thompson estimator.

For surveys defined by replicate weights, the standard errors are just the resampling standard errors.

Tests in regression

Basic idea: Rao & Scott (1981,1984) work out the sampling distribution of the weighted likelihood ratio statistic and the Pearson X^2 score statistic in contingency tables

Lumley & Scott (2013,2014) extend this to Cox model and generalised linear models with arbitrary covariates, implemented in `regTermTest`

Lumley & Scott (2019) extend it to AIC, implemented for `svyglm` objects in R.

Currently not in any other software.

Note: assumptions in regression

For inference about population associations:

- distribution of residuals not important
- getting the right functional form is important for confounders, not (as much) for exposure of interest

Do we really need weights?

If $E[Y|X = x] = x\beta$, so the model is correctly specified **and** the weights are independent of Y given X

- no bias from omitting weights
- loss of precision from including weights

Bias/variance tradeoff: the larger the survey, the more we care about bias, so the more we want to include the weights

The data **can** tell us the weights are needed – leaving them out makes a big difference to the estimates.

The data **can't** tell us they aren't needed (the bias can't be estimated accurately enough)

Examples

We will use data from the Scottish Household Survey (slightly modified for confidentiality). The survey is **stratified** by local authority (roughly, county in US) and by ten socioeconomic categories at the postcode level.

The survey uses cluster sampling for rural areas, but samples individuals for areas with higher population density, so the **primary sampling unit** is sometimes the individual and sometimes a cluster

survey design

```
shs<-read.csv("shs.csv")
names(shs)
```

```
##  [1] "X"         "psu"       "uniqid"    "ind_wt"    "shs_6cla"  "council"
##  [7] "rc5"       "rc7e"      "rc7g"      "intuse"    "groupinc"  "clust"
## [13] "stratum"   "age"       "sex"       "emp_sta"   "grosswt"   "groc"
```

```
dshs<-svydesign(id=~psu,weights=~grosswt,strata=~stratum,
             data=shs)
dshs
```

```
## Stratified 1 - level Cluster Sampling design (with replacement)
## With (11937) clusters.
## svydesign(id = ~psu, weights = ~grosswt, strata = ~stratum, data = shs)
```

summaries of internet use

```
svymean(~intuse, design=dshs, na.rm=TRUE)
```

```
##           mean     SE
## intuse 0.34156 0.0034
```

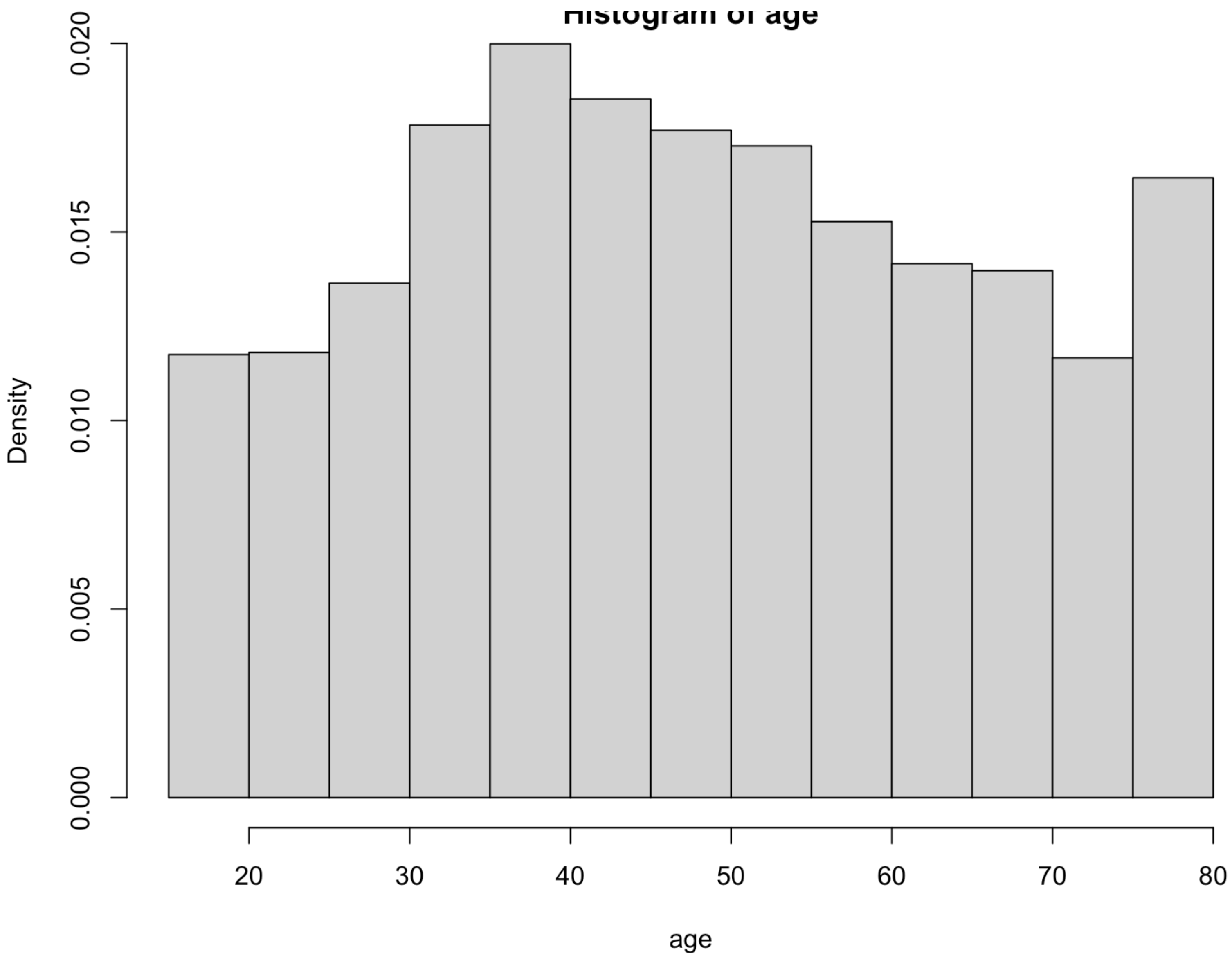
```
svymean(~sex, design=dshs, na.rm=TRUE)
```

```
##              mean     SE
## sexfemale 0.55815 0.0033
## sexmale   0.44185 0.0033
```

```
svymean(~age, design=dshs, na.rm=TRUE)
```

