Exact, user-oriented and reproducible variance estimation with the gustave package

Martin Chevalier

EU-SILC Best Practices Workshop
Prague, 14-15 september 2017

Exact, user-oriented and reproducible variance estimation with the gustave package

Variance estimation is becoming a **more and more important part** of the survey production process:

- tool to assess the quality of the data collected that may affect its dissemination;
- key indicator in quality reports but also in the IESS regulation currently under discussion.

It is also a **complex issue** from a **technical** and **organizational** point of view.

This talk:

- presents the way variance estimation is implemented in France for household surveys (including EU-SILC)
- introduces the R gustave package

Outline

Purpose: make variance estimation easier

Live demonstration

How-to: gustave package starter's guide

Discussion and perspectives

Purpose: make variance estimation easier

Sources of complexity: sampling design

Sampling design is the first source of complexity for exact analytical variance estimation:

- complex **sampling algorithms**: sampling with unequal probabilities, balanced sampling
- **multi-stages** sampling: two-stages sampling, master sample

Remarks

- the "ultimate cluster" methodology is applicable only if the sampling rate of the first stage is negligible
- bootstrap methodology might not be applicable under such complex sampling designs

Purpose: make variance estimation easier

Sources of complexity: estimation methodology

In order to be exact, variance estimation should also take into account the estimation methodology implemented:

- **non-response correction**: the unit non-response process can be described as an additional Poisson sampling phase
- **calibration on margins**: calibration on margins yields significant variance reduction provided the auxiliary variables are well correlated with the variables of interest

Remark Depending on the imputation methodology, **item non-response** can also be taken into account in the variance estimation process.

Sources of complexity: indicators

A final source of complexity comes from the indicators whose variance is to be estimated:

- **non-linear indicators**: mean, ratio, quantiles, Gini coefficient, At-risk of poverty rate, etc. $\rightarrow$ **linearization techniques**
- **domain estimations**: regional estimates

Remark Variance estimation with small area estimation methodology is beyond the scope of this talk.

Purpose: make variance estimation easier

Organizational consequence: division of labour

- **Methodologist**: methodological expertise, production of variance estimation programs
- **Survey specialist**: topic-related expertise, use of the variance estimation programs for studies and quality reporting

New challenges: in order to be used in practice, variance estimation programs should be

- as easy-to-use as possible
- comprehensive (e.g. linearization, domain estimation)
- self-contained (reproducibility)
- not too difficult to maintain (for future upgrades)

Introducing the gustave package

Gustave: a **U**ser-oriented **S**tatistical **T**oolkit for **A**nalytical **V**ariance **E**stimation

R package currently available on GitHub:
`https://github.com/martinchevalier/gustave`

Goal Make variance estimation as easy as possible in providing the methodologist with dedicated tools.

Live demonstration

Live demonstration

Variance estimation in the French EU-SILC

The French EU-SILC:

- 9 years rotating panel: two master samples involved, each with balanced sampling at the first stage
- general weight share methodology
- non-negligible unit non-response (attrition) and complex non-response scheme
- calibration on margins
- highly non-linear indicators: At-risk of poverty rate, Gini coefficient, etc.

Once processed by the `gustave` package: **one single standalone data file per year**.

Live demonstration

First run

```
# Load the standalone .RData file
load("silc_bpw/precisionSilc14.RData")
# Note : the 4 data files of the survey are loaded
# together with the precisionSilc() function

# Mean equivalised disposable income per household
precisionSilc(h, HX090)
```

```
##                call      est variance      std       cv
## 1 mean(y = HX090) 24856.48 62032.16 249.0626 1.002002
##       lower    upper
## 1 24368.33 25344.64
```

```
# The variance estimation takes about 1 second.
```

Live demonstration

Standard use

```
# Share of households below a given (absolute) threshold
# (see below for relative threshold e.g. arpr)
precisionSilc(h, HX090 < 20000)

# Work intensity
precisionSilc(r, as.factor(RX050))
```

```
##                              call mod        est     variance
## 1 mean(y = as.factor(RX050))   0 0.67350187 1.102257e-05
## 2 mean(y = as.factor(RX050))   1 0.07171251 9.872501e-06
## 3 mean(y = as.factor(RX050))   2 0.25478563 2.635200e-06
```

```
# Number of people with severe material deprivation
precisionSilc(r, total(RX060 == 1))

# Ratio of equivalised disposable income
# per person in the household
precisionSilc(h, ratio(HX090, HX040))
```

Live demonstration

Domain estimation

```
# Domain estimation with where (FR10 : Paris area)
precisionSilc(h, HX090, where = DB040 == "FR10")
```

```
##                                          call      est
## 1 mean(y = HX090, where = DB040 == "FR10") 29940.32
##   variance      std       cv    lower    upper
## 1 416716.1 645.5355 2.156074 28675.09 31205.54
```

```
# Domain estimation with by
precisionSilc(h, HX090, by = DB040)
```

```
##                             call   by      est variance
## 1 mean(y = HX090, by = DB040) FR10 29945.66 417135.9
## 2 mean(y = HX090, by = DB040) FR21 21562.55 811957.0
## 3 mean(y = HX090, by = DB040) FR22 22348.96 435425.5
## 4 mean(y = HX090, by = DB040) FR23 23205.41 602302.2
## 5 mean(y = HX090, by = DB040) FR24 23859.54 743971.3
```

