Analysing Multistage Survey Data Using R

Thomas Lumley
github/tslumley/svy-NY

University of Auckland (and R-core)

2020/2/11

Why you should care

Free Data!

- Health: NHANES, NHIS, CHIS
- Employment: CPS
- Attitudes: GSS, NLSY
- Everything: ACS

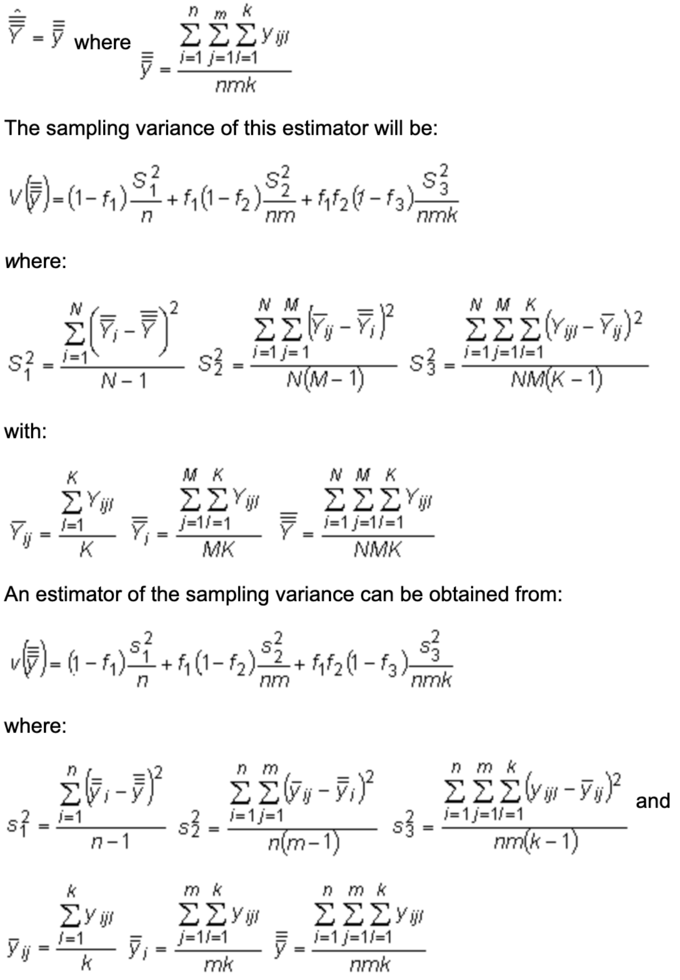
Or, you might be designing your own data collection

Scary warnings

> Standard statistical software packages generally do not take into account four common characteristics of sample survey data: (1) unequal probability selection of observations, (2) clustering of observations, (3) stratification and (4) nonresponse and other adjustments. Point estimates of population parameters are impacted by the value of the analysis weight for each observation. These weights depend upon the selection probabilities and other survey design features such as stratification and clustering. Hence, **standard packages will yield biased point estimates** if the weights are ignored. Estimated variance formulas for point estimates based on sample survey data are impacted by clustering, stratification and the weights. By ignoring these aspects, **standard packages generally underestimate the estimated variance of a point estimate, sometimes substantially so.** (Brogan, D, 1998, in *Encyclopedia of Biostatistics*)

😰😰😰😰

Typical presentation



$\hat{\bar{\bar{\bar{Y}}}} = \bar{\bar{\bar{y}}}$ where $\bar{\bar{\bar{y}}} = \dfrac{\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{l=1}^{k} y_{ijl}}{nmk}$

The sampling variance of this estimator will be:

$$V\left(\bar{\bar{\bar{y}}}\right) = (1-f_1)\frac{S_1^2}{n} + f_1(1-f_2)\frac{S_2^2}{nm} + f_1 f_2 (1-f_3)\frac{S_3^2}{nmk}$$

*w*here:

$$S_1^2 = \frac{\sum_{i=1}^{N}\left(\bar{\bar{Y}}_i - \bar{\bar{\bar{Y}}}\right)^2}{N-1} \quad S_2^2 = \frac{\sum_{i=1}^{N}\sum_{j=1}^{M}\left(\bar{Y}_{ij} - \bar{\bar{Y}}_i\right)^2}{N(M-1)} \quad S_3^2 = \frac{\sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{l=1}^{K}\left(Y_{ijl} - \bar{Y}_{ij}\right)^2}{NM(K-1)}$$

with:

$$\bar{Y}_{ij} = \frac{\sum_{l=1}^{K} Y_{ijl}}{K} \quad \bar{\bar{Y}}_i = \frac{\sum_{j=1}^{M}\sum_{l=1}^{K} Y_{ijl}}{MK} \quad \bar{\bar{\bar{Y}}} = \frac{\sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{l=1}^{K} Y_{ijl}}{NMK}$$