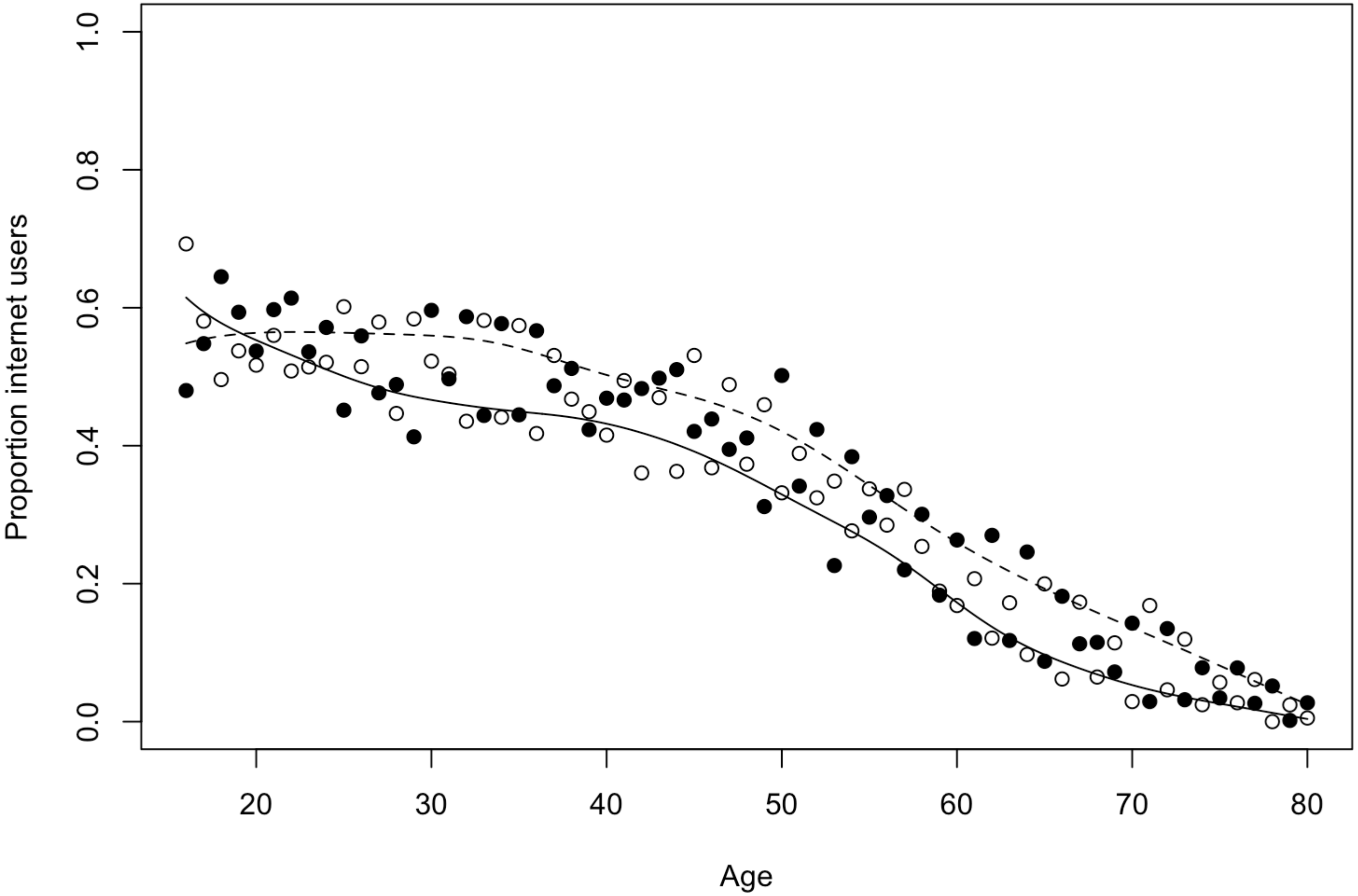
```
##       mean     SE
## age 48.224 0.1227
```

```
svyhist(~age,design=dshs)
```



