DE GRUYTER
OPEN

Journal of Official Statistics, Vol. 31, No. 2, 2015, pp. 177–203, http://dx.doi.org/10.1515/JOS-2015-0013

ReGenesees: an Advanced R System for Calibration, Estimation and Sampling Error Assessment in Complex Sample Surveys

Diego Zardetto[1]

ReGenesees is a new software system for design-based and model-assisted analysis of complex sample surveys, based on R. As compared to traditional estimation platforms, it ensures easier and safer usage and achieves a dramatic reduction in user workload for both the calibration and the variance estimation tasks. Indeed, *ReGenesees* allows the specification of calibration models in a symbolic way, using R model formulae. Driven by this symbolic metadata, the system automatically and transparently generates the right values and formats for the auxiliary variables at the sample level, and assists the user in defining and calculating the corresponding population totals. Moreover, *ReGenesees* can handle arbitrary complex estimators, provided they can be expressed as differentiable functions of Horvitz-Thompson or calibration estimators of totals. Complex estimators can be defined in a completely free fashion: the user only needs to provide the system with the symbolic expression of the estimator as a mathematical function. *ReGenesees* is in fact able to automatically linearize such complex estimators, so that the estimation of their variance comes at no cost at all to the user. Remarkably, all the innovative features sketched above leverage a particular strong point of the R programming language, namely its ability to process symbolic information.

Key words: Complex estimators; variance estimation; automated linearization; symbolic computation.

1. What is ReGenesees?

ReGenesees (R Evolved Generalized Software for Sampling Estimates and Errors in Surveys) is a full-fledged R software for design-based and model-assisted analysis of complex sample surveys. This system is the outcome of a long-term research and development project, aimed at defining a new standard for calibration, estimation and sampling error assessment to be adopted in all large-scale sample surveys routinely carried out by Istat (the Italian National Institute of Statistics).

The first public release of *ReGenesees* for general availability dates back to December 2011. The system is distributed as open source software under the European Union Public License (EUPL). It can be freely downloaded from JOINUP (the collaborative platform for interoperability and open source software of the European Commission) and from the Istat website.

Until the advent of *ReGenesees*, the estimation phase for sample surveys was handled at Istat by a SAS application named GENESEES (Falorsi and Falorsi 1997). The name of the

[1]Istat – Italian National Institute of Statistics, Via Cesare Balbo, 16 Rome, Rome 00184, Italy. Email: zardetto@istat.it

new R-based system has been deliberately chosen to emphasize Istat's seamless offer of software tools dedicated to that phase, while at the same time highlighting its *evolution* and *enhancement* through R. It is worth stressing, in any case, that the *ReGenesees* system is not the outcome of the simple migration to R of its SAS predecessor, but rather the fruit of a new, challenging and completely independent project.

The principal aim of this article is to introduce *ReGenesees* to the official statistics community. We will not try to provide a complete description of the statistical methods offered by *ReGenesees*, nor to describe its software implementation details: condensing this in an article would be beyond our ability. Instead, we will focus on few qualifying aspects which we perceive as *ReGenesees* "power features" and which, in our opinion, distinguish the system from other existing estimation platforms developed by National Statistical Institutes (NSIs). A broad overview of these qualifying aspects will be given in Section 3, after a brief discussion of the motivations of the project in Section 2.

Since many innovative features of *ReGenesees* can be traced back to a particular strong point of the R programming language, namely its ability to process *symbolic information*, the latter will form the leitmotif of the whole presentation.

R (R Core Team 2014) adheres to the functional programming paradigm and its semantics reveal some notable affinities with LISP (the ancestor of all functional languages). One relevant similarity is precisely the ability to manipulate symbolic expressions, a feature the R community usually refers to as "computing on the language". Perhaps the most popular materialization of this ability is the `formula` class with its ubiquitous usage in R statistical modelling functions (Chambers and Hastie 1992). Section 4 will be devoted to illustrate how *ReGenesees* exploits these potentialities in the calibration context.

A strictly related, though maybe less well known, fact is that R provides functionalities (e.g., symbolic differentiation and polynomial algebra) which are usually thought as hallmarks of Computer Algebra Systems (i.e., specialized software platforms where computation is performed on symbols representing mathematical objects rather than their numeric values). In Section 5 we will show how R facilities for computing symbolic partial derivatives of functions of several variables have been used in *ReGenesees* to fully automate the variance estimation of complex estimators.

Readers interested in assessing to what extent *ReGenesees* could cover the typical needs arising in NSIs during the estimation phase will find a list of the statistical methods made available by the system in Section 6. Section 7 will provide a quick overview of *ReGenesees* software design. There, we will acknowledge *ReGenesees*' indebtedness to the R *survey* package (Lumley 2004, 2010), but will also discuss those specific features which, in our opinion, make *ReGenesees* more fit than *survey* for the large-scale elaborations of the official statistics industry. The progress made so far in migrating Istat production processes to the new system will be reported in Section 8. Finally, Section 9 will address ongoing work and possible future extensions of the *ReGenesees* project.

2. Motivation of the ReGenesees Project

The tasks of calibrating survey weights, computing survey estimates, and assessing their precision, constitute a fundamental building block of the production process of official statistics. These are very complex tasks, whose correct execution requires a good

knowledge of the underlying statistical theory, full awareness of the adopted sampling plans, and often also some insight into the phenomena under investigation. Such skills, which rightfully contribute to define the ideal cultural background of a "good official statistician", cannot at present be entirely superseded by a software (nor probably will in the future), no matter how sophisticated and powerful it may be. However, most NSIs agree that the availability of highly evolved software systems (along with the definition of standard protocols for their optimal usage) is essential to ensure the *accuracy*, the *safety* and the *full reproducibility* of statistical production processes.

In the past, this strategic vision drove Istat and many other NSIs to invest in developing in-house software dedicated to the estimation phase: one may think, for example, of GES of Statistics Canada (Estevao et al. 1995), CLAN of Statistics Sweden (Andersson and Nordberg 1994), CALMAR and POULPE of French NSI INSEE (Sautory 1993; Caron 1998), BASCULA of Statistics Netherlands (Nieuwenbroek et al. 2000), g-CALIB of Statistics Belgium (Vanderhoeft 2001) and GENESEES of Istat (Falorsi and Falorsi 1997).

Today, the same strategic vision (even reinforced by the awareness of the ongoing rapid technological change, with its challenges and opportunities) pushes the same NSIs to renew, enrich or even redesign their software systems from scratch.

Some NSIs are continuously extending their traditional estimation platforms, trying to accommodate additional statistical capabilities as new needs arise in production and the methodology matures. Evidently, this approach is aimed at preserving the overall "look and feel" of the original system as much as possible (in terms of requirements, application logic and user experience), so that no abrupt and costly transitions can affect the production process. Relevant examples are Statistics Canada's project StatMx (Statistical Macro Extensions), which extends GES (Mohl 2007), as well as Statistics Sweden's new estimation tool ETOS (Estimation of Totals and Order Statistics), which enhances CLAN (Andersson 2009).

Istat decided to follow a different path, namely to redesign its estimation platform from scratch and to implement it in a different programming language. While the attempt to soften Istat dependence on proprietary technologies was undeniably a major driving factor, it was not the only relevant one. We were also interested in: (i) pushing our system towards *automation* (to reduce user workload and errors), (ii) improving its *modularity* (so that it could become easier to maintain and evolve), and (iii) providing a *wider choice* of statistical methods (beyond those formerly available). *ReGenesees* is the fruit of the efforts made by Istat in this direction.

3. ReGenesees: a Paradigm Shift

In the design phase of the *ReGenesees* project, it emerged fairly soon that the expectations described in Section 2 could only be met through a radical paradigm shift. As a consequence, *ReGenesees* has turned out to be rather different from its SAS predecessor GENESEES (and, incidentally, from most of other existing estimation tools) from the standpoint of both application logic and user experience. Indeed, besides allowing computing estimates and sampling errors for a much wider range of estimators, *ReGenesees* ensures easier and safer usage and a dramatic reduction in user workload. In a nutshell (see Sections 4, 4.1 and 5, 5.1 for more on points (1) and (2) below):

(1) User interaction with the new system takes place at a *very high level of abstraction*. *ReGenesees* users in fact no longer need to preprocess the survey data relying on ad hoc programs; instead, they only have to feed the software with (i) the data as they are, plus (ii) *symbolic metadata* that describe the adopted sampling design and calibration model (by "calibration model", we mean the assisting linear model underlying a specific calibration problem). At that point, it is up to the system itself to transform, in an automatic and transparent way, the survey data into the complex data structures required to solve the calibration problem and to compute estimates and errors.