An estimator of the sampling variance can be obtained from:

$$v\left(\bar{\bar{\bar{y}}}\right) = (1-f_1)\frac{s_1^2}{n} + f_1(1-f_2)\frac{s_2^2}{nm} + f_1 f_2 (1-f_3)\frac{s_3^2}{nmk}$$

where:

$$s_1^2 = \frac{\sum_{i=1}^{n}\left(\bar{\bar{y}}_i - \bar{\bar{\bar{y}}}\right)^2}{n-1} \quad s_2^2 = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m}\left(\bar{y}_{ij} - \bar{\bar{y}}_i\right)^2}{n(m-1)} \quad s_3^2 = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{l=1}^{k}\left(y_{ijl} - \bar{y}_{ij}\right)^2}{nm(k-1)} \text{ and}$$

$$\bar{y}_{ij} = \frac{\sum_{l=1}^{k} y_{ijl}}{k} \quad \bar{\bar{y}}_i = \frac{\sum_{j=1}^{m}\sum_{l=1}^{k} y_{ijl}}{mk} \quad \bar{\bar{\bar{y}}} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{l=1}^{k} y_{ijl}}{nmk}$$

R to the rescue

R[1] lets you do most of the analyses you're used to with complex survey data.

You need to know three things

1. Weights
2. Clusters
3. Strata

1: and Stata

Weights

A weight of 100 means *this observation represents 100 people in the population*

- perhaps sampled with probability $1/100$
- but also non-response adjustment
- and frame corrections
- sometimes household size
- maybe telephones per household
- and calibration to Census data

Sometimes we oversample (and have smaller weights) because a group is interesting; sometimes because they are inexpensive; sometimes it just happens.

Clusters

If you have to go visit people, it's easier to do it in groups

- cities or counties
- neighbourhoods
- workplaces
- schools
- households

You get less information per observation, but more information per hour or per dollar.

Strata

If you sample 1000 people from the US, you get

- 10 **on average** from Iowa
- 2 **on average** from DC

You might fix these numbers, sample

- exactly 10 from Iowa
- exacty 2 from DC

These are strata.

Stratified sampling makes the sample more representative of the population; increases information per observation.

Multistage sampling

Sample clusters within strata, with specified weights

Within each cluster, sample subclusters within substrata with specified weights

And so on.

If the sample is much smaller than the population, you can pretend you had just one stage of cluster/strata (but all the weights)

Survey statistics calls this approximation **with replacement**, for boring maths reasons.

Nearly all public-use data uses the with-replacement approximation.

The survey package in 1 slide

Put your data and metadata in a survey design object.

```
data(nhanes)
d_nhanes <- svydesign(id=~SDMVPSU, strata=~SDMVSTRA, weights=~WTMEC2YR,
           nest=TRUE,data=nhanes)
```

Use the survey design object where you'd normally use a data frame, in a function that probably has a `svy` prefix

```
svymean(~HI_CHOL, design=d_nhanes, na.rm=TRUE)
svyby(~HI_CHOL, ~agecat, svymean, design=d_nhanes, na.rm=TRUE)
svyglm(HI_CHOL~race+agecat+RIAGENDR, design=d_nhanes, family=quasibinomial)
```

Data exploration

Putting weights in graphics

- boxplot: sample quartiles -> estimated population quartiles
- histogram: sample proportions -> estimated population proportions
- density: sum of kernels -> weighted sum of kernels
- scatterplots: ???

Scatterplots

- thinning: subsample the data to equal probability
- alpha channel: opacity proportional to weight
- hexbins: sample count -> estimated population count

`svyplot(,type="??")`

- "bubble": don't do this
- "hex": hexbins using hex size
- "grayhex": hexbins using shading
- "subsample": thinning
- "transparent": partially transparent

Sodium intake in NHANES

National Health and Nutrition Examination Survey

- continuous health survey
- samples about 15k people in each two-year wave
- includes detailed medical exam, so few clusters
- highly stratified sampling
- public use data!
- actually four-phase design, but public use files use 'with-replacement' approximation

We will look at dietary sodium intake and blood pressure.

