

TUTORIAL: SAMPLING, WEIGHTING AND ESTIMATION DAY 4

Stefan Zins, Matthias Sand
and Jan-Philipp Kolb

GESIS - Leibniz Institute
for the Social Sciences

GESIS Summer School

Cologne, Germany

August 27th, 2015

Computation of contrasts

- Using the multi-stage sample of day 3 (`mul.surv`)

```
means <-svymean(~api00+api99,mul.surv)
means
```

```
      mean    SE
api00  644 11.4
api99  613 13.2
```

```
svycontrast(means,quote(api00-api99))
```

```
      nlcon    SE
contrast  30.6 3.19
```

- The command `svycontrast()` can be used to estimate contrasts for linear and nonlinear survey statistics and their standard error

```
svycontrast(means,quote(api00/api99))
```

```
      nlcon    SE
contrast  1.05 0.01
```

⇒ `svyratio(~0 api00,~0 api99,mul.surv)` generates the same results

Weighted regression

```
summary(svyglm(api00~enroll+meals+api99,mul.surv))
```

Call:

```
svyglm(formula = api00 ~ enroll + meals + api99, mul.surv)
```

Survey design:

```
svydesign(id = ~cds + id, fpc = ~fpc + fpc2, strata = NULL, data = DATA.s,  
  pps = "brewer")
```

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	114.31730	38.64877	2.96	0.0049 **
enroll	-0.00535	0.00472	-1.13	0.2623
meals	0.06558	0.18715	0.35	0.7277
api99	0.86531	0.04505	19.21	<2e-16 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 406.9)

Number of Fisher Scoring iterations: 2

Weighted regression

```
summary(svyglm(api00~enroll+meals+api99,mul.surv))
```

- The `svyglm()` function fits linear and generalized lin. models to the data stored in a survey object
- syntax almost identical to the `glm`, except that the data argument is replaced by a design argument → weighted least square
- Main difference to `glm()`: `svyglm()` does not use a maximum likelihood approach

Survey weighted contingency tables and chi squared tests

```
tab1 <- svytable(~stype+sch.wide,mul.surv)
tab1
```

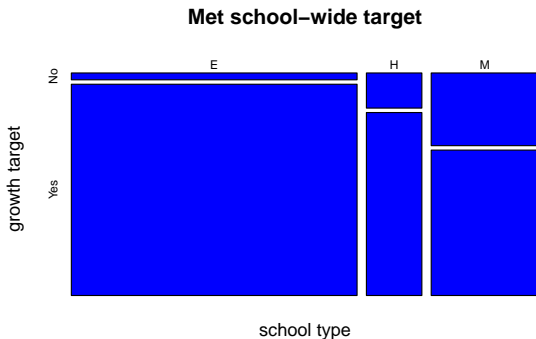
```
      sch.wide
stype   No   Yes
  E  2003 62108
  H  2003 10441
  M  8014 16028
```

- Creates the weighted contingency table of two or more variables

```
plot(tab1, col="blue",xlab = "school type",
+      ylab = "growth target",cex.lab=0.25)
```

- Such contingency tables can easily be visualized to learn more about the sampled variables

Survey weighted contingency tables and chi squared tests



Survey weighted contingency tables and chi squared tests

```
chi <- svychisq(~stype+sch.wide,mul.surv,statistic="adjWald")  
chi
```

Design-based Wald test of association

```
data: svychisq(~stype + sch.wide, mul.surv, statistic = "adjWald")  
F = 4.071, ndf = 2, ddf = 48, p-value = 0.02327
```

- Tests for the association of sample variables
- Numerous tests are applicable
- In this case it is tested for independence under the consideration of the number of PSUs

Estimation of subpopulations

- Within a stratified random sample, subtotals for each stratum are easy to compute, since every stratum can be treated like a separate srs

Estimation of subpopulations

- Within a stratified random sample, subtotals for each stratum are easy to compute, since every stratum can be treated like a separate srs
- For an unstratified design or subpopulations that are no strata, the situation is more complicated, since the joint inclusion probabilities π_{kl} are unequal to those that would be, if this group had been a stratum
- Sampling weights would be correct, but pairwise sampling probability would be incorrect

Estimation of subpopulations

- Within a stratified random sample, subtotals for each stratum are easy to compute, since every stratum can be treated like a separate srs
 - For an unstratified design or subpopulations that are no strata, the situation is more complicated, since the joint inclusion probabilities π_{kl} are unequal to those that would be, if this group had been a stratum
 - Sampling weights would be correct, but pairwise sampling probability would be incorrect
- ⇒ Unbiased point estimates; but standard errors would be wrong