(2) Besides totals and absolute frequency distributions (estimators that were already covered by GENESEES), *ReGenesees* is able to compute estimates and sampling errors with respect to means, ratios, multiple regression coefficients, quantiles, and, more generally, with respect to any *complex estimator*, provided it can be expressed as a differentiable function of Horvitz-Thompson or calibration estimators. It is worth stressing that such complex estimators can be defined in a completely free fashion: the user only needs to provide the system with the *symbolic expression* of the estimator as a mathematical function. *ReGenesees* is in fact able to automatically linearize such complex estimators, so that the estimation of their variance comes at no cost at all to the user.

Existing estimation software (the Istat traditional SAS system being no exception) generally gave little support to users in preparing auxiliary variables and population totals for calibration, or in deriving the Taylor expansion of nonlinear estimators and in computing the corresponding linearized variable for variance estimation purposes. As a consequence, ad hoc (often very complex) programs for data preparation, transformation and validity checking were developed and maintained outside the scope of the estimation system: a time-consuming and error-prone practice. *ReGenesees* frees its users from such needs, with an evident gain in terms of workload reduction, better usability and increased robustness against possible errors. Furthermore, letting *ReGenesees* carry out tasks that traditional platforms delegated to a (skilled) human makes the statistical production workflow fully reproducible, as the system persistently logs all the elaboration steps it executes.

Interestingly, both the innovative *ReGenesees* features sketched above leverage R's ability to process *symbolic information*. As a matter of fact, developing the same functionalities in SAS would have been prohibitive: a striking example of what we meant, in Section 2, when referring to "opportunities of technological change".

A technological shift, on the other hand, always involves challenges and some price to be paid. The most threatening challenge faced by the *ReGenesees* project has been to demonstrate that an R-based system would actually be able to manage efficiently the huge amounts of data involved in processing Istat large-scale surveys (see Section 7 for an illustrative example).

A lot of effort has been invested during the whole development cycle of the new system to meet this challenge. Thanks to the empirical evidence and to the reproducible results accumulated during an extensive and thorough testing campaign, we are certain that the challenge has been definitely overcome. Indeed, since its beta release, *ReGenesees* has been

successfully tested on both the Labour Force Survey (LFS) and the Small and Medium Enterprises Survey (SME): where the tasks of calibration and computation of estimates and errors are concerned, these two surveys constitute (each one in its own domain) the most severe test bed available at Istat. Furthermore, today about 20 Istat large-scale surveys have successfully integrated *ReGenesees* into their production workflow.

ReGenesees also underwent an independent validation, which was carried out by colleagues from the UK statistical institute (ONS). A first comparative study, performed on their Life Opportunities Survey, measured *ReGenesees* effectiveness and efficiency using Statistics Canada's GES as a benchmark. The outcome was that *ReGenesees* replicated the results achieved by GES exactly, while ensuring a significant increase in efficiency (in their testing environment, execution times turned out to be halved on average). This result, in turn, triggered a second ONS initiative. *ReGenesees* was used to calibrate three important surveys for the Scottish Government, whose weighting procedures had till then been contracted to three separate external companies: (i) Scottish Household Survey, (ii) Scottish Health Survey, and (iii) Scottish Crime and Justice Survey. Again the results were very satisfactory, and the ONS Methodology Advisory Service suggested *ReGenesees* as a possible "calibration engine" to be adopted in the novel *centralized weighting* framework designed for the Scottish Government (Davidson 2013). Eventually, *ReGenesees* was indeed used in production for the last round of all the aforementioned surveys (Scottish Government (2013a), (2013b), and (2014)).

4. Leveraging Symbolic Information: the Calibration Side

Real-world calibration tasks in the field of official statistics can simultaneously involve several hundreds of auxiliary variables (just to give an impression: each quarterly round of the Italian LFS entails calibrating to known population totals for over 4,000 auxiliary variables). Moreover, the construction of such auxiliary variables is in general highly non-trivial, as they need to be carefully derived from the original survey variables according to the (possibly very complex) adopted calibration models. With respect to such operations, traditional calibration facilities (as those listed in Section 2, mostly based on SAS) gave limited practical support to users, instead devoting dozens of user manuals' pages to describing the standard data structures they expected as input. As a consequence, users had to develop customized programs (typically SAS scripts) in order to generate the right input data to feed the calibration system, with "right" here meaning: (i) appropriate to the survey data and to the calibration task at hand, and (ii) compliant with all the documented rules imposed by the system.

Conversely, users interact with the *ReGenesees* system at very high level of abstraction, as they only need to specify the calibration model in a symbolic way, via R-model formulae. Model formulae are R objects of class `formula` (Chambers and Hastie 1992). Thanks to the flexible syntax of this class and to the powerful semantics of its methods, R-model formulae can be used to compactly specify a wide range of statistical models (far beyond the ANOVA context from which the inspiring notation of Wilkinson and Rogers (1973) originally came). In particular, R-model formulae have the expressive power to represent arbitrarily complex calibration models, including sophisticated modelling aspects such as, for example, interactions between numeric and categorical auxiliary variables, multi-way interactions,

factor crossing, nesting and conditioning, collinearity prevention and term aliasing, handling customized contrasts, referencing auxiliary variables defined on the fly, and so on.

Driven by a calibration-model formula, *ReGenesees* is able to transparently generate the right values and formats for the auxiliary variables at the sample level. In addition, the system assists in defining and calculating the population totals corresponding to the generated auxiliary variables. Indeed, by leveraging again the calibration model formula, *ReGenesees* provides the user with a *template* dataset appropriate to store the requested totals. Whenever the actual population totals are available to the user as such, that is, in the form of already computed aggregated figures, the user has only to fill in the template. This case typically occurs in Italy for household surveys (whose sampling design is two-stage: municipalities + households) because demographic balance figures are updated and released on a monthly basis, whereas a centralized population register is not yet available.

An even bigger benefit is achieved when the sampling frame of the survey is available as a single database table and the actual population totals can be calculated from this source. In such cases, *ReGenesees* is able to automatically compute the totals of the auxiliary variables from the sampling frame, and to safely arrange and format these values so that they can be directly used for calibration. This scenario applies to all the structural business surveys carried out in Italy, whose samples are drawn (typically with a one-stage design) from ASIA, the Istat comprehensive archive of about 4.5 million Italian active enterprises.

In the next section, the abstract considerations sketched above will be illustrated in practice by two calibration examples.

4.1. Two Calibration Examples: Global and Partitioned Calibration

Here we provide two simple examples illustrating how a *ReGenesees* user can tackle a calibration task. Both examples will be discussed in the light of the considerations set out in Section 4.

Both examples will address the same calibration problem, which will be solved *globally* in the first case, and in a *partitioned* way in the second case. For the sake of clarity, each example will be decomposed into atomic elaboration steps. For each elaboration step, first, we will provide a quick natural-language description of what is going on; then, we will show the corresponding R-code statements. Such code fragments will be reported as if they were typed by a user interacting with *ReGenesees* through the ordinary R command-line interface; the same elaborations could also be obtained by exploiting the graphical user interface of the system (see Section 7).

Both examples will involve two artificial datasets mimicking structural business statistics data, which ship with the *ReGenesees* system. The first dataset, `sbs`, represents a sample of enterprises, while the second, `sbs.frame`, is the sampling frame from which the sample has been selected.

We will calibrate `sbs` weights with calibration constraints imposed simultaneously on the total number of employees (`emp.num`) and enterprises (`ent`) inside domains obtained by:

i) crossing two-digit classification of economic activity or industry (`nace2`) and `region` (a convenience, threefold territorial division);
ii) crossing employment size classes (`emp.cl`), economic activity macro-sector (`nace.macro`) and `region`.

Let us now proceed step by step with the first example.

S1. Bind survey data (`sbs`) to sampling design metadata;

```
> sbsdes <- e.svydesign(data=sbs, ids=~id, strata=~strata, weights=~weight, fpc=~fpc)
> sbsdes
Stratified Independent Unit Sampling Design
- [664] strata
- [6909] units

Call:
e.svydesign(data = sbs, ids = ~id, strata = ~strata, weights = ~weight,
    fpc = ~fpc)
```

S2. Specify the calibration model symbolically (`calmodel`) and build a template dataframe to store the corresponding known population totals;

```
> pop <- pop.template(data=sbsdes,
+                     calmodel=~((emp.num+ent):(nace2+emp.cl:nace.macro)):region-1)
> dim(pop)
[1]   1 462
```

S3. Automatically compute the requested totals from the sampling frame (`sbs.frame`) and safely fill the template;

```
> pop <- fill.template(universe=sbs.frame, template=pop)
```

S4. Pass the known totals to the system and perform the calibration task;

```
> sbscal <- e.calibrate(design=sbsdes, df.population=pop, calfun="linear",
+                       bounds=c(0.01,3))
> sbscal
Calibrated, Stratified Independent Unit Sampling Design
- [664] strata
- [6909] units

Call:
e.calibrate(design = sbsdes, df.population = pop, calfun = "linear",
    bounds = c(0.01, 3))
```

Code fragment **S1** simply tells the *ReGenesees* system that the `sbs` sample was selected with a stratified one-stage unit sampling design without replacement. Since the present focus is on calibration, we cannot go into any detail on function `e.svydesign`. Rather, we only point out that it can handle a wide range of complex sampling designs (see Section 6 for a list).