Set up data

```r
library(foreign)

demo<-read.xport("demo_c.xpt")[,c(1:8,28:31)]
bp<-read.xport("bpx_c.xpt")
bm<-read.xport("bmx_c.xpt")[,c("SEQN","BMXBMI")]
diet<-read.xport("dr1tot_c.xpt")[,c(1:52,63,64)]

nhanes34<-merge(demo,bp,by="SEQN")
nhanes34<-merge(nhanes34,bm,by="SEQN")
nhanes34<-merge(nhanes34,diet,by="SEQN")

demo5<-read.xport("demo_d.xpt")[,c(1:8,39:42)]
bp5<-read.xport("bpx_x.xpt")
bp5$BPXSAR<-rowMeans(bp5[,c("BPXSY1","BPXSY2","BPXSY3","BPXSY4")],
    na.rm=TRUE)
bp5$BPXDAR<-rowMeans(bp5[,c("BPXDI1","BPXDI2","BPXDI3","BPXDI4")],
    na.rm=TRUE)
bm5<-read.xport("bmx_d.xpt")[,c("SEQN","BMXBMI")]
diet5<-read.xport("dr1tot_d.xpt")[,c(1:52,64,65)]

nhanes56<-merge(demo5,bp5,by="SEQN")
nhanes56<-merge(nhanes56,bm5,by="SEQN")
nhanes56<-merge(nhanes56,diet5,by="SEQN")

nhanes<-rbind(nhanes34,nhanes56)
```

Survey stuff

Weights

We're using *dietary interview* data, need to use dietary interview weights `WTDRD1`

Each wave has weights that rescale the two-year dietary interview sample to the US [civilian, non-institutionalised] population

We have two waves: need to scale each weight down by half.

```
nhanes$fouryearwt<-nhanes$WTDRD1/2
```

People who aren't in the dietary interview sample don't exist. They are dead to us.

Survey stuff

Structure

The documentation tells us the strata and cluster ('PSU') variables

`nest=TRUE` tells R to just assume the clusters are properly nested in the strata and the cluster `1` in each stratum is different.

```
des<-svydesign(id=~SDMVPSU,strat=~SDMVSTRA,weights=~fouryearwt,
   nest=TRUE, data=subset(nhanes, !is.na(WTDRD1)))
```

```
des
```

```
## Stratified 1 - level Cluster Sampling design (with replacement)
## With (60) clusters.
## svydesign(id = ~SDMVPSU, strat = ~SDMVSTRA, weights = ~fouryearwt,
##     nest = TRUE, data = subset(nhanes, !is.na(WTDRD1)))
```

```
colnames(des)
```

```
##  [1] "SEQN"       "SDDSRVYR"   "RIDSTATR"   "RIAGENDR"   "RIDAGEYR"
##  [6] "RIDAGEMN"   "RIDAGEEX"   "RIDRETH1"   "WTINT2YR"   "WTMEC2YR"
## [11] "SDMVPSU"    "SDMVSTRA"   "PEASCST1"   "PEASCTM1"   "PEASCCT1"
## [16] "BPXCHR"     "BPQ150A"    "BPQ150B"    "BPQ150C"    "BPQ150D"
## [21] "BPAARM"     "BPACSZ"     "BPXPLS"     "BPXDB"      "BPXPULS"
## [26] "BPXPTY"     "BPXML1"     "BPXSY1"     "BPXDI1"     "BPAEN1"
## [31] "BPXSY2"     "BPXDI2"     "BPAEN2"     "BPXSY3"     "BPXDI3"
## [36] "BPAEN3"     "BPXSY4"     "BPXDI4"     "BPAEN4"     "BPXSAR"
## [41] "BPXDAR"     "BMXBMI"     "WTDRD1"     "WTDR2D"     "DR1DRSTZ"
## [46] "DR1EXMER"   "DRABF"      "DRDINT"     "DR1DAY"     "DR1LANG"
## [51] "DR1MNRSP"   "DR1HELPD"   "DBQ095Z"    "DBD100"     "DRQSPREP"
## [56] "DRQSDIET"   "DRQSDT1"    "DRQSDT2"    "DRQSDT3"    "DRQSDT4"
## [61] "DRQSDT5"    "DRQSDT6"    "DRQSDT7"    "DRQSDT8"    "DRQSDT91"
## [66] "DR1TNUMF"   "DR1TKCAL"   "DR1TPROT"   "DR1TCARB"   "DR1TSUGR"
## [71] "DR1TFIBE"   "DR1TTFAT"   "DR1TSFAT"   "DR1TMFAT"   "DR1TPFAT"
## [76] "DR1TCHOL"   "DR1TATOC"   "DR1TATOA"   "DR1TRET"    "DR1TVARA"
## [81] "DR1TACAR"   "DR1TBCAR"   "DR1TCRYP"   "DR1TLYCO"   "DR1TLZ"
## [86] "DR1TVB1"    "DR1TVB2"    "DR1TNIAC"   "DR1TVB6"    "DR1TFOLA"
## [91] "DR1TFA"     "DR1TFF"     "DR1TFDFE"   "DR1TSODI"   "DR1TPOTA"
## [96] "fouryearwt"
```

Units

Dietary sodium/potassium a are large numbers in mg/day.