Live demonstration

Complex linearizations using the vardpoor package

```
# Installation of the vardpoor package
install.packages("vardpoor")

# Gini coefficient
precisionSilc(r, gini(HX090))
```

```
##              call      est  variance       std
## 1 gini(y = HX090) 29.18782 0.2856686 0.5344798
##          cv    lower    upper
## 1 1.831174 28.14026 30.23538
```

```
# At-risk of poverty rate
precisionSilc(r, arpr(HX090))
```

```
##              call      est  variance       std      cv
## 1 arpr(y = HX090) 13.35197 0.1812268 0.4257075 3.18835
##     lower    upper
## 1 12.5176 14.18634
```

How-to: `gustave` package starter's guide

"Wrap" a complex function in an easier one

The aim of the gustave package is to hide the complexity of the variance estimation process from the end user.

Basic idea "Wrapping" the complex variance estimation *function* within a variance estimation *wrapper*.

- **variance estimation *function***: survey-specific function with a numerical matrix of data as input and the estimated variances as output
- **variance estimation *wrapper***: generic function which handles systematic operations (linearizations, domain estimation, etc.), calls the variance function and displays the results

Case study: stratified SRS with calibration

- simple random sampling of 2,000 households among 5,000,000
- 10 strata of size 200 each
- calibration on margins with one calibration variable

Remark No unit non-response for this example.

```
names(sample)
  ## [1] "id"            "stratum"       "auxiliary_var"
  ## [4] "pik"           "d"             "w"

names(survey)
  ## [1] "id"            "w"             "auxiliary_var"
  ## [4] "var1"          "var2"
```

Step 1: Define the variance *function*

```r
# Install the gustave package from GitHub
install.packages("devtools")
devtools::install_github("martinchevalier/gustave")
library(gustave)
```

```r
# Define the variance function
variance_function <- function(y){
  # Taking calibration into account
  e <- rescal(y, sample$auxiliary_var)
  # Variance estimation
  var <- varDT(
    y = e, pik = sample$pik, strata = sample$stratum
  )
  return(var)
}
variance_function(survey$var1)
  ## [1] 49383061901
```

Step 2: Define the variance *wrapper*

```
# Define the variance wrapper
variance_wrapper <- define_variance_wrapper(
  variance_function = variance_function
  , reference_id = sample$id
  , default = list(id = "id", weight = "w")
  , objects_to_include = "sample"
)

# Export the survey data and the variance wrapper
# for later use or dissemination
save(
  survey, variance_wrapper
  , file = "precisionSurvey.RData"
)
```

Step 3: Use the variance wrapper

```
# Load the .RData (for instance in an empty environment)
load("precisionSurvey.RData")
ls()
  ## [1] "survey"           "variance_wrapper"

# Use the variance wrapper
variance_wrapper(survey, var1)[, 1:3]
  ##                call      est    variance
  ## 1 total(y = var1) 49844011 49383061901
variance_wrapper(survey, mean(var1), by = var2)
  ##                          call by       est    variance
  ## 1 mean(y = var1, by = var2)  a  9.906593 0.007367807
  ## 2 mean(y = var1, by = var2)  b 10.044675 0.006980245
  ## 3 mean(y = var1, by = var2)  c  9.938034 0.007152497
  ##           std        cv    lower    upper
  ## 1 0.08583593 0.8664526 9.738357 10.07483
  ## 2 0.08354786 0.8317627 9.880924 10.20843
  ## 3 0.08457244 0.8509976 9.772276 10.10379
```

Discussion and perspectives

Use of the gustave package at Insee

The gustave package is currently **used in production** at Insee for the variance estimation of: EU-SILC, LFS, the French victimation survey (and AES by the end of 2017).

The programs it produces are used by survey specialists with no expertise in variance estimation nor in the R software.

The Department of statistical methods sees the gustave package as an **investment**:

- the code of the most generic operations (e.g. linearizations, domain estimation) is documented and open for review
- it will be easier to maintain than a set of separated codes

Compatibility with the `vardpoor` package

The `vardpoor` package is developed and maintained by the Central Statistical Bureau of Latvia (presented in the 2015 EU-SILC Best Practices Workshop in London).

It is also dedicated to variance estimation, but with a different focus:

- it provides a wide variety of linearization functions (Laeken indicators)
- variance estimation is not exact as it relies on the "ultimate cluster" methodology

The **linearization functions provided by the `vardpoor` package can be used within a variance wrapper** produced by the `gustave` package through a **linearization wrapper**.

Milestones and perspectives

- Under intensive development until the end of 2017: consolidation, documentation, code review
- A version will be available on CRAN at the latest in august 2018 (available on GitHub as of now)
- **Key features before the first release**:
  - new and enhanced linearization wrappers (e.g. quantiles)
  - "ready-to-estimate" variance wrapper for simple cases (one-stage stratified SRS with non-response and calibration)

Conclusion

- Variance estimation in household surveys is a complex topic, which implies a **strong collaboration between survey specialists and methodologists**.
- The development of a unifying R package is a source of **efficiency** and improves the **quality** of the whole process.
- It is an investment that also relies on other **open-source initiatives**, e.g. the `vardpoor` package.
- It stimulates the use of softwares **alternative to SAS** at Insee.

Thank your for your attention!

Questions?

Martin Chevalier
martin.chevalier@insee.fr
https://github.com/martinchevalier/gustave