The `calmodel` formula in code fragment **S2** specifies the assisting linear model underlying our complex calibration problem. Without going into syntax details: '$\sim$' declares a model formula, '**:**' means interaction, '**+**' means sum of effects (not arithmetic addition), '**-1**' means no intercept term needed (otherwise, it would be tacitly implied in R). More specifically, `calmodel` identifies the calibration constraints, as well as the related auxiliary variables. Given the nature of the `sbs` dataset, such a calibration model – which involves only six *original* survey variables, two numeric and four categorical – translates into 462 different numeric auxiliary variables.

Code fragment **S2** generates a template dataframe (`pop`) to store properly the known totals of these 462 variables (in fact `pop` has one row and 462 columns, as shown). It is a *template* dataframe in the sense that all the known totals it must be able to store are still

missing, but it has the right certified structure (in terms of dimension, column names, variable types, . . .) to be processed successfully by the calibration facility of the *ReGenesees* system once filled. Note that the `pop.template` function works in a *declarative* way: it avoids any need for the user to understand, comply with, or even be aware of, the structure of the template that is being built.

Function `fill.template`, in code fragment **S3**, transparently computes the requested population totals from the sampling frame, and arranges and formats such values according to the template structure. Note that these elaborations are again driven by the calibration model formula `calmodel`, which is attached as an *invisible* attribute to the template dataframe `pop` returned by the previous code fragment **S2**.

Function `e.calibrate` in code fragment **S4** first automatically generates the model matrix storing the values of the 462 numeric auxiliary variables for the whole sample, then computes the desired calibrated weights.

In conclusion, fragments **S1 – S4** show that we were able to perform a complex calibration task by passing to *ReGenesees* only the data as they were plus symbolic metadata, without any need to work out the 462 numeric auxiliary variables and their population totals.

Let us come back to the `calmodel` formula in code fragment **S2**. This formula identifies a *factorizable* calibration model, as variable `region` acts as a *common factor* interacting with *all* the other variables appearing in the formula. Factor variables with this property split the sample (and the target population) into nonoverlapping subsets known as *"calibration domains"* (*"model groups"* in the terminology of Estevao et al. 1995). The interest in factorizable calibration models lies in the fact that the *global* calibration problem they describe can actually be broken down into smaller *local* subproblems, one per calibration domain, which can be solved separately. This opportunity can, in many cases, result in a dramatic reduction in computational complexity. Obviously the computational efficiency gain increases with the size of the survey and (most importantly) with the number of auxiliary variables involved.

Now, code fragments **S2 – S4** above show how the *ReGenesees* calibration facility `e.calibrate` can be used to solve a factorizable calibration problem with a *global* approach. In the next example, we will instead show how to solve the same problem with a *partitioned* approach. Again no data preparation effort is required of the user.

We are now ready to go on with the second example.

S5. Specify symbolically the reduced calibration model (`calmodel`) and the calibration domains (`partition`), and build the appropriate known totals template;

```
> pop2 <- pop.template(data=sbsdes,
+                      calmodel=~((emp.num+ent):(nace2+emp.cl:nace.macro))-1,
+                      partition=~region)

> dim(pop2)

[1]   3 155
```

S6. Automatically compute the requested totals from the sampling frame and safely fill the template;

```
> pop2 <- fill.template(universe=sbs.frame, template=pop2)
```

S7. Pass the known totals to the system and perform the partitioned calibration task;

```
> sbscal2 <- e.calibrate(design=sbsdes, df.population=pop2, calfun="linear",
+                        bounds=c(0.01,3))

> sbscal2

Calibrated, Stratified Independent Unit Sampling Design
- [664] strata
- [6909] units

Call:
e.calibrate(design = sbsdes, df.population = pop2, calfun = "linear",
    bounds = c(0.01, 3))
```

S8. Check that the calibrated weights obtained through the partitioned algorithm are indeed identical to those obtained previously through the global approach;

```
> all.equal(weights(sbscal), weights(sbscal2))

[1] TRUE
```

Code fragment **S5** shows how we can tell *ReGenesees* to solve our factorizable calibration problem in a *partitioned* way. We only need to: (i) pass to the `calmodel` argument the *reduced* model obtained by factoring the common factor variable `region` out from the original model; (ii) pass the same variable to the `partition` argument in order to identify the calibration domains. Note that this implies a different structure of the known totals template `pop2` as compared to `pop` in **S2**. Specifically, we get as many rows as calibration domains (three in our example, since we have three regions) and a new column to identify such domains. Hence, the number of cells to be filled with actual population totals remains the same, that is, $462 = 3 \times (155 - 1)$, as it must be. Again the user does not need to take care of such format details, as they are automatically handled by function `fill.template` in code fragment **S6**.

Function `e.calibrate` in code fragment **S7** sequentially runs the three calibration subproblems corresponding to the calibration domains identified by `partition`. Lastly, code fragment **S8** shows that the calibrated weights achieved by the partitioned algorithm are indeed equal to those obtained previously through the global approach: we reported this result for illustration only, as it is guaranteed by construction.

We conclude by pointing out two technical aspects of the partitioned calibration task seen in code fragments **S5 – S7**. First, the computational efficiency gain of the partitioned approach is evident even for our toy example: execution time indeed turns out to be reduced by a factor of 10 with respect to the global calibration alternative. Second, solving the calibration problem in a partitioned way has some nontrivial consequences in the variance estimation phase, as discussed in Estevao et al. (1995). Nevertheless, as we will show in Subsection 5.1, the *ReGenesees* system will automatically take care of all the involved technical issues (mainly arising from the interplay between estimation domains and calibration domains), again without requiring any specific effort of the users.

5. Leveraging Symbolic Information: the Linearization Variance Side

The Taylor linearization method is a well-established, approximate tool (Woodruff 1971; Wolter 2007) for estimating the variance of complex estimators, namely estimators that can be expressed as nonlinear (but "smooth", say of class C^2 at least) functions of Horvitz-Thompson (HT) estimators of totals:

$$\hat{\theta} = f(\hat{Y}_1, ..., \hat{Y}_m) \tag{1}$$

where $\hat{Y}_j = \sum_{k \in s} d_k y_{jk}$ and the design weights d_k are reciprocals of first-order inclusion probabilities, $d_k = \pi_k^{-1}$.

The key assumption of the method, generally justified by large-sample arguments (see e.g., Krewski and Rao 1981), is that, as far as variance estimation is concerned, a complex estimator can be approximated by its first-order Taylor series expansion:

$$\hat{\theta} \approx \theta + \sum_{j=1}^{m} \left. \frac{\partial f}{\partial \hat{Y}_j} \right|_{\mathbf{Y}} (\hat{Y}_j - Y_j) \doteq \hat{\theta}_{lin} \tag{2}$$

The linear approximation of the original complex estimator (1) is then expressed (up to constant, though unknown, terms) as the HT total of a single artificial variable $\hat{z}$:

$$\hat{\theta}_{lin} \approx \sum_{k \in s} d_k \hat{z}_k + const \tag{3}$$

Variable $\hat{z}$ in Equation (3) is the so-called *linearized variable* (Woodruff 1971) of the complex estimator (1):

$$\hat{z}_k = \sum_{j=1}^{m} \left. \frac{\partial f}{\partial \hat{Y}_j} \right|_{\hat{\mathbf{Y}}} y_{jk} \tag{4}$$

The approximate identity symbol $\approx$ in Equation (3) and the hat over z in Equations (3)-(4) rest on having evaluated the gradient of function f at estimated totals $\hat{\mathbf{Y}}$ rather than at the corresponding true (but unknown) values $\mathbf{Y}$. From Equation (3) it follows that an estimator of the (approximate) variance of a complex estimator can be built for all sampling designs for which a good estimator of the variance of an HT total is known:

$$\hat{V}(\hat{\theta}) \approx \hat{V}(\sum_{k \in s} d_k \hat{z}_k) \tag{5}$$

Equation (5) summarizes the "golden rule" of the method: estimating the variance of a complex estimator boils down to the much simpler problem of estimating the variance of the HT total of its linearized variable $\hat{z}$, under the sampling design at hand.

The extension to "smooth" functions of calibration estimators (see Särndal 2007 and references therein) of totals is straightforward. Let us indicate an estimator of this kind as follows:

$$\hat{\theta} = f\left(\hat{Y}_1^{CAL}, \ldots, \hat{Y}_m^{CAL}\right) \tag{6}$$

where $\hat{Y}_j^{CAL} = \sum_{k \in s} w_k y_{jk}$ and the calibrated weights w_k are obtained by minimizing an appropriate distance function $G(\mathbf{w}, \mathbf{d})$ from the design weights d_k, subject to calibration constraints involving a set of auxiliary variables $\mathbf{x}$ and the corresponding known population totals: $\sum_{k \in s} w_k \mathbf{x}_k = \sum_{k \in U} \mathbf{x}_k$.