I want grams/day. Or moles, like normal countries use.

```
des<-update(des, sodium=DR1TSODI/1000, potassium=DR1TPOTA/1000)
des<-update(des, namol=sodium/23, kmol=potassium/39)
```

Summaries

```r
svymean(~sodium+potassium, des, na.rm=TRUE)
```

```
##               mean     SE
## sodium    3.3691 0.0280
## potassium 2.5979 0.0245
```

```r
svyquantile(~sodium+potassium, des, quantiles=c(0.25,0.5,0.75),na.rm=TRUE)
```

```
##            0.25   0.5     0.75
## sodium    2.115 3.069 4.243766
## potassium 1.703 2.392 3.254000
```

```r
svymean(~sodium, subset(des, RIAGENDR==1), na.rm=TRUE)
```

```
##          mean     SE
## sodium 3.8944 0.0373
```

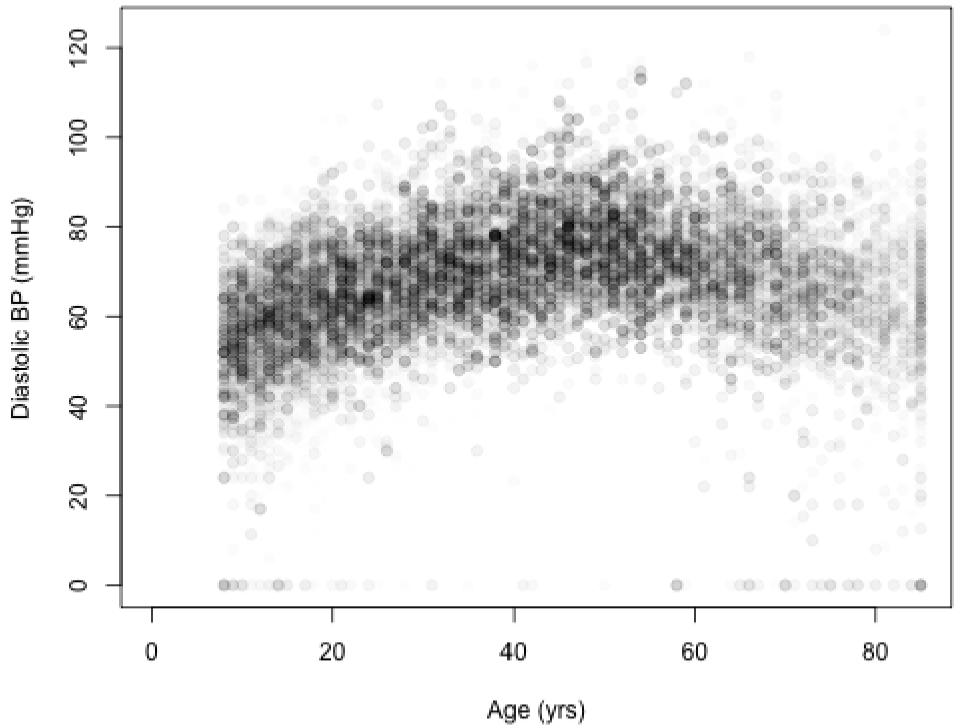
```r
svyby(~sodium, ~RIDRETH1, svymean, design=des, na.rm=TRUE)
```

```
##   RIDRETH1   sodium         se
## 1        1 3.188710 0.05530042
## 2        2 2.986853 0.08027165
## 3        3 3.445924 0.03587407
## 4        4 3.189214 0.04762457
## 5        5 3.321049 0.08583119
```