Graph by age and sex

```
byagesex<-svyby(~intuse,~age+sex,svymean, design=dshs,na.rm=TRUE)
m<-svysmooth(intuse~age,design=subset(dshs,sex=="male"))
f<-svysmooth(intuse~age,design=subset(dshs,sex=="female"))
plot(rep(16:80,2),coef(byagesex),pch=c(1,19),ylim=c(0,1),
     xlab="Age",ylab="Proportion internet users")
lines(m,lty=2)
lines(f,lty=1)
```

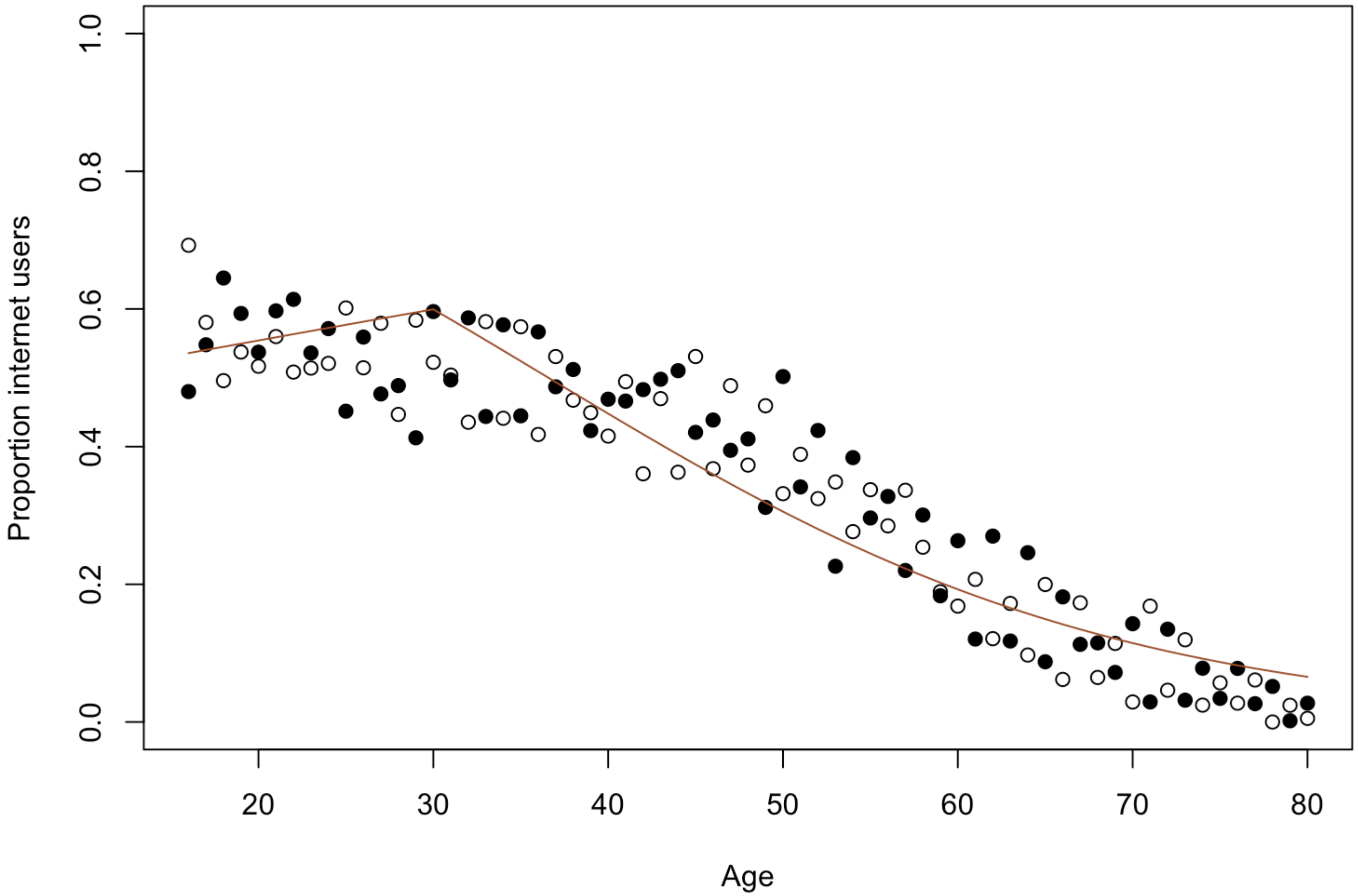

Proportion internet users
0.0
0.2
0.4
0.6
0.8
1.0
20
30
40
50
60
70
80
Age

with logistic regression

```
agemodel<-svyglm(intuse~(pmin(age,30)+pmax(age,30)),design=dshs,
                 family=quasibinomial)
coef(summary(agemodel))
```