The golden rule (5) still applies to estimators such as (6), the only relevant change being a different expression for the linearized variable (Deville 1999):

$$\hat{z}_k = \sum_{j=1}^{m} \frac{\partial f}{\partial \hat{Y}_j^{CAL}}\bigg|_{\hat{\mathbf{Y}}^{CAL}} g_k \hat{e}_{jk} \tag{7}$$

namely the value of the original variable y_{jk} has been replaced by the product of the g-weight, $g_k = w_k/d_k$, with the estimated *residual* of that variable under the adopted calibration model:

$$\hat{e}_{jk} = y_{jk} - \boldsymbol{x}'_k \cdot \hat{\boldsymbol{\beta}}_j \tag{8}$$

The g-weighted residuals in (7) result from linearizing the GREG estimator (Särndal et al. 1989), and from Deville and Särndal's (1992) well-known finding that, under mild conditions on the involved distance functions, all calibration estimators are actually asymptotically equivalent to the GREG estimator.

Besides asymptotic theory, there is solid empirical evidence that the Taylor linearization method can yield reliable variance estimates, as long as: (i) the functional form of the estimator is well behaved, (ii) the sample is large and the sampling design is not awkward, and (iii) the variables involved in the estimator are not highly skewed at the population level. These conditions are frequently met in official statistics production, but there are notable exceptions, such as the skewness characterizing many business statistics variables. An investigation of the empirical properties of the variance estimators would therefore be an interesting topic for future study.

While the mathematical framework outlined by Equations (1)-(8) is clear, its software implementation involves some subtle and tricky technical points. For instance, domain estimation of standard errors tends to become cumbersome, especially for complicated functions of calibration estimators. Moreover, as anticipated in Subsection 4.1, when a *partitioned* calibration is performed, the interplay between estimation domains and calibration domains has to be carefully taken into account for variance estimation (see the "general case" discussed in sec. 5 of Estevao et al. 1995). Note that this issue has a relevant impact on software implementation, since the *partitioned* calibration approach is computationally far more efficient than the *global* one, if not even the only feasible alternative for some large-scale surveys (see Section 7 for an illustrative example).

From a software development standpoint, the linearization approach to variance estimation has a fundamental drawback: the Taylor series expansion of a nonlinear estimator *does* depend on its functional form *f*. Therefore, using traditional computing environments (e.g., SAS) that are *unable* to perform *symbolic differentiation*, different programs have to be developed separately for each nonlinear statistic *f*. As a direct consequence, traditional systems like those listed in Section 2 suffer from two main limitations.

First, they support only a rather limited set of nonlinear estimators, typically ratios of totals (EUROSTAT 2002, 2013). As a somewhat extreme example, Istat traditional system GENESEES cannot automatically handle any nonlinear estimator. At the other extreme, Statistics Sweden's system CLAN, despite being unquestionably more general and versatile, still can only handle *rational* functions of totals.

The second limitation is that traditional platforms generally cannot allow their users to define *their own* complex estimators, namely statistics which are not built in. Whenever users of such systems need non-built-in estimators, they have to develop ad hoc programs to compute the appropriate linearized variables on their own.

Even CLAN users must actually write SAS (%FUNCTION) macros to let the system understand the functional form of the rational function they are interested in. Moreover, those macros become more and more complicated as the complexity of the estimator grows (see the examples reported in the Appendix), with the risk of impairing the usability of the system (Davies and Smith 1999; Ollila et al. 2004). In fact, CLAN users have to successively decompose their original function into simpler subfunctions until no further simplification is possible, using the four elementary algebraic operations (i.e., *addition*, *subtraction*, *multiplication* and *division*) provided by the system as preprogrammed macros (i.e., %ADD, %SUB, %MULT and %DIV). The purpose of this coding burden is to enable CLAN to compute the linearized variable of a complex rational function of totals in a stepwise fashion, that is, by successively applying appropriate Woodruff transformations corresponding to elementary *binary* functions $+$, $-$, $\times$, and /, which are the only ones the system can *directly* cope with.

ReGenesees overcomes both the limitations mentioned above, again leveraging R's ability to process symbolic information. We achieved this goal through the following steps. First, we devised a simple syntax for specifying arbitrary complex estimators through their functional form, and enabled it by exploiting R methods for manipulating `expression` objects (see the next section for further details). Then, we used advanced R facilities for calculating *symbolic derivatives* to develop a sort of "universal" linearization program. Once equipped with it, we were in the position to add to the system new nonlinear estimators almost for free (see Section 6 for a list). Lastly, we engineered our universal linearization program, making it friendly and fully visible to users. The resulting function, named `svystatL`, handles arbitrary user-defined complex estimators, as we show in the next section with two practical examples.

5.1. *Two Examples of Complex Estimators: Geometric Mean and Standard Deviation of a Variable*

As we have already sketched above, we equipped *ReGenesees* with a simple syntax for specifying arbitrary complex estimators through their functional form, via R `expression` objects. According to this syntax:

i) the estimator of the *total* of a variable is simply represented by the *name* of the variable itself;
ii) the convenience name *ones* identifies an *artificial* variable whose value is 1 for each elementary unit.

Evidently, the system variable `ones` can be used directly to estimate the size (in terms of elementary units) of the population, as well as to define the estimator of the mean of a given survey variable.

By combining rules (i) and (ii) above, and by making use of whatever algebraic operators and mathematical functions R understands, *ReGenesees* users can actually define any estimator they need.

Here are some elementary examples:

- $\hat{Y}$ *maps to* `expression(y)`
- $\hat{N}$ *maps to* `expression(ones)`
- $\hat{R} = \frac{\hat{Y}}{\hat{X}}$ *maps to* `expression(y/x)`
- $\hat{\mu}_y = \frac{\hat{Y}}{\hat{N}}$ *maps to* `expression(y/ones)`
- $\hat{B} = \frac{\hat{T}_{in} - \hat{T}_{out}}{\hat{T}_{in} + \hat{T}_{out}}$ *maps to* `expression((in − out)/(in + out))`

with $\hat{T}_{\text{var}}$ in the last item indicating the estimator of the total of variable *var*.

After this necessary preamble, let us now switch to our complex estimator examples. For each example, we will proceed step by step, illustrating each atomic elaboration step with the same style we adopted in Subsection 4.1.

Note that, since the estimators we are going to tackle cannot be expressed as rational functions of estimators of totals, they could *not* be handled directly (i.e., automatically) by any of the traditional estimation platforms listed in Section 2, not even by CLAN which is the most general and flexible among them.

Our first example will address the *geometric mean* of a (non-negative) survey variable *y*. To this end, recall that the geometric mean of *y* can be expressed as the exponential of the average of the logarithm of *y*, so that our complex estimator reads:

$$\hat{G}_y = e^{(\hat{T}_{log(y)}/\hat{N})} \tag{9}$$

We will work with the same sbs-like data we used in Subsection 4.1, and will select number of employees (`emp.num`) as study variable *y*.

S9. Add a new computed variable, i.e. the log in equation (9), to the original survey design object (`sbsdes`);

```
> sbsdes <- des.addvars(sbsdes, log.emp.num=log(emp.num))
```

S10. Estimate the geometric mean of number of employees (`emp.num`), its standard error and confidence interval;

```
> G <- svystatL(sbsdes, expression(exp(log.emp.num/ones)), conf.int=TRUE)
```

S11. Print on screen the obtained results;

```
> G
                        Complex       SE   CI.l(95%)   CI.u(95%)
exp(log.emp.num/ones)   20.5156   0.0608     20.3965     20.6347
```

The purpose of code fragment **S9** ought to be clear: if we included `log(emp.num)` *directly* inside the `expression` in **S10**, this would have been interpreted – according to the syntax introduced before – as the *logarithm of the estimator of the total* of `emp.num`, rather than as the *estimator of the total of the logarithm* of `emp.num`, which is what Equation (9) actually dictates.

As a whole, code fragments **S9 – S11** above show that *ReGenesees* was able to estimate the variance of a user-defined complex estimator in a completely automated manner, overcoming any need for developing ad hoc programs.

Our second example focuses on the estimator of the *Standard Deviation* of a survey variable y:

$$\hat{S}_y = \sqrt{\frac{\hat{N}}{\hat{N}-1}\left[\hat{\mu}_{y^2} - (\hat{\mu}_y)^2\right]} \tag{10}$$

We will keep working with the same survey data and we will select value added (`va`) as study variable. This time we will compute estimates and sampling errors on a *calibrated* design object, more specifically on the result of our *partitioned* calibration example of Subsection 4.1, `sbscal2`. Moreover, we will not estimate `va`'s standard deviation for the whole population, but rather for domains identified by employment size classes, `emp.cl`: note that each of these estimation domains intersects all the calibration domains of `sbscal2` (which were identified by variable `region`). The purpose of such choices is to show that no additional effort is actually required of a *ReGenesees* user for handling the additional technical complexities of this second example.