Scatterplots

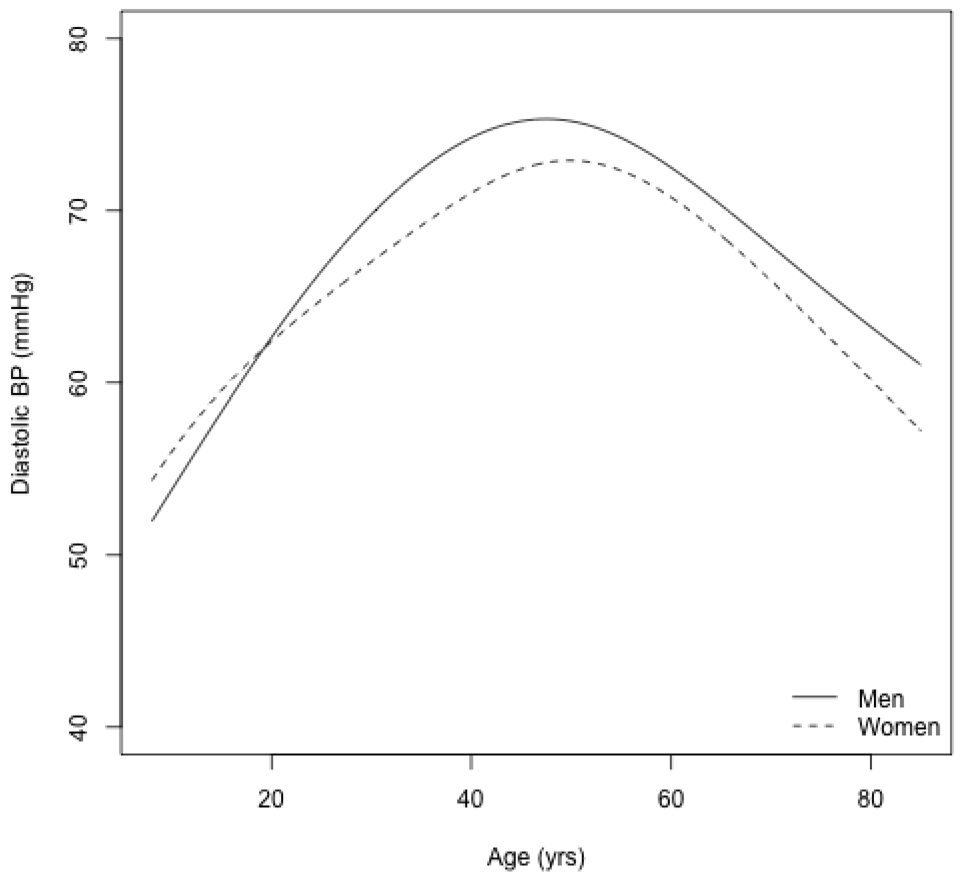
```
svyplot(BPXDAR~RIDAGEYR,style="hex",design=des,legend=0,
        xlab="Age (yrs)",ylab="Diastolic BP (mmHg)")
```

```
svyplot(BPXDAR~RIDAGEYR,style="trans",design=des,legend=0,
        xlab="Age (yrs)",ylab="Diastolic BP (mmHg)", pch=19,alpha=c(0,0.3))
```