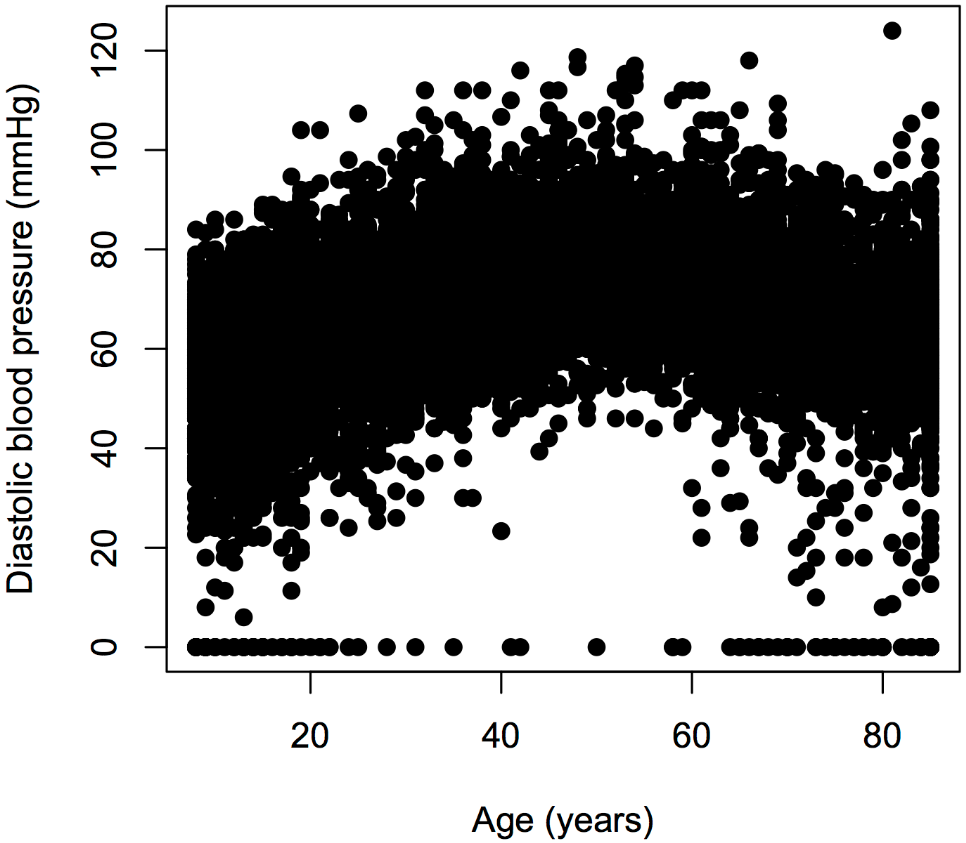
```
##                    Estimate  Std. Error    t value     Pr(>|t|)
## (Intercept)    1.68279935 0.144283650  11.663133 2.932990e-31
## pmin(age, 30)  0.01853921 0.005454601   3.398821 6.790367e-04
## pmax(age, 30) -0.06119598 0.001150673 -53.182750 0.000000e+00
```

```
fittedage<-predict(agemodel,newdata=data.frame(age=16:80),type="response")
plot(rep(16:80,2),coef(byagesex),pch=c(1,19),ylim=c(0,1),
     xlab="Age",ylab="Proportion internet users")
lines(16:80,(fittedage),col="sienna")
```


1.0
0.8
0.6
0.4
0.2
0.0
Proportion internet users
20
30
40
50
60
70
80
Age

Scatterplots?