S12. Add a new computed variable, i.e. the square in equation (10), to the calibrated survey design object (`sbscal2`);

```
> sbscal2 <- des.addvars(sbscal2, va2=va^2)
```

S13. Estimate the standard deviation of value added (`va`), its standard error and confidence interval for employment size classes (`emp.cl`);

```
> S <- svystatL(sbscal2, expression(sqrt((ones/(ones-1))*((va2/ones) - (va/ones)^2))),
+               by=~emp.cl, conf.int=TRUE)
```

S14. Print on screen the obtained results;

```
> S
  emp.cl     Complex    SE.Complex   CI.l(95%).Complex   CI.u(95%).Complex
   [6,9]     4970.64        429.58             4128.69             5812.60
  (9,19]     4252.76        231.81             3798.42             4707.10
 (19,49]     7535.00        780.52             6005.21             9064.78
 (49,99]     8883.48        588.78             7729.49            10037.48
(99,Inf]    20022.11          0.00            20022.11            20022.11
```

Code fragment **S12** has exactly the same purpose as **S9**. Code fragments **S12 – S14** above demonstrate that *ReGenesees* was actually able to cope with the very complex analysis we set up for our second example in a completely automated way, again overcoming any need for developing ad hoc programs. More specifically, function `svystatL` in fragment **S13** handled all the following tasks rigorously (though transparently to the user): (i) compute symbolically the gradient of the function specifying the complex estimator, f, with respect to the estimators of totals f depends on (i.e., $\hat{Y}$, $\hat{T}_{y^2}$ and $\hat{N}$); (ii) compute the (calibrated) estimates of such totals for all the requested estimation domains; (iii) for each estimation domain, evaluate the gradient of f at the corresponding estimated totals; (iv) for each elementary unit belonging to a given estimation domain, compute the g-weighted residuals of the variables whose totals appear in f, taking into account the different estimated regression coefficients pertaining to the calibration domains which happen to be intersected by the given estimation domain;

(v) for each elementary unit in the sample, compute the linearized variable obtained by putting together all the aforementioned ingredients according to Formula (7); (vi) pass the linearized variable and the direct weights to the variance estimation algorithm appropriate for an HT total under the sampling design at hand; (vii) return the obtained results for the requested estimation domains.

6. ReGenesees Statistical Methods in a Nutshell

From a statistical point of view, the *ReGenesees* system is quite rich and flexible, as it is able to handle a wide range of sampling designs, calibration models and estimators. As anticipated in the article outline, we cannot provide a thorough description of the methods offered by *ReGenesees* here. Instead, we will report them concisely in a list (see Table 1 below) and limit ourselves to clarifying those expressions which might not be self-evident to readers who are not official statistics practitioners. Further details, along with a wealth

Table 1. A summary of ReGenesees statistical methods

- **Complex sampling designs**
  - Multistage, stratified, clustered, sampling designs
  - Sampling with equal or unequal probabilities, with or without replacement
  - "Mixed" sampling designs (i.e., with both self-representing and non-self-representing strata)
- **Calibration**
  - Global and partitioned (for factorizable calibration models)
  - Unit-level and cluster-level weights adjustment
  - Homoscedastic and heteroscedastic models
  - Linear, raking and logit distance functions
  - Bounded and unbounded weights adjustment
  - Multi-step calibrations
- **Basic estimators**
  - Horvitz-Thompson
  - Calibration estimators
- **Variance estimation**
  - Multistage formulation
  - Ultimate cluster approximation
  - Collapsed strata technique for handling lonely PSUs
  - Taylor linearization of nonlinear "smooth" estimators
  - Generalized variance functions method
- **Estimates and sampling errors (standard error, variance, coefficient of variation, confidence interval, design effect) for:**
  - Totals
  - Means
  - Absolute and relative frequency distributions (marginal, conditional and joint)
  - Ratios between totals
  - Multiple regression coefficients
  - Quantiles
- **Estimates and sampling errors for complex estimators**
  - Handles arbitrary differentiable functions of Horvitz-Thompson or calibration estimators
  - Complex estimators can be freely defined by the user
  - Automated Taylor linearization
  - Design covariance and correlation between complex estimators
- **Estimates and sampling errors for subpopulations (domains)**
  - All the analyses above can be carried out for arbitrary domains

of practical examples, can be found in the reference manual of the system (Zardetto 2014). Lastly, we will point out some resources for assessing how *ReGenesees* statistical methods compare to those offered by other existing systems cited in Section 2.

In multistage sampling, it is common jargon to call "self-representing" those primary sampling units (PSUs) which are selected with probability one. Thus, for instance, all PSUs belonging to a "take-all" stratum are self-representing. Sometimes efficiency considerations push survey designs to the extreme of selecting just a single PSU per stratum: this happens, for instance, in the Italian LFS. In these cases, population strata containing just a single self-representing PSU are referred to as self-representing strata (SR strata); all the other strata, containing many PSUs among which just a single one is randomly selected with probability less than one, are called non-self-representing (NSR strata).

The resulting sampling design is sometimes called "mixed", because the actual stages of selection differ for SR and NSR strata. For instance, the Italian LFS is actually a two-stage cluster sampling design inside NSR strata (namely a *pps* selection of municipalities followed by a *srs* of households) and a one-stage cluster sampling inside SR strata (namely a *srs* of households inside those municipalities which are always included in the sample). Note also that for the Italian LFS *all* the PSUs selected inside NSR strata are "lonely PSUs" *by design*.

Both the mixed nature of a sampling design and the occurrence of lonely PSUs (i.e., PSUs which are alone inside a NSR stratum at the sample level) are issues that need to be carefully taken into account in variance estimation, and this is ensured by *ReGenesees*. In particular, the system overcomes problems arising from lonely PSUs by adopting the collapsed strata technique, as proposed by Rust and Kalton (1987).

When clusters selected at subsequent sampling stages ($k \geq 2$) have *equal* inclusion probabilities (e.g., for a stratified two-stage design with *srs* of both PSUs and SSUs), *ReGenesees* correctly estimates the full multistage variance, without neglecting subleading contributions arising from stages after the first ($2, \ldots, k$). This exploits a recursive algorithm similar to the one proposed in Bellhouse (1985), inherited and extended from package *survey* (see Section 7 for details).

Conversely, for *unequal* probability sampling *without* replacement (e.g., the pps selection of PSUs in NSR strata of the Italian LFS), in order to get exact variance estimates second-order inclusion probabilities should be known, which is generally unfeasible. In such cases, *ReGenesees* resorts to so-called Ultimate Cluster approximation (Kalton 1979), which rests on pretending that PSUs were sampled *with* replacement, even if this is not actually the case. If PSUs were sampled with replacement, the *only* contribution to the variance would come from estimated PSU totals, in that one could simply ignore any available information about subsequent sampling stages – which explains the phrase *"ultimate clusters"*. This approximation is known to result in conservative variance estimates, with an upward bias which is negligible as long as the actual sampling fractions of PSUs are very small.

As a rule, *ReGenesees* computes variance estimates of nonlinear estimators with the Taylor approach summarized in Section 5. Quantiles – being *implicit*, *non-smooth* functions of totals – are a mandatory exception. Here *ReGenesees* switches to the method proposed in Woodruff (1952), whose main ingredients are the estimation and the local inversion of the cumulative distribution function of the interest variable, again using facilities from and extending the *survey* package.

The Generalized Variance Functions (GVF) method (see, e.g., sec. 7 of Wolter 2007) is the first (and so far only) *ReGenesees* incursion in the model-based realm. The GVF method (whose justification is largely empirical, with few exceptions) amounts to modelling the variance of an estimator as a function of its expected value, and using the fitted model to *predict* variance estimates, rather than computing them directly.

It is worth stressing that only a rather limited subset of the statistical methods covered by *ReGenesees* was already available inside its SAS predecessor GENESEES. For instance, the only estimators provided were totals and absolute frequencies, and variance estimation in multistage designs could be tackled only under the ultimate cluster approximation.

Readers interested in assessing how *ReGenesees* statistical methods compare to those offered by existing systems developed by NSIs are referred to the following resources. EUROSTAT (2002) provides a synoptic table summarizing the suitability of software tools for sampling designs and related issues on variance estimation (p. 34): this covers BASCULA, CLAN, GENESEES, GES, and POULPE. A similar table, this time also taking into account *ReGenesees* (but loosing GES), can be found in Appendix 7 of EUROSTAT (2013). Davies and Smith (1999) review and compare CLAN and GES in depth. Lastly, Ollila et al. (2004) offer the most comprehensive and thorough comparative study we are aware of, even though restricted to BASCULA, CLAN, and POULPE (only a brief overview is given of g-CALIB and GENESEES).