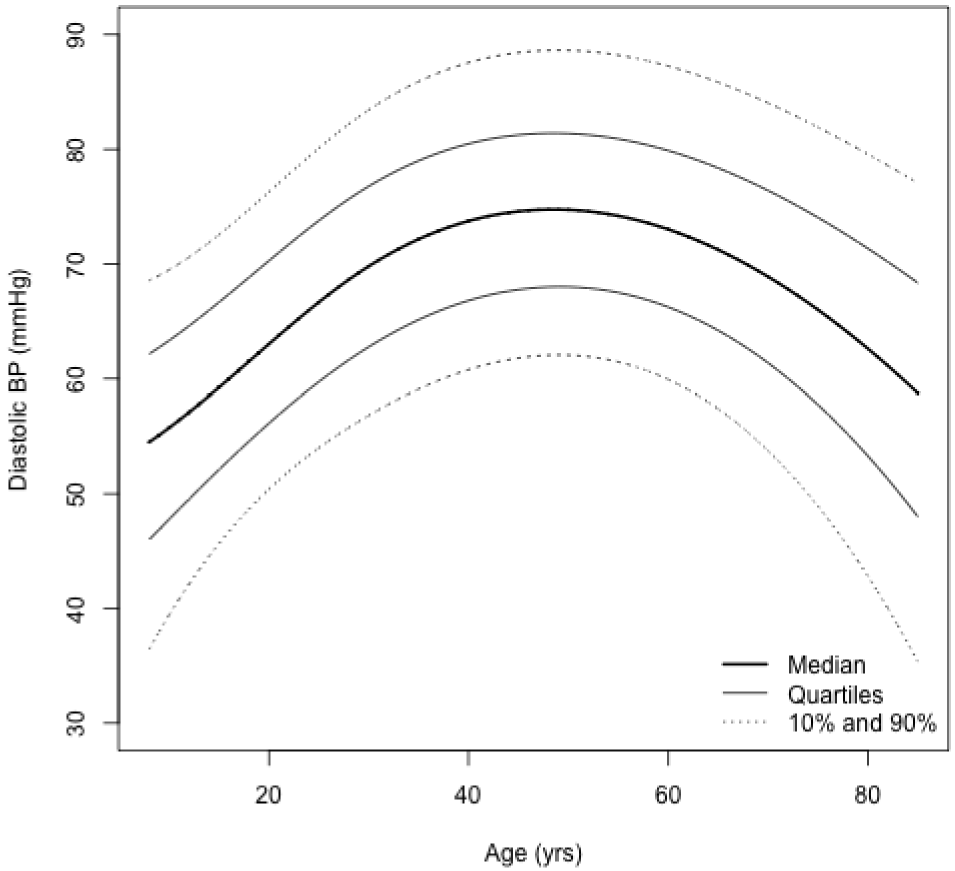


Smoothers

```r
men<-svysmooth(BPXDAR~RIDAGEYR,design=subset(des, RIAGENDR==1),bandwidth=10)
women<-svysmooth(BPXDAR~RIDAGEYR,design=subset(des, RIAGENDR==2),bandwidth=10)
plot(men,ylim=c(40,80),ylab="Diastolic BP (mmHg)",xlab="Age (yrs)")
lines(women,lty=2)
legend("bottomright",lty=1:2,bty="n",legend=c("Men","Women"))
```


Diastolic BP (mmHg)
Age (yrs)
80
70
60
50
40
20
40
60
80
Men
Women

```r
median<-svysmooth(BPXDAR~RIDAGEYR,design=des,method="quantreg",quantile=0.5)
plot(median,ylim=c(30,90),ylab="Diastolic BP (mmHg)",xlab="Age (yrs)",lwd=2)
for(qi in c(.25,.75)){
  quartile <- svysmooth(BPXDAR~RIDAGEYR,design=des,
                          method="quantreg",quantile=qi)
    lines(quartile,lwd=1)
}
for(qi in c(.1,.9)){
    decile <- svysmooth(BPXDAR~RIDAGEYR,design=des,
                        method="quantreg",quantile=qi)
    lines(decile,lwd=1,lty=3)
}
legend("bottomright",lty=c(1,1,3),lwd=c(2,1,1),bty="n",
       legend=c("Median","Quartiles","10% and 90%"))
```


Diastolic BP (mmHg)
90
80
70
60
50
40
30
20
40
60
80
Age (yrs)
Median
Quartiles
10% and 90%

Regression: ISH

We'll use **isolated systolic hypertension** as the outcome variable

```
des<-update(des, ish=(BPXSAR>140) & (BPXDAR<90))
```

Also, linear splines in age, because linear is *obviously* not sound

```
des<-update(des,
            age1=pmin(RIDAGEYR,50)/10,
            age2=pmin(pmax(RIDAGEYR,50),65)/10,
            age3=pmin(pmax(RIDAGEYR,65),90)/10)
```

Sequence of models

```
ish0s <- svyglm(ish~age1+age2+age3,
  design=des, family=quasibinomial)
ish1s<- svyglm(ish~age1+age2+age3+factor(RIDRETH1),
  design=des,family=quasibinomial)
ish2s<- svyglm(ish~age1+age2+age3+RIAGENDR+factor(RIDRETH1),
  design=des, family=quasibinomial)
ish3s<- svyglm(ish~(age1+age2+age3)*RIAGENDR+factor(RIDRETH1),
  design=des,family=quasibinomial)
ish4s<- svyglm(ish~(age1+age2+age3)*RIAGENDR+factor(RIDRETH1)+sodium,
  design=des,family=quasibinomial)
```

```r
anova(ish3s)
```

```
## Anova table:  (Rao-Scott LRT)
## svyglm(formula = ish ~ age1, design = des, family = quasibinomial)
##                      stats     DEff        df ddf          p
## age1             1820.4005   6.1408    1.0000  29 < 2.2e-16 ***
## age2              462.2813   3.4693    1.0000  28 4.166e-12 ***
## age3               40.0039   1.7249    1.0000  27 5.440e-05 ***
## RIAGENDR           12.6366   1.6384    1.0000  26   0.01059 *
## factor(RIDRETH1)   30.8047   2.1703    4.0000  22   0.02800 *
## age1:RIAGENDR     110.5041   2.9146    1.0000  21 4.561e-06 ***
## age2:RIAGENDR       1.5871   2.8334    1.0000  20   0.46210
## age3:RIAGENDR       0.7468   1.6674    1.0000  19   0.50951
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova(ish3s,ish4s)
```

```
## Working (Rao-Scott+F) LRT for sodium
##  in svyglm(formula = ish ~ (age1 + age2 + age3) * RIAGENDR + factor(RIDRETH1) +
##     sodium, design = des, family = quasibinomial)
## Working 2logLR =  0.7482066 p= 0.39902
## df=1;  denominator df= 18
```

No real evidence that sodium matters?