Estimation of subpopulations

- The survey package deals with these problems without any further specification as long as the full sample is used to define a survey object
- Subpopulations either by `subset` or `svyby`

```
sub1 <- subset(mul.surv,sch.wide=="Yes")  
svymean(~api00+api99,sub1)
```

	mean	SE
api00	638	12.7
api99	602	14.4

```
sub2 <- svyby(~api00,~stype,svymean,design = mul.surv)  
print(sub2)
```

	stype	api00	se
E	E	645.1	16.69
H	H	619.9	11.10
M	M	653.6	16.26

The ratio estimator

$$\frac{\hat{t}_{y,HT}}{\hat{t}_{x,HT}}$$

- Drawing a sample of $n = 130$ from bm

```
bm$pik <- inclusionprobabilities(bm$Tot03,130)
s <- UPmaxentropy(bm$pik)
ratioest(y=bm$TaxableIncome[s==1],x=bm$averageincome[s==1],
+       Tx=sum(bm$averageincome),pik=bm$pik[s==1])

[1] 1.385e+11
```

- Returns the ratio estimator of the population total
- ⇒ Another way to calculate the ratio estimator is implemented in the survey package

The ratio estimator

```
samp <- bm[s==1,]  
IPk1 <- UPmaxentropyp2(bm$pik)  
obj <- svydesign(id=~1, fpc = ~pik, data=samp,  
+               pps = ppsmat(IPk1[s==1, s==1]), variance="YG")  
svyratio(~TaxableIncome, ~averageincome, obj)
```

Ratio estimator: `svyratio.pps(~TaxableIncome, ~averageincome, obj)`

Ratios=

averageincome

TaxableIncome 9370

SEs=

averageincome

TaxableIncome 474.5

- The `svyratio()` command returns the ratio between the two weighted sampling variables.

The ratio estimator

- To obtain an estimator for the total, the ratio must be multiplied with the population total of the auxiliary variable

```
as.numeric(svyratio(~TaxableIncome, ~averageincome, obj)) *  
+ sum(bm$averageincome)
```

```
[1] 1.385e+11 3.327e+12
```

Variance estimation

- Estimating the Variance of the ratio estimator via *Taylor Linearization* with the sampling package

```
vartaylor_ratio(ys=samp$TaxableIncome, xs=samp$averageincome,  
+               pikls = IPkl[s==1,s==1])
```

```
$ratio  
[1] 9370
```

```
$estvar  
[1] 234061
```

```
[1] 483.8
```

Variance estimation

- Estimating the Variance of the ratio estimator via *Taylor Linearization* with the sampling package

```
vartaylor_ratio(Ys=samp$TaxableIncome,Xs=samp$averageincome,  
+              pikls = IPkl[s==1,s==1])
```

```
$ratio  
[1] 9370
```

```
$estvar  
[1] 234061
```

```
[1] 483.8
```

- The survey package uses *Taylor Linearization* by default to estimate standard errors based on design information

Variance estimation

- Estimating the Variance of the ratio estimator via *Taylor Linearization* with the sampling package

```
vartaylor_ratio(ys=samp$TaxableIncome, xs=samp$averageincome,  
+              pikls = IPkl[s==1, s==1])
```

```
$ratio  
[1] 9370
```

```
$estvar  
[1] 234061
```

```
[1] 483.8
```

- The survey package uses *Taylor Linearization* by default to estimate standard errors based on design information
- The `vartaylor_ratio()` returns the HT-estimator of the variance, not the YG-type

- Within the survey package all of the *common* strategies are implemented and easy to use
- First step: redefine the survey object as a replicate weights object with the `as.svrepdesign()` command
- Second step: choosing a strategy with the `type=` argument

- Within the survey package all of the *common* strategies are implemented and easy to use
 - First step: redefine the survey object as a replicate weights object with the `as.svrepdesign()` command
 - Second step: choosing a strategy with the `type=` argument
- ⇒ Most of the strategies cannot create replicate weights under unequal inclusion probabilities and resample only the PSUs to create replicate weights

Jackknife

```
dclus1 <- svydesign(id=~dnum,weights=~pw,data=apiclus1,
+                 fpc= rep(757/15,times=nrow(apiclus1)))
clus.rep <- as.svrepdesign(dclus1,type = "JK1")
svyquantile(~api00 , clus.rep, c (.25 ,.5 ,.75) ,
+          interval.type ="probability")
```

Statistic:

```
      api00
q0.25 551.8
q0.5  652.0
q0.75 717.5
```

SE:

```
      api00
q0.25 30.34
q0.5  34.21
q0.75 15.83
```

- Allows the estimation of standard errors for quantiles, etc.
- In case of estimates for the totals and means, the estimates for the standard error are comparable to those of the Taylor Linearization

Bootstrap

```
rep.clus2 <- as.svrepdesign(dclus1,type = "bootstrap",
+                          replicates = 100)
svyquantile(~api00,rep.clus2,c (.25 ,.5 ,.75) ,
+          interval.type ="probability")
```

Statistic:

```
      api00
q0.25 551.8
q0.5  652.0
q0.75 717.5
SE:
```

```
      api00
q0.25 29.33
q0.5  30.90
q0.75 15.75
```

⇒ Reduces the standard error

- design weights are the starting weights multiplied by the times they have been sampled within each iteration (srswr!)