7. ReGenesees Software Design: a Quick Overview

System Architecture. The *ReGenesees* system has a clear-cut two-layer architecture. The application layer of the system is embedded into an R package named `ReGenesees` (Zardetto 2014). A second R package, called `ReGenesees.GUI` (Zardetto and Cianchetta 2014), implements the presentation layer of the system (namely a Tcl/Tk GUI). Both packages can be run in Windows as well as in Mac, Linux and most of the Unix-like operating systems. While the `ReGenesees.GUI` package requires the `ReGenesees` package, the latter can be used also without the GUI on top. This means that the statistical functions of the system will always be accessible to users interacting with R through the traditional command-line interface (as for code fragments in Subsections 4.1 and 5.1). Conversely, less experienced R users will benefit from the user-friendly mouse-click graphical interface.

Related R Projects. It is worth mentioning that, especially in terms of software design principles, the `ReGenesees` package owes a lot to the beautiful, rich and still growing `survey` package written by Thomas Lumley (Lumley 2004, 2010). Retrospectively, the original seeds of the *ReGenesees* project can be traced back to late 2006, when we were trying to optimize and extend `survey` in order to enable its critical functions to successfully process Istat's large-scale surveys. Fairly soon, this attempt required us to rethink the technical implementation of the package globally, that is, consider its internal structure at a deeper level. Over time, this line of work coagulated into an R package in its own right, with many advanced and useful new features that were not covered by `survey`. A tentative list of *ReGenesees* user-visible improvements over the `survey` package is presented in Table 2.

Table 2. Some ReGenesees improvements over the survey package

Feature	How it works
Calibration and variance estimation functions can efficiently process large-scale surveys even in environments with low computational resources (e.g., ordinary PCs)	• Exploits calibration models factorization through a dedicated divide and conquer algorithm • Accelerates execution and saves memory by means of ad hoc optimizations of many internal functions (e.g., variance, design effects and domain estimation for nonlinear estimators)
Provides estimates and sampling errors for arbitrary user-defined complex estimators, i.e., any nonlinear differentiable function of Horvitz-Thompson or calibration estimators of totals	• Enables users to define their own complex estimators symbolically (i.e., as mathematical functions) by means of R expressions • Exploits R symbolic differentiation facilities to linearize complex estimators automatically, so that their variance is estimated on the fly
Assists users in computing and organizing population totals for calibration tasks	• Driven by the calibration model formula, automatically generates a template dataframe to be filled with actual population totals • If the sampling frame of the survey is available, the template is filled automatically • Able to cope with sampling frames of several million rows and thousands of auxiliary variables by means of a dedicated adaptive chunking algorithm
Interaction with all summary statistics functions (i.e., estimators of totals, means, frequencies, ratios, quantiles, multiple regression coefficients, and complex estimators) has been standardized, so that they are easier to assemble in an industrialized process	• All estimators share (nearly) the same interface, even for domain estimation • All estimators produce return values with the same structure, even for subpopulation estimation • Estimates and sampling errors can be written to database tables or exported to external files in a common data model
New statistical capabilities and utilities	• Hints on feasible bounds for range-restricted calibration • Quick estimates of auxiliary variables totals • Compression of nested factors to reduce model-matrix sparseness in calibration tasks • Detailed diagnostics on the calibration process and on its results • Merge of new survey data into existing design objects • Collapsed strata technique for getting rid of lonely PSUs in variance estimation • Detailed diagnostics on the collapsing process and on the generated superstrata • Covariance and correlation between complex estimators • A generalized variance functions (GVF) infrastructure, i.e., facilities for defining, fitting, testing and plotting GVF models, and to exploit them to predict variance estimates

Table 2. Continued

Feature	How it works
Provides a comprehensive and user-friendly point-and-click GUI	• Pure R implementation, relies on tcltk and tcltk2 packages

Providing a comprehensive head-to-head comparison of *ReGenesees* to `survey` here would be beyond the present article's scope. However, we are able to present to the interested reader at least one example highlighting how the development trajectory of our system happened to diverge from `survey`. This example relates to the *partitioned* calibration functionality of *ReGenesees*, which was introduced in Subsection 4.1 and whose impact on variance estimation has been stressed in Sections 5 and 5.1.

In late 2006 we started studying, analyzing and testing `survey` in order to verify whether that package was able to satisfy, at least partially, the typical needs of Istat sample surveys. By using data from the Italian LFS as empirical test case, we soon realized that `survey` could not have been adopted at Istat "as it was" (Scannapieco et al. 2007). Indeed, every attempt to exploit its calibration function `calibrate` on LFS data invariably led either to memory allocation failures or to unaffordable execution times, whatever testing environment (i.e., hardware and operating system configuration) we set up. The point was that, despite being anything but naive, `survey` code was not optimized for processing such huge amounts of data.

To be slightly more specific: for each quarterly sample, typical LFS datasets have about 200,000 rows, and survey weights must be calibrated using over 4,000 numeric auxiliary variables. If the calibration task was to be tackled *globally* (and this is indeed the only available option in `survey`) those numbers would require: (i) computing a sample model matrix of over 8×10^8 numeric entries (i.e., about 6 GB of memory space), (ii) computing the cross-product of that model matrix, yielding a matrix of over 1.6×10^7 elements (i.e., about 120 MB), (iii) computing the (generalized) inverse of (an appropriate scaling of) that cross-product matrix, and (iv) repeating steps (ii)-(iii) for all the needed iterations of a Newton-Raphson algorithm until convergence. Note that, in spite of appearances, point (iii) – rather than (i) – turns out to be the hardest one, due to the high (typically cubic) computational and space complexity of generalized inverse algorithms.

As anticipated, the viable alternative we figured out was a "divide and conquer" calibration program exploiting a common feature of most Istat large-scale surveys, namely the *factorizable* nature of their calibration tasks. Coming back to the Italian LFS example: since known population benchmarks are defined at NUTS 2 level (i.e., for the 21 Italian administrative regions), *ReGenesees* calibration facility `e.calibrate` can split the unaffordable *global* calibration problem into 21 smaller subproblems. With respect to points (i)–(iii) defined above, each one of these local calibration subproblems involves matrices whose size is, on average, about 400 (i.e., 21^2) times smaller than those arising in the global approach.

Unfortunately, it would have been impossible to overcome `survey` limitations by locally modifying and extending just the calibration function, because the side effects of our partitioned algorithm would have propagated through `survey`'s variance estimation

backbone. The reason is that, as discussed in Section 5 of Estevao et al. (1995), solving the calibration problem in a partitioned way has subtle, nontrivial consequences in the variance estimation phase. Stated differently: in order to ensure that computed variance estimates of calibration estimators (as well as of functions of calibration estimators) are *identical*, *irrespective* of whether the underlying calibration problem has been solved globally or in a partitioned way, the software program addressing variance estimation must behave *differently* in the two cases. This explains why we were forced to override `survey`'s variance estimation facility and to implement dedicated solutions appropriate to our partitioned calibration approach.

Similarly, technical issues arising from the interplay between estimation domains and calibration domains when estimating the variance of partitioned calibration estimators in subpopulations (recall point (iv) at the end of Subsection 5.1.) prevented us from retaining `survey`'s workhorse function for domain estimation `svyby`. Again we had to override that function and to develop suitable alternatives.

In summary, none of the user-visible *ReGenesees* features reported in Table 2 as improvements over the `survey` package has been obtained as a simple add on. Instead, each one is the result of extensive and thorough programming effort at a deeper level. This is not surprising, given the sophisticated and highly specialized nature of both software tools.

Software Interoperability. As testified by recent standardization initiatives such as GSIM (UNECE 2013a) and CSPA (UNECE 2013b), NSIs are striving to modernize their production workflows in such a way that software components could be shared between different organizations and assembled "LEGO-wise" into industrialized processes. Devising and implementing a common information model and a standard production architecture are mandatory steps still to be completed to reach this challenging goal. Anyway, we think *ReGenesees* complies with many of the requirements implied by this modernization vision. Just to mention some of them: (i) it is free, (ii) it is open source, (iii) it is cross-platform, (iv) it supports input and output of data in "open" formats, (v) it is a collection of modules (namely R functions) with clearly defined interfaces, (vi) its main functionalities have been designed to reduce (or avoid, whenever possible) human intervention, (vii) its main functionalities are clearly mapped to GSBPM (UNECE 2013c), (viii) it is available for download and sharing by all interested users, (ix) its technical and end-user documentation is available in English.

8. Migrating Istat Procedures Towards ReGenesees

As already stated above, the first public release of the *ReGenesees* system is quite recent (December 2011). The software began to spread in Istat from late 2010 onwards during its beta-testing cycle. A recent (informal) internal survey revealed that – to date – it is being used in production by 20 Istat large-scale surveys. These include: (i) ten surveys on enterprises carried out in compliance with Eurostat regulations, including seven structural business surveys (e.g., Community Innovation Survey, Information and Communication Technology, Labour Cost Survey) and three short-term statistics surveys (e.g., Services Turnover Indices); (ii) three agri-environmental surveys (e.g., Survey on Permanent Crops); (iii) five surveys in the social demographic domain (e.g., Time Use, Health Conditions and Use of Medical

Services, Safety of Women); (iv) and two census-related surveys (Post Enumeration Survey of the 6th Agricultural Census and 9th Industry and Services Census).