Try some more variables

```
ish5<- svyglm(ish~(age1+age2+age3)*RIAGENDR+factor(RIDRETH1)+BMXBMI,
  design=des,family=quasibinomial)
ish6<- svyglm(ish~(age1+age2+age3)*RIAGENDR+factor(RIDRETH1)+BMXBMI+sodium+potassium,
  design=des,family=quasibinomial)
anova(ish5, ish6)
```

```
## Working (Rao-Scott+F) LRT for sodium potassium
##  in svyglm(formula = ish ~ (age1 + age2 + age3) * RIAGENDR + factor(RIDRETH1) +
##     BMXBMI + sodium + potassium, design = des, family = quasibinomial)
## Working 2logLR =  1.505635 p= 0.46569
## (scale factors:  1.4 0.62 );  denominator df= 16
```

Nonlinearity?

```
orary(splines)
n7<- svyglm(ish~(age1+age2+age3)*RIAGENDR+factor(RIDRETH1)+BMXBMI+ns(sodium,3)+potassium,
design=des,family=quasibinomial)
ova(ish5, ish7)
```

```
## Working (Rao-Scott+F) LRT for ns(sodium, 3) potassium
##  in svyglm(formula = ish ~ (age1 + age2 + age3) * RIAGENDR + factor(RIDRETH1) +
##     BMXBMI + ns(sodium, 3) + potassium, design = des, family = quasibinomial)
## Working 2logLR =  5.378236 p= 0.28088
## (scale factors:  1.9 0.79 0.73 0.56 );  denominator df= 14
```

Continuous outcome

```r
sysbp<- svyglm(BPXSAR~(age1+age2+age3)*RIAGENDR+factor(RIDRETH1)+BMXBMI,
  design=des)
sysbp_sodium<- svyglm(BPXSAR~(age1+age2+age3)*RIAGENDR+factor(RIDRETH1)+BMXBMI+sodium+pot
  design=des)
anova(sysbp, sysbp_sodium)
```

```
## Working (Rao-Scott+F) LRT for sodium potassium
##  in svyglm(formula = BPXSAR ~ (age1 + age2 + age3) * RIAGENDR + factor(RIDRETH1) +
##     BMXBMI + sodium + potassium, design = des)
## Working 2logLR =  12.53263 p= 0.011866
## (scale factors:  1.3 0.66 );  denominator df= 16
```

```r
coef(summary(sysbp_sodium))
```

```
##                        Estimate  Std. Error    t value     Pr(>|t|)
## (Intercept)         116.8097161 11.10084650 10.5225954 1.345988e-08
## age1                  3.6083977  0.48489800  7.4415602 1.399511e-06
## age2                 -4.6559457  1.69526808 -2.7464362 1.433872e-02
## age3                  1.4125794  2.33057584  0.6061075 5.529459e-01
## RIAGENDR            -60.4325610  7.25752800 -8.3268795 3.285723e-07
## factor(RIDRETH1)2     0.4066314  1.32225042  0.3075298 7.624070e-01
## factor(RIDRETH1)3    -0.4055510  0.62490182 -0.6489836 5.255558e-01
## factor(RIDRETH1)4     3.1716349  0.64006450  4.9551801 1.432652e-04
## factor(RIDRETH1)5     1.4484140  0.90467554  1.6010315 1.289267e-01
## BMXBMI                0.4171254  0.04015294 10.3884130 1.612343e-08
## sodium                0.3629780  0.15851769  2.2898268 3.595705e-02
## potassium            -0.7409521  0.17089288 -4.3357690 5.111297e-04
## age1:RIAGENDR         0.2239508  0.28565533  0.7839896 4.444983e-01
## age2:RIAGENDR         6.6738476  1.13463416  5.8819378 2.317427e-05
## age3:RIAGENDR         3.1239664  1.59087787  1.9636746 6.718831e-02
```

Ignoring survey design

```
wrong_sodium<-glm(BPXSAR~(age1+age2+age3)*RIAGENDR+factor(RIDRETH1)+BMXBMI+sodium+potassi
  data=model.frame(des))
summary(coef(wrong_sodium)/coef(sysbp_sodium))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -1.6392  0.6410  0.9691  0.8128  1.1020  3.1855
```

```
summary(SE(wrong_sodium)/SE(sysbp_sodium))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.5116  0.5559  0.5765  0.6228  0.7236  0.7769
```

Other things we can do

- `svyolr`: ordinal logistic regression and similar
- `svykm`, `svycoxph`: survival analysis
- `svyloglin`: loglinear models
- `svyranktest`: design-based rank tests, *if you must*
- `AIC`, `BIC`: for `svyglm` only, so far, design-based versions of AIC and BIC
- `svrepdesign`: designs specificed by replicate weights
- `as.svrepdesign`: creating replicate weights (cf bootstrap/jackknife)
- `postStratify`, `rake`, `calibrate`: rescale weights to match known population information.

