Functions in SamplingStrata package - III

Marco Ballin, Giulio Barcaroli

20 May 2019

Use of models to take into account the anticipated variance

It has been assumed that, when optimizing the stratification of a sampling frame, values of the target variables Y's are available for the generality of the units in the frame, or at least for a sample of them by means of which it is possible to estimate means and standard deviation of Y's in atomic strata.

Of course, this assumption is seldom expected to hold. The situation in which some proxy variables are available in the frame is much more likely to happen.

In these situations, instead of directly indicating the real target variables, proxy ones are named as Y's.

By so doing, there is no guarantee that the final stratification and allocation can ensure the compliance to the set of precision constraints.

In order to take into account this problem, and to limit the risk of overestimating the expected precision levels of the optimized solution, it is possible to carry out the optimization by considering, instead of the expected coefficients of variation related to proxy variables, the anticipated coefficients of variation (ACV) that depend on the model that is possile to fit on couples of real target variables and proxy ones. In the current implementation, only models linking continuous variables can be considered.

Use of models to take into account the anticipated variance

In order to introduce the anticipated variance in the overall optimization, it is necessary first of all to model the target variable (dependent variable) with respect to the available proxy variable (independent variable).

For instance, let us consider the swiss municipalities example, and let have the same stratification variables and same target variables:

- "Surfacesbois","Surfacescult"
- "Pop020", "Pop2040"

But suppose now that the target variables are not available. Instead, only the area covered by buildings in each municipality is available ("Airbat").

Let us suppose to model the relationships between these variables for instance having the availability of a subset of observations:

```
set.seed(123)
mun_samp <- swissmunicipalities[sample(1:nrow(swissmunicipalities),500),]
```

Use of models to take into account the anticipated variance

Now we can fit the model between "Pop020" and "Airbat" in this way:

```
mod_Pop_020_Airbat <- lm(log(Pop020) ~ log(Airbat), data=mun_samp)
summary(mod_Pop_020_Airbat)
```

```
##
## Call:
## lm(formula = log(Pop020) ~ log(Airbat), data = mun_samp)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.5360 -0.2323  0.0271  0.2711  1.4012
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.49590    0.06732   22.22   <2e-16 ***
## log(Airbat)  1.20166    0.01987   60.49   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4661 on 498 degrees of freedom
## Multiple R-squared:  0.8802, Adjusted R-squared:  0.8799
## F-statistic:  3659 on 1 and 498 DF,  p-value: < 2.2e-16
```

Use of models to take into account the anticipated variance

and between “Pop2040” and “Airbat” in this way:

```
mod_Pop_2040_Airbat <- lm(log(Pop2040) ~ log(Airbat), data=mun_samp)
summary(mod_Pop_2040_Airbat)
```

```
##
## Call:
## lm(formula = log(Pop2040) ~ log(Airbat), data = mun_samp)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.01182 -0.26627  0.00991  0.29389  1.79449
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.37129    0.07000   19.59   <2e-16 ***
## log(Airbat)  1.25934    0.02066   60.97   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4846 on 498 degrees of freedom
## Multiple R-squared:  0.8818, Adjusted R-squared:  0.8816
## F-statistic:  3717 on 1 and 498 DF,  p-value: < 2.2e-16
```

Use of models to take into account the anticipated variance

We can now define the following dataframe:

```
model <- NULL
model$beta[1] <- mod_Pop_020_Airbat$coefficients[2]
model$sig2[1] <- summary(mod_Pop_020_Airbat)$sigma
model$type[1] <- "loglinear"
model$gamma[1] <- 0
model$beta[2] <- mod_Pop_2040_Airbat$coefficients[2]
model$sig2[2] <- summary(mod_Pop_2040_Airbat)$sigma
model$type[2] <- "loglinear"
model$gamma[2] <- 0
model <- as.data.frame(model)
model
```

```
##       beta      sig2      type gamma
## 1 1.201665 0.4660857 loglinear     0
## 2 1.259340 0.4846068 loglinear     0
```

Optimization without taking into account the models

We define the frame in this way:

```
swissmunicipalities$id <- c(1:nrow(swissmunicipalities))
swissmunicipalities$dom <- 1

swissframe <- buildFrameDF(df = swissmunicipalities,
                           id = "id",
                           X = c("Surfacesbois",
                                 "Surfacescult"),
                           Y = c("Airbat",
                                 "Airbat"),
                           domainvalue = "dom")
swissframe$X1 <- var.bin(swissframe$X1, bins=15)
swissframe$X2 <- var.bin(swissframe$X2, bins=15)
```

and the precision constraints

```
swisserrors <- as.data.frame(list(DOM="DOM1",
                                  CV1=0.08,
                                  CV2=0.12,
                                  domainvalue=1
                             ))

swisserrors
```

```
##    DOM  CV1  CV2 domainvalue
## 1 DOM1 0.08 0.12           1
```