In the opinion of survey statisticians involved, the migration of the standard calibration and estimation procedures from GENESEES to *ReGenesees* resulted in a significant reduction in both user workload and execution time. The observed 3:1 prevalence of business surveys on household surveys is arguably related to what we discussed at the end of Section 4: when samples are drawn from a centralized frame (like the Italian archive of enterprises ASIA), *ReGenesees* automatically computes the totals of the auxiliary variables from the sampling frame, and safely arranges and formats those values so that they can be directly used for calibration.

Though as yet partial, *ReGenesees* penetration can be deemed quite satisfactory, especially considering that SAS has been a *de facto* standard for statistical elaborations in Istat since the early '80s. Many Istat statisticians have strong SAS skills, some of them have been familiar with Istat's legacy estimation platform GENESEES for decades, and there are always some costs involved in changing a consolidated production workflow: it would have been unrealistic to expect an immediate transition to *ReGenesees*.

9. Ongoing Work and Future Extensions

Since its beta release, *ReGenesees* has been steadily gaining ground in Istat: to date, as already sketched above, it has been successfully integrated into the production workflow of about 20 large-scale surveys. Moreover, other Istat surveys are migrating to the new system at present. Surveys with a bigger "mass" (i.e., involving more actors operating in a more complex context) arguably have a greater "inertia" (i.e., are more resilient to changes). In this respect, our next challenge will be to undertake the migration towards *ReGenesees* of the production workflow of the Italian LFS and SME, a task whose *technical* feasibility has already been proven. Internal training courses dedicated to the new R-based system, whose first edition was launched in 2013, will allow an even faster and wider diffusion of *ReGenesees* in Istat production processes.

In the meantime, the *ReGenesees* project is in full swing and still growing. *ReGenesees* version 1.6, whose public release took place in April 2014, added facilities implementing the *Generalized Variance Functions* (GVF) method (Wolter 2007). The option of *predicting* variance estimates based on a fitted model explaining the variance of an estimator in terms of its expected value, rather than directly *computing* such variance estimates, can benefit surveys with very demanding dissemination schedules and publication plans.

We are currently assessing the feasibility of integrating the EVER package (Zardetto 2012) with *ReGenesees*, thus bringing the extended DAGJK technique (Kott 2001) for variance estimation into the latter system. This would make it possible to handle estimators which cannot be expressed as closed-form mathematical functions of sample observations, for example, due to hot-deck imputation variability (Miller and Kott 2011).

Another open line of research points towards the software implementation of the *generalized linearization* technique, which hinges upon the notion of *functional derivatives* of estimators (influence functions in the seminal paper of Deville (1999)). This would be useful whenever the ordinary Taylor method cannot be applied, for example, for

estimators which are based on order statistics or expressed as non-smooth, implicit functions of totals, like the Gini index, the at-risk-of-poverty rate, and other Laeken indicators (Osier 2009). Besides quantiles (which are available in *ReGenesees* too), Statistics Sweden's tool ETOS already provides sampling errors for the Gini index and quantile shares. Anyway, our aim would be to enable *ReGenesees* to cope with arbitrary user-defined functions of estimators of quantiles and cumulative distribution functions. Of course, we are aware of the difficulty of this task.

We are confident that some of the aforementioned methodological enrichments will be included in the next major release of the *ReGenesees* system, very likely together with further developments on the software engineering side.

Appendix

With respect to the ability to *automatically* estimate the variance of nonlinear estimators with the Taylor approach (thus excluding replication methods), Statistics Sweden's tool CLAN (nowadays extended by ETOS) is, to the best of our knowledge, the most general and flexible of the traditional estimation platforms cited in Section 2. Indeed, it is able to cope with arbitrary *rational* functions of estimators of totals. However, as anticipated in Section 5, this comes at the price of asking CLAN users to program SAS macros which become more and more complicated as the complexity of the desired estimator grows. The examples below illustrate this point, draw a comparison with *ReGenesees*, and eventually distil some insights from it.

Example 1

Suppose we want to estimate the mean of income (variable `income`) and its variance for each cell of a two-way table crossing age group (variable `agegrp`) and sector of activity (variable `sector`).

Ex1: CLAN Solution

The %FUNCTION SAS macro we would have to write in CLAN is as follows:

```
%macro function(a, b);
   %tot(tab, income, (sector = &a) and (agegrp = &b))
   %tot(nab, 1, (sector = &a) and (agegrp = &b))
   %div(rab, tab, nab)
   %estim(rab)
%mend;
```

Afterwards, in order to practically obtain the desired results in CLAN, we would have to embed the macro above into an enclosing SAS program ending with a call to macro %CLAN, and run that program.

Ex1: ReGenesees Solution

As *ReGenesees* users we would obtain the desired results by directly invoking function `svystatL` as follows:

```
> svystatL (ex1, expression(income/ones), by=~sector:agegrp)
```

Ex1: Comment

The most relevant point in this example is not the different amount of involved lines of code in itself, but rather the reason for such a difference: the logic with which users interact with CLAN and *ReGenesees*. In fact, while *ReGenesees* users must only ask the system *what* they need (i.e., they have to *invoke* a program), CLAN users must also provide the system with an explanation of *how* it has to compute *what* they need (i.e., they have to *write* a program).

As we are going to see in the next example, the effects induced by these diverse interaction paradigms become more evident as soon as a more complex estimator is addressed.

Example 2

Suppose we want to compute estimates and sampling errors for a product of two ratios between totals, say $\hat{Q} = (\hat{Y}_1/\hat{Y}_2) \times (\hat{Y}_3/\hat{Y}_4)$, again for each cell of the two-way table used in Example 1.

Ex2: CLAN Solution

The %FUNCTION SAS macro we would have to write in CLAN is as follows:

```
%macro function(a, b);
   %tot(y1ab, y1, (sector = &a) and (agegrp = &b))
   %tot(y2ab, y2, (sector = &a) and (agegrp = &b))
   %div(r1ab, y1ab, y2ab)
   %tot(y3ab, y3, (sector = &a) and (agegrp = &b))
   %tot(y4ab, y4, (sector = &a) and (agegrp = &b))
   %div(r2ab, y3ab, y4ab)
   %tot(q1ab, r1ab, (sector = &a) and (agegrp = &b))
   %tot(q2ab, r2ab, (sector = &a) and (agegrp = &b))
   %mult(Q, q1ab, q2ab)
   %estim(Q)
%mend;
```

Afterwards, in order to practically obtain the desired results in CLAN, we would have to embed the macro above into an enclosing SAS program ending with a call to macro %CLAN, and run that program.

Ex2: ReGenesees Solution

As *ReGenesees* users we would obtain the desired results by directly invoking function `svystatL` as follows:

```
> svystatL(ex2, expression((y1/y2)*(y3/y4)), by=~sector:agegrp)
```

Ex2: Comment

The most relevant point here is that the increased complexity of the estimator (as compared to Example 1) doesn't affect a *ReGenesees* user at all, whereas a CLAN user has to write a trickier and lengthier SAS macro. As explained in Section 5, this is because CLAN users always need to tell CLAN *how* to tackle the estimator they are interested in, and this is achieved by successively decomposing it into simpler subfunctions, until no further simplification is possible. This decomposition requires more and more steps as the estimator complexity grows.

10. References

Andersson, C. 2009. Using Auxiliary Information in the Calculation of Order Statistics and Estimated Totals in a Large Scale Production Environment. In Proceedings of the 57th Session of the International Statistical Institute (ISI). Durban, South Africa, 16–22 August 2009. Available at: http://isi.cbs.nl/iamamember/CD8-Durban2009/index.htm (accessed May 2015).

Andersson, C. and L. Nordberg. 1994. "A Method for Variance Estimation of Non-Linear Functions of Totals in Surveys – Theory and Software Implementation." *Journal of Official Statistics* 10: 395–405.

Bellhouse, D.R. 1985. "Computing Methods for Variance Estimation in Complex Surveys." *Journal of Official Statistics* 1: 323–329.

Caron, N. 1998. Le logiciel POULPE: aspects méthodologiques. In: INSEE: Actes des Journées de Méthodologie. Available at: http://jms.insee.fr/files/documents/1998/513_1-JMS1998_S3-1_CARON_P173-200.PDF (accessed May 2015).

Chambers, J.M. and T.J. Hastie. 1992. *Statistical Models in S*. London: Chapman & Hall.