Balanced Repeated Replicates (BRR)

- BRR is appealing because it requires less computational effort than the Jackknife method
- BRR standard errors are valid for quantiles and median, while the Jackknife method might produce invalid results

```
dstrat <- svydesign(id=~1,weights=~pw,strata = ~stype,
+                 data = apistrat,fpc = ~fpc)
rep3 <- as.svrepdesign(dstrat,type = "BRR")
svyquantile(~api00,rep3,c (.25 ,.5 ,.75) ,
+           interval.type ="probability")
```

Statistic:

```
      api00
q0.25 562.2
q0.5  667.2
q0.75 755.1
```

SE:

```
      api00
q0.25 15.05
q0.5  11.33
q0.75 13.18
```

Summary: replicate weights and single stage sampling

- All three methods can only be applied to single-stage samples or resample only on the PSU-level
- All three approaches ignore the fpc

Summary: replicate weights and single stage sampling

- All three methods can only be applied to single-stage samples or resample only on the PSU-level
- All three approaches ignore the fpc
- Bootstrap is only straightforward, when all strata sizes are large

Summary: replicate weights and single stage sampling

- All three methods can only be applied to single-stage samples or resample only on the PSU-level
- All three approaches ignore the fpc
- Bootstrap is only straightforward, when all strata sizes are large
- The BRR approach yields the smallest estimates for the standard error, but has difficulties when dealing with cluster samples

Summary: replicate weights and single stage sampling

- All three methods can only be applied to single-stage samples or resample only on the PSU-level
 - All three approaches ignore the fpc
 - Bootstrap is only straightforward, when all strata sizes are large
 - The BRR approach yields the smallest estimates for the standard error, but has difficulties when dealing with cluster samples
- ⇒ BRR can only be applied for samples with two clusters per stratum

Summary: replicate weights and single stage sampling

- All three methods can only be applied to single-stage samples or resample only on the PSU-level
 - All three approaches ignore the fpc
 - Bootstrap is only straightforward, when all strata sizes are large
 - The BRR approach yields the smallest estimates for the standard error, but has difficulties when dealing with cluster samples
- ⇒ BRR can only be applied for samples with two clusters per stratum
- ⇒ All three approaches have problems with multi-stage samples and unequal inclusion probabilities

Multi-stage samples and replicate weights

- In case of multi-stage samples, the *mrbs* or *mrbootstrap* method of Preston should be employed

⇒ Resamples PSUs and SSUs rather than only PSUs

```
summary(svyglm(api00~enroll+meals+api99,mul.surv))
```

Call:

```
svyglm(formula = api00 ~ enroll + meals + api99, mul.surv)
```

Survey design:

```
svydesign(id = ~cds + id, fpc = ~fpc + fpc2, strata = NULL, data = DATA.s,  
  pps = "brewer")
```

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	114.31730	38.64877	2.96	0.0049 **
enroll	-0.00535	0.00472	-1.13	0.2623
meals	0.06558	0.18715	0.35	0.7277
api99	0.86531	0.04505	19.21	<2e-16 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 406.9)

Number of Fisher Scoring iterations: 2

Multi-stage samples and replicate weights

```
rep4 <- as.svrepdesign(mul.surv,type = "mrbootstrap")
summary(svyglm(api00~enroll+meals+api99,rep4))
```

Call:

```
svyglm(formula = api00 ~ enroll + meals + api99, rep4)
```

Survey design:

```
as.svrepdesign(mul.surv, type = "mrbootstrap")
```

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	114.31730	19.95635	5.73	7.4e-07 ***
enroll	-0.00535	0.00256	-2.09	0.042 *
meals	0.06558	0.09243	0.71	0.482
api99	0.86531	0.02384	36.30	< 2e-16 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 406.7)

Number of Fisher Scoring iterations: 2

- Reduces standard error of the estimates
 - Estimates are far more significant
 - Needs more computational time then other resampling strategies
- ⇒ But: correct approach for multi-stage samples