Optimization without taking into account the models

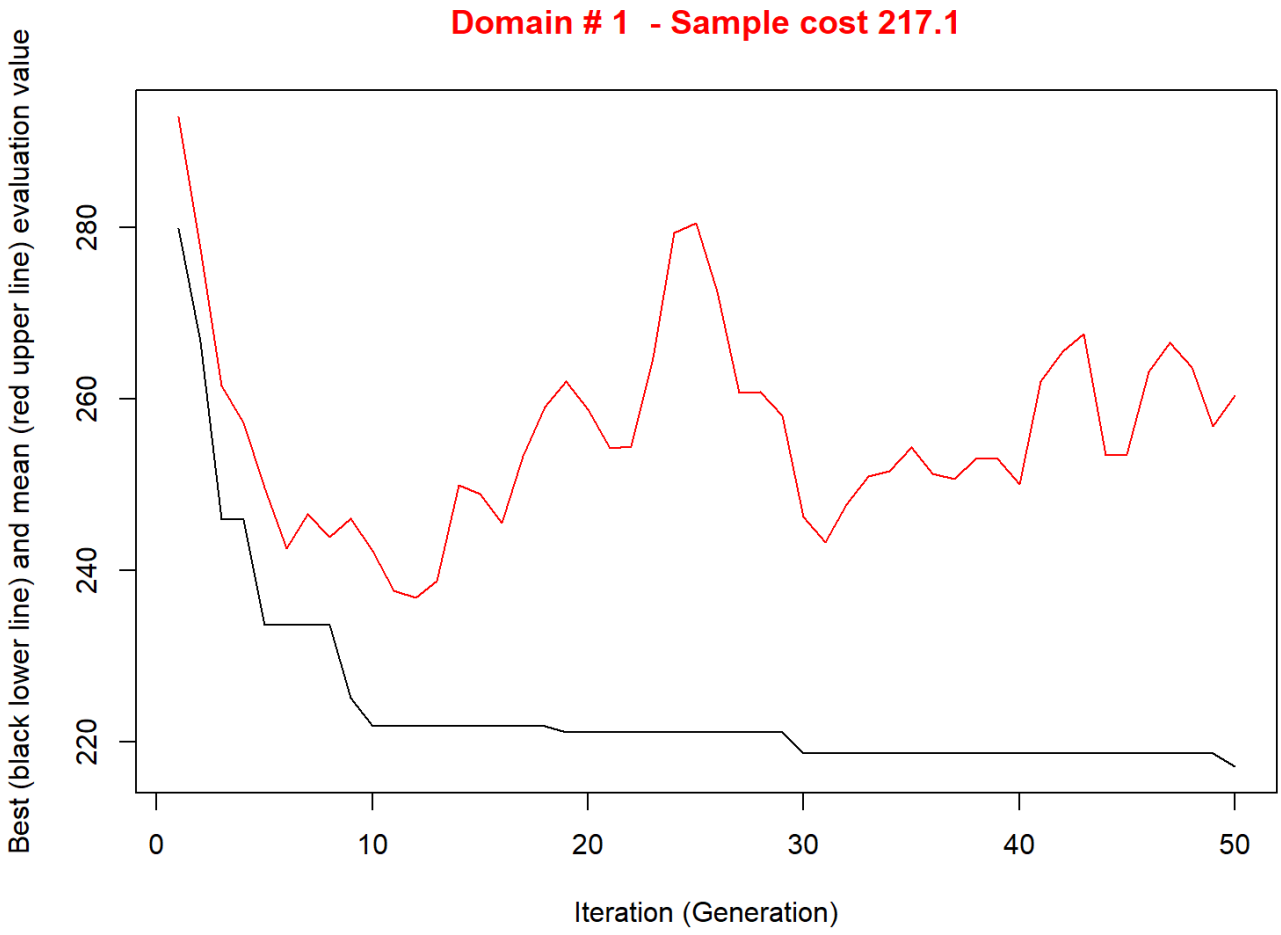
The optimization step:

```
swissstrata1 <- buildStrataDF(swissframe, progress = FALSE)
```

```
##
## Computations are being done on population data
##
## Number of strata:  197
## ... of which with only one unit:  15
```

```
set.seed(123)
solution1 <- optimizeStrata(
  errors = swisserrors,
  strata = swissstrata1,
  iter = 50,
  pops = 10,
  writeFiles = FALSE,
  showPlot = FALSE,
  parallel = FALSE)
```

```
##
##  *** Domain :  1   1
##  Number of strata :  197
## ------------------------------------------
## Optimal stratification with Genetic Algorithm
## ------------------------------------------
##  *** Parameters ***
## --------------------------
## Domain:  1
## Maximum number of strata:  197
## Minimum number of units per stratum:  2
## Take-all strata (TRUE/FALSE):  FALSE
## number of sampling strata :  197
## Number of target variables:  2
## Number of domains:  1
## Number of GA iterations:  50
## Dimension of GA population:  10
## Mutation chance in GA generation:  NA
## Elitism rate in GA generation:  0.2
## Chance to add strata to maximum:  0
## Allocation with real numbers instead of integers:  TRUE
##  *** Sample cost:  217.0986
##  *** Number of strata:  33
```




```
##
##  *** Sample size :  217
##  *** Number of strata :  33
## --------------------------
```

```
sum(round(solution1$aggr_strata$SOLUZ))
```

```
## [1] 217
```

```
expected_CV(solution1$aggr_strata)
```

```
##      cv(Y1) cv(Y2)
## DOM1  0.079  0.079
```