NHANES (14000 points)



Hard because

- data often large
- want to incorporate weights
- want to incorporate correlation within sampling units

Approaches:

- alpha-blending
- hexagonal binning

Code (not Stata)

`svyplot()`: `style` can be `transparent` or `hex`, `grayhex`,

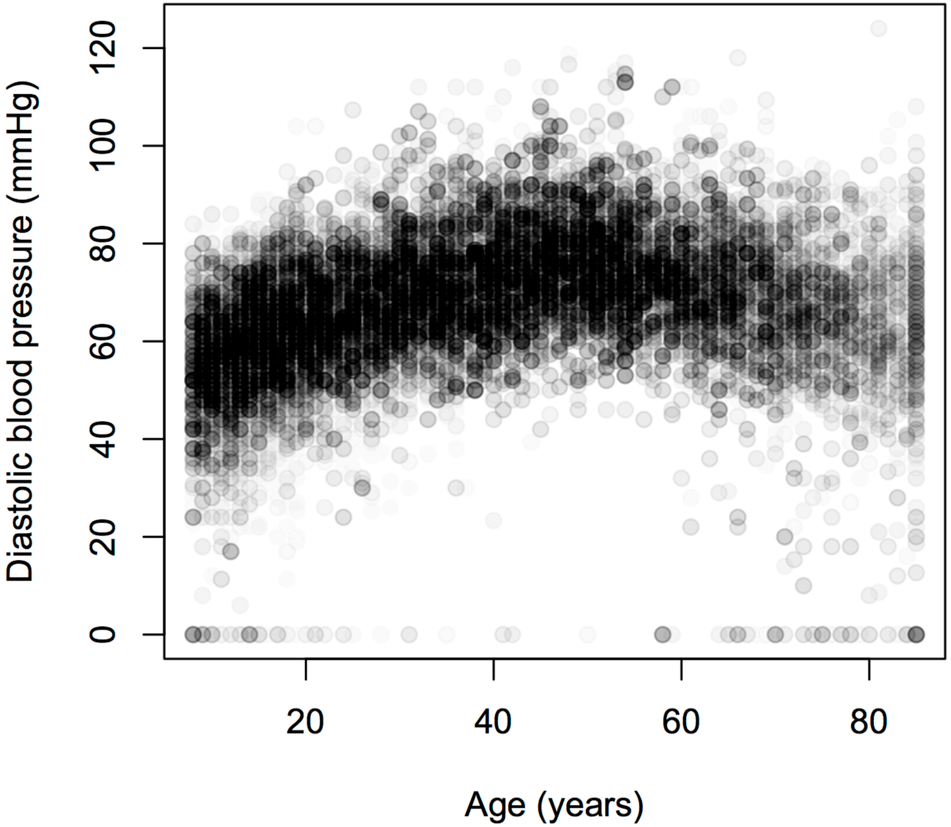
`svycoplot()`: `style` can be `transparent` or `hexbin`

Alpha-blending

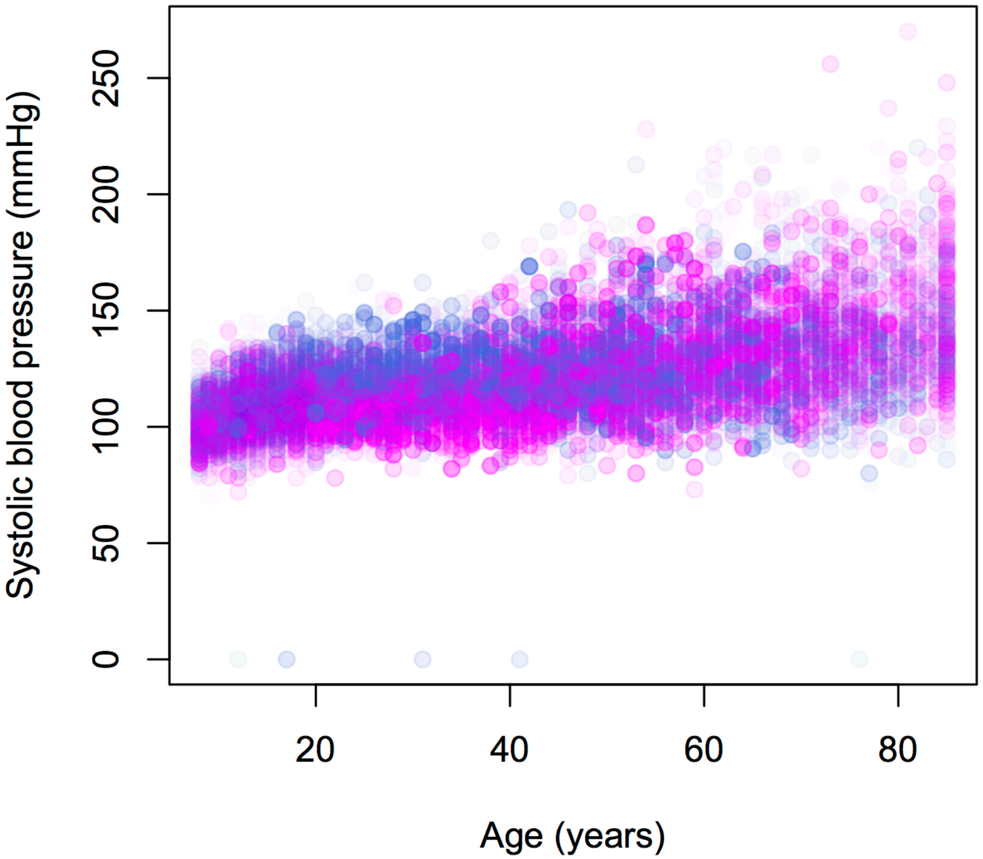
Use partially-transparent points:

- overplotting can still be seen
- amount of ink proportional to sampling weight
- can still use colour to identify groups

NHANES: 14000 points



NHANES: 14000 points

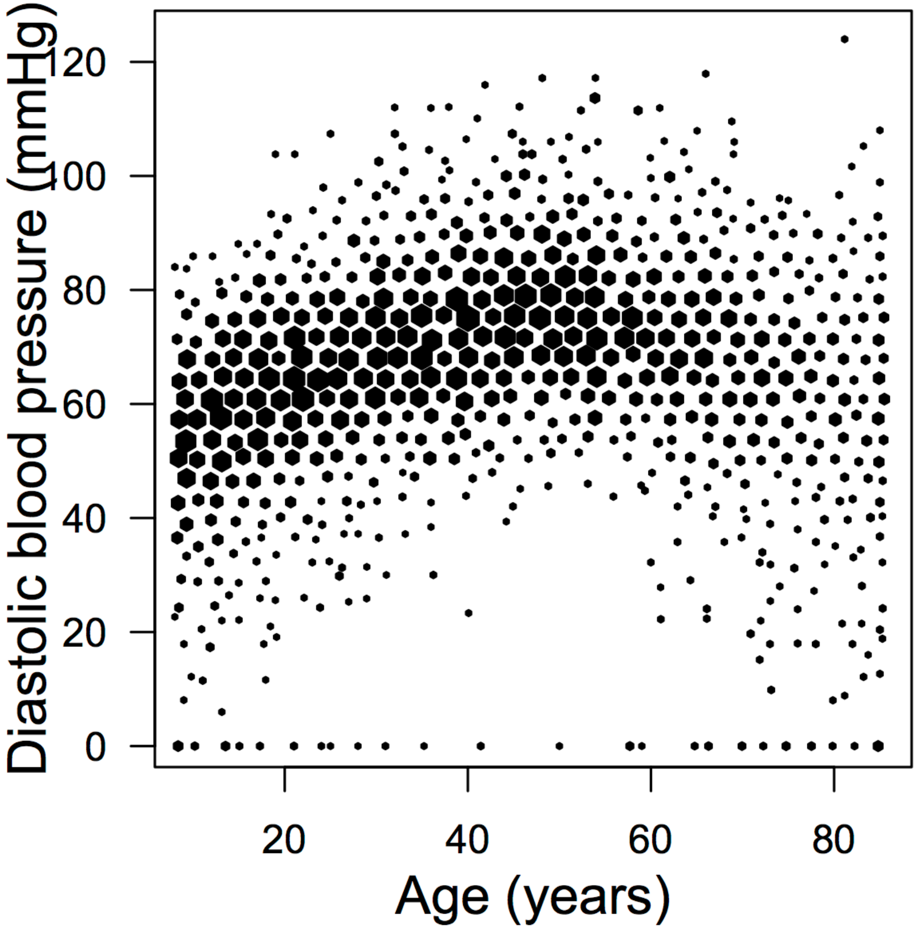


Hexagonal binning

- Divide plotting area into hexagons
- Collapse all the points in a grid cell into a small hexagon at the centre of mass
- Fast, gives small files, even for very large data
- Outliers are still visible

(Dan Carr, 1987)