Davidson, M. 2013. *Scottish Population Surveys Centralised Weighting Project. Weighting project report of the Scottish Government*. Available at: http://www.scotland.gov.uk/Topics/Statistics/About/Surveys/WeightingProjectReport (accessed August 2014).

Davies, P. and P. Smith. 1999. *Model Quality Report in Business Statistics. Volume II: Comparison of Variance Estimation Software and Methods, EUROSTAT*. Available at: http://epp.eurostat.ec.europa.eu/portal/page/portal/quality/documents/MODEL%20QUALITY%20REPORT%20VOL%202.pdf (accessed August 2014).

Deville, J.C. and C.-E. Särndal. 1992. "Calibration Estimators in Survey Sampling." *Journal of the American Statistical Association* 87: 376–382.

Deville, J.C. 1999. "Variance Estimation for Complex Statistics and Estimators: Linearization and Residual Techniques." *Survey Methodology* 25: 193–203.

Estevao, V., M.A. Hidiroglou, and C.-E. Särndal. 1995. "Methodological Principles for a Generalized Estimation System at Statistics Canada." *Journal of Official Statistics* 11: 181–204.

EUROSTAT. 2002. Variance estimation methods in the European Union. Monographs of Official Statistics, Publications Office of the European Union, Luxembourg. Available at: http://epp.eurostat.ec.europa.eu/portal/page/portal/research_methodology/documents/MOS_20VARIANCE_ESTIMATION_202002.pdf (accessed August 2014).

EUROSTAT. 2013. Handbook on precision requirements and variance estimation for ESS households surveys. Methodologies & Working papers, Publications Office of the European Union, Luxembourg. Available at: http://epp.eurostat.ec.europa.eu/cache/ITY_OFFPUB/KS-RA-13-029/EN/KS-RA-13-029-EN.PDF (accessed August 2014).

Falorsi, P.D. and S. Falorsi. 1997. "The Italian Generalised Package for Weighting Persons and Families: Some Experimental Results with Different Non-Response Models." *Statistics in Transition* 3: 357–381.

Kalton, G. 1979. "Ultimate Cluster Sampling." *Journal of the Royal Statistical Society* 142: 210–222.

Kott, P.S. 2001. "The Delete-A-Group Jackknife." *Journal of Official Statistics* 17: 521–526.

Krewski, D. and J.N.K. Rao. 1981. "Inference from Stratified Sample: Properties of Linearization, Jackknife, and Balanced Repeated Replication Methods." *The Annals of Statistics* 9: 1010–1019.

Lumley, T. 2004. "Analysis of Complex Survey Samples." *Journal of Statistical Software* 9: 1–19.

Lumley, T. 2010. *Complex Surveys: A Guide to Analysis Using R*. New York: John Wiley & Sons.

Miller, D. and P.S. Kott. 2011. "Using the DAG Jackknife to Measure the Variance of an Estimator in the Presence of Item Nonresponse." In Proceedings of the JSM (July 30–August 4, 2011) Alexandria, VA: American Statistical Association, 1121–1129. Available at: http://nass.usda.gov/Education_and_Outreach/Reports,_Presentations_and_Conferences/reports/conferences/JSM-2011/JSM-2011-Miller.pdf

Mohl, C. 2007. "The Continuing Evolution of Generalized Systems at Statistics Canada for Business Survey Processing." In Proceedings of the Third International Conference on Establishment Surveys (ICESIII) (June 18–21, 2007), American Statistical Association, 758–768. Available at: http://www.amstat.org/meetings/ices/2007/proceedings/ICES2007-000135.PDF

Nieuwenbroek, N., R. Renssen, and L. Hofman. 2000. "Towards a Generalized Weighting System." In Proceedings of the Second International Conference on Establishment Surveys (ICESII) (June 17–21, 2000), American Statistical Association, 667–676. Available at: http://www.amstat.org/meetings/ices/2000/proceedings/S09.pdf

Ollila, P., Y. Berger, H.J. Boonstra, A. Davison, A. Laaksonen, K. Magg, R. Munnich, D. Ohly, S. Sardy, K. Sostra, and J. van den Brakel. 2004. "Evaluation of Software for Variance Estimation in Complex Surveys, DACSEIS project, Deliverables 4.1 and 4.2." . Available at: https://www.uni-trier.de/fileadmin/fb4/projekte/SurveyStatisticsNet/Dacseis_Deliverables/DACSEIS-D4-1-4-2.pdf (accessed August 2014).

Osier, G. 2009. "Variance Estimation for Complex Indicators of Poverty and Inequality Using Linearization Techniques." *Survey Research Methods* 3: 167–195.

R Core Team. 2014. R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. Available at: http://www.R-project.org (accessed August 2014).

Rust, K. and G. Kalton. 1987. "Strategies for Collapsing Strata for Variance Estimation." *Journal of Official Statistics* 3: 69–81.

Särndal, C.-E., B. Swensson, and J. Wretman. 1989. "The Weighted Residual Technique for Estimating the Variance of the General Regression Estimator of the Finite Population Total." *Biometrika* 76: 527–537. Doi: http://dx.doi.org/10.1093/biomet/76.3.527.

Särndal, C.-E. 2007. "The Calibration Approach in Survey Theory and Practice." *Survey Methodology* 33: 99–119.

Sautory, O. 1993. La macro CALMAR: Redressement d'un Echantillon par Calage sur Marges. Document de travail de la Direction des Statistiques Démographiques et Sociales, no. F9310. Available at: http://www.insee.fr/fr/methodes/outils/calmar/doccalmar.pdf (accessed May 2015).

Scannapieco, M., D. Zardetto, and G. Barcaroli. 2007. *La Calibrazione dei Dati con R: una Sperimentazione sull'Indagine Forze di Lavoro ed un Confronto con GENESEES/SAS*. Collana Contributi, 4, Istat, Italy. Available at: http://www3.istat.it/dati/pubbsci/contributi/Contributi/contr_2007/2007_4.pdf (accessed August 2014).

Scottish Government. 2013a. *Scottish Household Survey – Methodology and Fieldwork Outcomes 2012*. Available at: http://www.scotland.gov.uk/Resource/0044/00443332.pdf (accessed August 2014).

Scottish Government. 2013b. *Scottish Health Survey 2012 – Volume 2 Technical Report*. Available at: http://www.scotland.gov.uk/Resource/0043/00434643.pdf (accessed August 2014).

Scottish Government. 2014. *Scottish Crime and Justice Survey 2012/13 – Technical Report*. Available at: http://www.scotland.gov.uk/Resource/0044/00445791.pdf (accessed August 2014).

UNECE. 2013a. *Generic Statistical Information Model (GSIM), version 1.1*. Available at: http://www1.unece.org/stat/platform/pages/viewpage.action?pageId = 59703371 (accessed August 2014).

UNECE. 2013b. *Common Statistical Production Architecture (CSPA), version 1.0*. Available at: http://www1.unece.org/stat/platform/display/CSPA/CSPA+v1.0 (accessed August 2014).

UNECE. 2013c. *Generic Statistical Business Process Model (GSBPM), version 5.0*. Available at: http://www1.unece.org/stat/platform/display/metis/The+Generic+Statistical+Business+Process+Model (accessed August 2014).

Vanderhoeft, C. 2001. *Generalised Calibration at Statistics Belgium. SPSS Module g–CALIB–S and Current Practices*. Statistics Belgium Working Paper no. 3. Available at: http://statbel.fgov.be/nl/binaries/paper03%5B1%5D_tcm325-35412.pdf (accessed May 2015).

Wilkinson, G.N. and C.E. Rogers. 1973. Symbolic Description of Factorial Models for Analysis of Variance. *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 22: 392–399.

Wolter, K.M. 2007. *Introduction to Variance Estimation*, Second Edition. New York: Springer.

Woodruff, R.S. 1952. Confidence Intervals for Medians and Other Position Measures. *Journal of the American Statistical Association* 47: 635646. Doi: http://dx.doi.org/10.1080/01621459.1952.10483443

Woodruff, R.S. 1971. "A Simple Method for Approximating the Variance of a Complicated Estimate." *Journal of the American Statistical Association* 66: 411–414. Doi: http://dx.doi.org/10.1080/01621459.1971.10482279.

Zardetto, D. 2012. *EVER: Estimation of Variance by Efficient Replication*. R package version 1.2, Istat, Italy. Available at: http://cran.r-project.org/web/packages/EVER/index.html (accessed August 2014).

Zardetto, D. 2014. *ReGenesees: R Evolved Generalized Software for Sampling Estimates and Errors in Surveys*. R package version 1.6, Istat, Italy. Available at: https://joinup.ec.europa.eu/software/regenesees/description (accessed August 2014).

Zardetto, D. and R. Cianchetta. 2014. *ReGenesees.GUI: a TclTk Interface for the ReGenesees Package*. R package version 1.6, Istat, Italy. Available at: https://joinup.ec.europa.eu/software/regenesees/description (accessed August 2014).

Received July 2013
Revised August 2014
Accepted August 2014