Optimization without taking into account the models

We now evaluate the expected precision on the true target variables:

```
newstrata <- updateStrata(swissstrata1,solution1)
framenew1 <- updateFrame(swissframe,newstrata)
framenew1 <- framenew1[order(framenew1$ID),]
swissmunicipalities <- swissmunicipalities[order(swissmunicipalities$id),]

frame_pop <- framenew1
frame_pop$Y1 <- swissmunicipalities$Airbat
frame_pop$Y2 <- swissmunicipalities$Pop020
frame_pop$Y3 <- swissmunicipalities$Pop2040
eval1 <- evalSolution(frame_pop,
                      solution1$aggr_strata,
                      nsampl=100,
                      writeFiles=FALSE,
                      progress=FALSE)
eval1$coeff_var
```

domain	cv(Y1)	cv(Y2)	cv(Y3)
1	0.0799	0.1307	0.1634

Optimization taking into account the models

Let us now build the atomic strata taking into account the fitted models:

```
swissstrata2 <- buildStrataDF(swissframe, model = model, progress = FALSE)
```

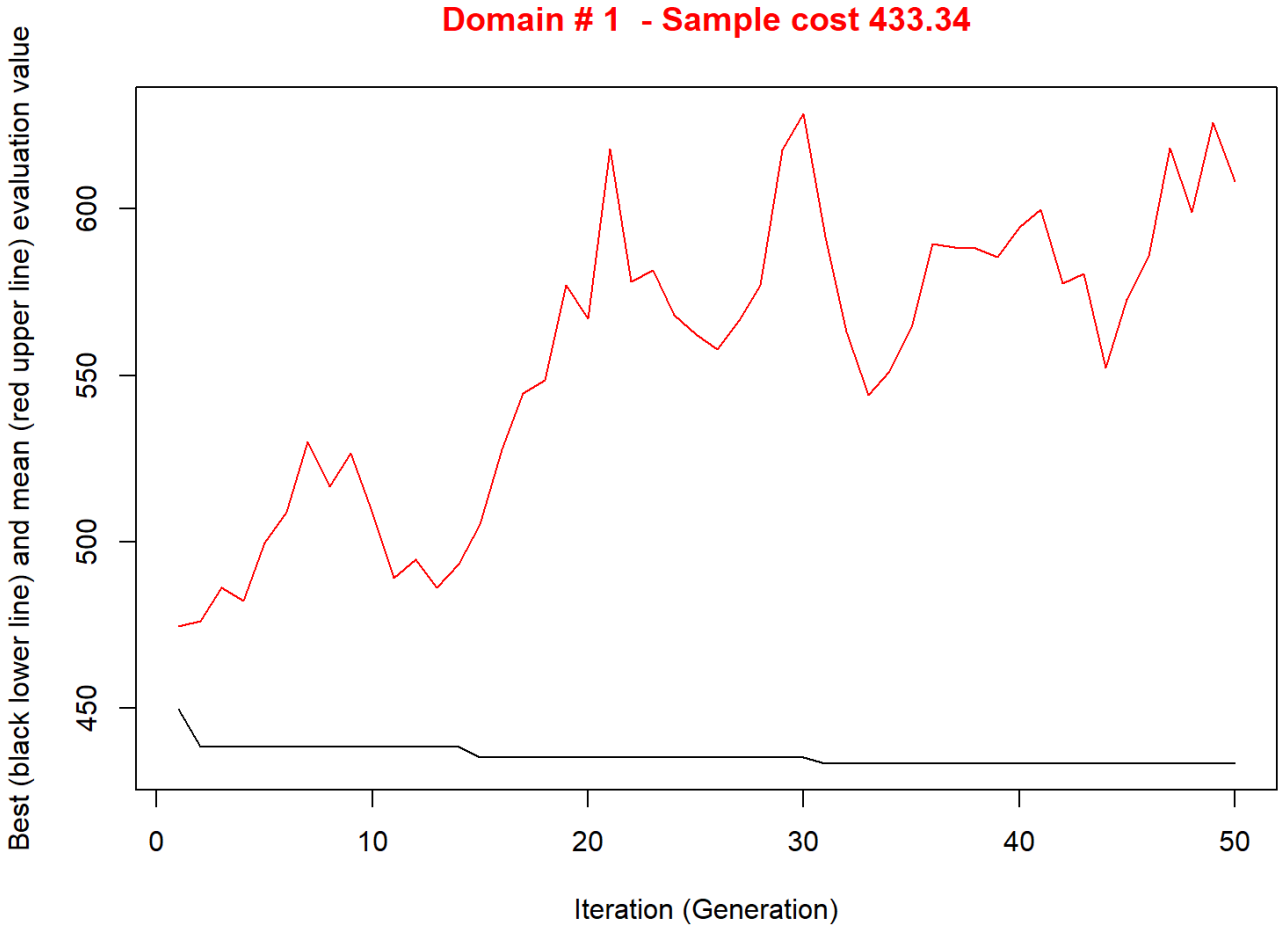
```
##
## Computations are being done on population data
##
## Number of strata:  197
## ... of which with only one unit:  15
```