NHANES, again



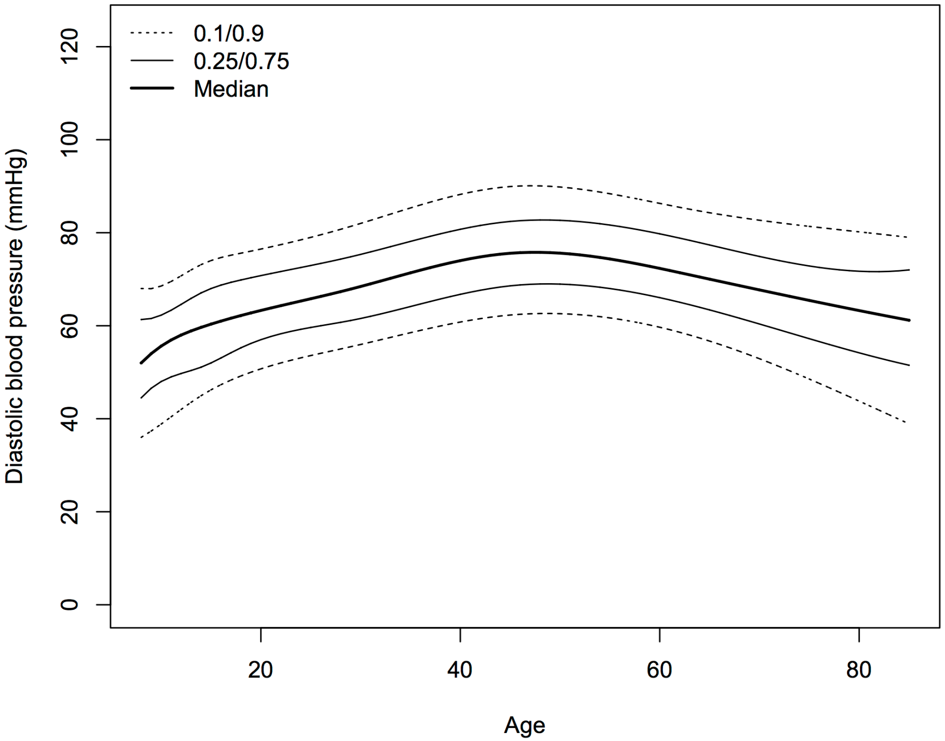
Scatterplot smoothers

We only need the curve, not a standard error estimate, so this is easy

- For local linear/polynomial regression smoothers (eg loess) just add weights to local regressions
- For quantile smoothers,
  - *use regression splines and weights in a quantile regression*