Two-phase sampling

phase: sampling at phase II can depend on phase I data

Technical difference: we only know sampling probability conditional on **this phase-I sample**

Doesn't affect most of the formulas.

Examples

- trivially: case-control sampling: Y known at phase I, X sampled at phase II
- case-cohort: random subset plus all the cases
- joint sampling based on (surrogates for) outcome and exposure

Example: National Wilms' Tumour Studies: validation of histology assessment

Lots of opportunities to think about optimal design!

```r
data(nwtco)
nwtco$in_cc2<-as.logical(with(nwtco, ifelse(rel | instit==2,1,rbinom(nrow(nwtco),1,.1))))
{{dccs8<-twophase(id=list(~seqno,~seqno),
                strata=list(NULL,~interaction(rel,stage,instit)),
                data=nwtco, subset=~in_cc2)}}

svyglm(rel~factor(stage)*factor(histol),
       design=dccs8,family=quasibinomial())
```

```
## Two-phase sparse-matrix design:
##  twophase2(id = id, strata = strata, probs = probs, fpc = fpc,
##     subset = subset, data = data)
## Phase 1:
## Independent Sampling design (with replacement)
## svydesign(ids = ~seqno)
## Phase 2:
## Stratified Independent Sampling design
## svydesign(ids = ~seqno, strata = ~interaction(rel, stage, instit),
##     fpc = `*phase1*`)
##
## Call:  svyglm(formula = rel ~ factor(stage) * factor(histol), design = dccs8,
##     family = quasibinomial())
##
## Coefficients:
##                  (Intercept)                factor(stage)2
##                      -2.6781                        0.7392
##               factor(stage)3                factor(stage)4
##                       0.7633                        1.0101
##              factor(histol)2  factor(stage)2:factor(histol)2
##                       1.1457                        0.5686
## factor(stage)3:factor(histol)2  factor(stage)4:factor(histol)2
##                       0.7959                        1.8694
##
## Degrees of Freedom: 1119 Total (i.e. Null);  1097 Residual
## Null Deviance:       914.2
## Residual Deviance: 806.1     AIC: NA
```

More accurately

Sampling was actually stratified just on outcome, then post-stratified

```r
dccs2<-twophase(id=list(~seqno,~seqno),strata=list(NULL,~interaction(rel,instit)),
    data=nwtco, subset=~in_cc2)
gccs8<-calibrate(dccs2, phase=2, formula=~interaction(rel,stage,instit))
svyglm(rel~factor(stage)*factor(histol), design=gccs8,family=quasibinomial())
```

```
## Two-phase sparse-matrix design:
##  gccs8<-calibrate(dccs2, phase=2, formula=~interaction(rel,stage,instit))
## Phase 1:
## Independent Sampling design (with replacement)
## svydesign(ids = ~seqno)
## Phase 2:
## Stratified Independent Sampling design
## calibrate(phase2, formula, population, calfun = calfun, ...)
##
## Call:  svyglm(formula = rel ~ factor(stage) * factor(histol), design = gccs8,
##     family = quasibinomial())
##
## Coefficients:
##                    (Intercept)                  factor(stage)2
##                        -2.6781                          0.7392
##                 factor(stage)3                  factor(stage)4
##                         0.7633                          1.0101
##                factor(histol)2  factor(stage)2:factor(histol)2
##                         1.1457                          0.5686
## factor(stage)3:factor(histol)2  factor(stage)4:factor(histol)2
##                         0.7959                          1.8694
##
## Degrees of Freedom: 1119 Total (i.e. Null);  1109 Residual
## Null Deviance:       914.2
## Residual Deviance: 806.1     AIC: NA
```

Adding features

Ask

Ideally, with

- a paper describing the estimator exactly
- a test data set

I can try to just invent things (AIC, BIC, rank tests), but it takes longer

Note: mixed models are **hard** for non-obvious reasons. I'm working on `svylme` but it's not really done yet.

Where to find things

- CRAN `survey`
- http://r-forge.r-project.org/projects/r-survey/
- `@tslumley`
- https://notstatschat.rbind.io
- github/tslumley/svylme
- **github/tslumley/svy-NY**