Optimization taking into account the models

and then proceed with the optimization:

```
set.seed(123)
solution2 <- optimizeStrata(
  errors = swisserrors,
  strata = swissstrata2,
  iter = 50,
  pops = 10,
  writeFiles = FALSE,
  showPlot = FALSE,
  parallel = FALSE)
```

```
##
##  *** Domain :  1   1
##  Number of strata :  197
## ---------------------------------------------
## Optimal stratification with Genetic Algorithm
## ---------------------------------------------
##  *** Parameters ***
## --------------------------
## Domain:  1
## Maximum number of strata:  197
## Minimum number of units per stratum:  2
## Take-all strata (TRUE/FALSE):  FALSE
## number of sampling strata :  197
## Number of target variables:  2
## Number of domains:  1
## Number of GA iterations:  50
## Dimension of GA population:  10
## Mutation chance in GA generation:  NA
## Elitism rate in GA generation:  0.2
## Chance to add strata to maximum:  0
## Allocation with real numbers instead of integers:  TRUE
##  *** Sample cost:  433.3408
##  *** Number of strata:  87
```




```
##
##  *** Sample size :  430
##  *** Number of strata :  87
## --------------------------
```

```
sum(round(solution2$aggr_strata$SOLUZ))
```

```
## [1] 430
```

```
expected_CV(solution2$aggr_strata)
```

```
##      cv(Y1) cv(Y2)
## DOM1  0.079  0.083
```

Optimization taking into account the models

Again, let us evaluate the expected precision on the true target variables:

```
newstrata <- updateStrata(swissstrata2,solution2)
framenew2 <- updateFrame(swissframe,newstrata)
framenew2 <- framenew2[order(framenew2$ID),]
swissmunicipalities <- swissmunicipalities[order(swissmunicipalities$id),]
# evaluation of precision on real target variables

frame_pop <- framenew2
frame_pop$Y1 <- swissmunicipalities$Airbat
frame_pop$Y2 <- swissmunicipalities$Pop020
frame_pop$Y3 <- swissmunicipalities$Pop2040
eval2 <- evalSolution(frame_pop,
                      solution2$aggr_strata,
                      nsampl=100,
                      writeFiles=FALSE,
                      progress=FALSE)
eval2$coeff_var
```

domain	cv(Y1)	cv(Y2)	cv(Y3)
1	0.0466	0.0718	0.0828

```
swisserrors
```

```
##    DOM  CV1  CV2 domainvalue
## 1 DOM1 0.08 0.12           1
```