Both are done by `svysmooth()` in R: `method="quantreg"` or `method="locpoly"`

NHANES blood pressure trends



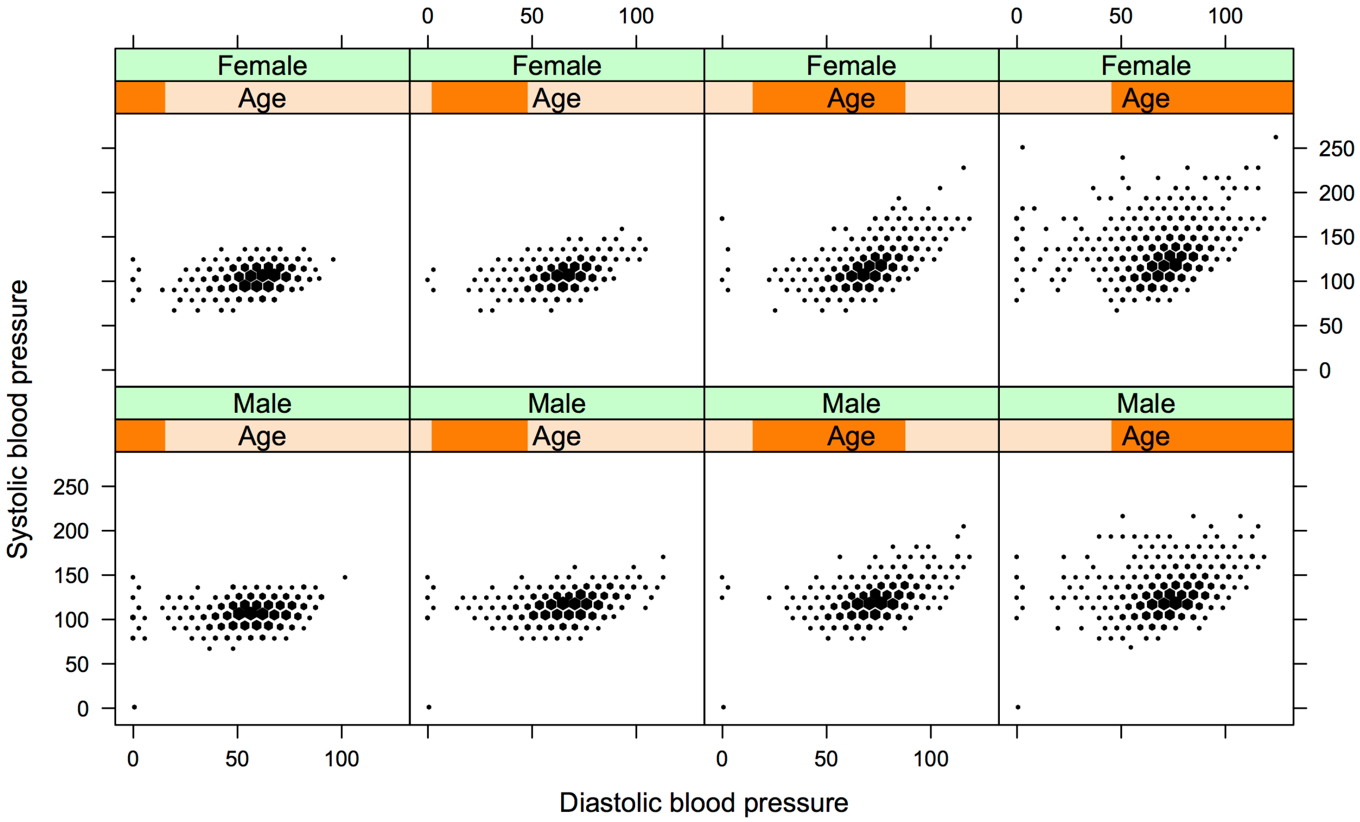
Conditioning plots

Show relationships in more than two dimensions by plotting Y~X conditioned on a range of Z

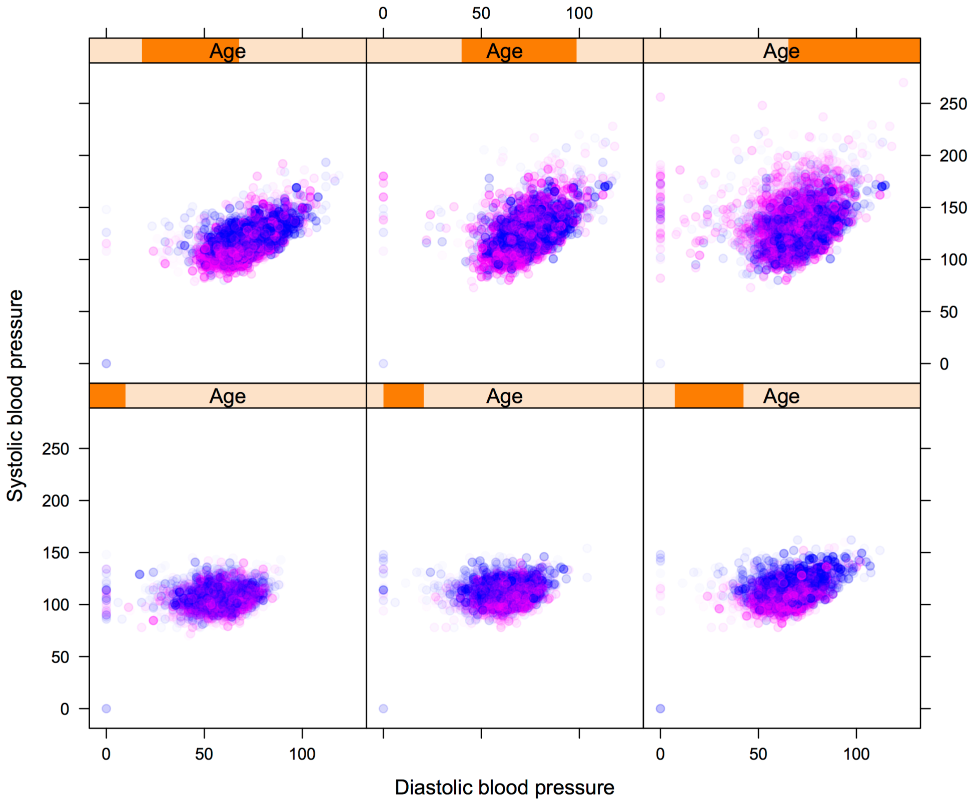
- “Trellis” graphics, invented by Bill Cleveland
- Implemented in R “lattice” package
- survey versions in survey package: transparent or hexbin

```
svycoplot(y~x|z, style="transparent")
```

Blood pressure, age, and sex



Blood pressure, age, and sex


Age
Age
Age
Age
Age
Age
Systolic blood pressure
Diastolic blood pressure