

```

1  conteo <- function(datos_j,N_j){
2  #
3  #   This function counts the number of individuals in  $S_j$  and
4  #    $\widetilde{S}_j$ , with respect to the total number of individuals
5  #   in a given planned domain  $j$ .
6  #
7  #   Input:
8  #       "datos_j"   -   This object should contain two columns labelled: "n_i" and
9  #                       "domplan".
10 #                       "n_i"   -   Number of individuals in the ith group of
11 #                               individuals, and
12 #                       "domplan" - Column vector with categories for the planned domains
13 #       "N_j"       -   This object should include two columns labelled: "N_j" and
14 #                       "domplan".
15 #                       "N_j"   -   Number of individuals in the jth planned domain, and
16 #                       "domplan" - Column vector with categories for the planned domains
17 #
18 #   Output:
19 #       M           -   Number of groups in "S_j"
20 #       CardS       -   Number of individuals in "S_j" (if "n_i"=1 for any "i", then
21 #                       Cards=M=nrow(datos_j))
22 #       CardNoS     -   Number of individuals out of sample "Stilde_j"
23 #
24 #   Counting the number of individuals in the sample
25 CardS <- as.matrix(sum(datos_j[, "n_i"]))
26 colnames(CardS) <- c("CardS")
27
28 #   Counting the number of individuals outside the sample
29 CardNoS <- as.matrix(N_j - CardS)
30 colnames(CardNoS) <- c("CardNoS")
31
32 #   Number of groups in the sample
33 M <- as.matrix(nrow(datos_j))
34 colnames(M) <- c("M")
35
36 #   Output
37 salida <- list(M,CardS,CardNoS)
38 return(salida)
39 }
40
41 domnoplan <- function(datos,datos_ant,domplan_N,alphaDP,colid_D,alpha_D,inter,part,nSim){
42 #
43 #   Generates Monte Carlo samples of the predictive distribution of totals of a finite
44 #   population
45 #   segmented in planned and unplanned domains, along with simulations of the
46 #   predictive distribution
47 #   for the composition of the population between the unplanned domains.
48 #
49 #   Input:
50 #       datos        -   (Mxp)-dimensional array with positive entries for S_j

```

```

49 #      datos_ant      -   (Mxp)-dimensional reference array for calibration of G_0
50 #      domplan_N      -   Matrix array with counts of individuals in each planned domain
51 #      alphaDP        -   J-dimensional array with positive entries for the parameters of
the Dirichlet process for F_j (with J being the number of planned domains)
52 #      colid_D        -   D-dimensional matrix array with the columns in 'datos' that
correspond to the indicator variables of the planned domains (those indicator variables
represent a portion of 'datos')
53 #      alpha_D        -   (JxD)-dimensional array with positive entries for the
parameters of the multinomial-Dirichlet component for the composition across unplanned
domains.
54 #                               Note: Each one of the J rows is a vector of composition for P_j
divided across the D unplanned domains
55 #      inter          -   Tuning parameter for model comparison and selection (related to
calibration of G_0)
56 #      part           -   Number of partitions for predictive cross-validation (related
to calibration of G_0)
57 #      nSim           -   Number of Monte Carlo simulated replicates of the predictive
distribution
58 #
59 # Details:
60 #      - datos : Represents the data sample of the target population, unplanned
domains labelled.
61 #                               It should contain the following columns:
62 #                               "domplan" - Planned domains categories.
63 #                               "y_i"      - Actual measurements of individual positive and
the sample group.
64 #                               "n_i"      - Number of members in the group (if the unit of
observation in the sample are individuals, then "n_i" must be equal to 1)
65 #
66 #      - datos_ant : Represents the data reference used to calibrate G_{j0}
67 #                               The data must be labelled by domains planned. It should contain
the following columns:
68 #                               "domplan" - Planned categories of domains
69 #                               "y_i"      - Positive real and individual measurements of
each group in the sample (when the units of observation are the groups, "y_i" should be
per capita measurement)
70 #
71 #      - domplan_N : Represents counts (or reference population) of the target
population, divided by the planned domains.
72 #                               Tagged data must be labelled by domains planned. It should
contain the following columns:
73 #                               "domplan" - Planned domains categories
74 #                               "N_j"      - Number of individuals in each population
planned domain
75 #
76 # Output:
77 #      total_domnoplan_sim      -   Matrix array of dimension "J x (3 + 2*D) x nSim"
with predictions for relevant quantities of planned and unplanned domains
78 #                               Column 1 - Indicator of the planned domains
79 #                               Column 2 - T_j (totals of the planned domains)
80 #                               Column 3 - N_j (composition of the planned domains)
81 #                               Column 4 to (4+D-1) - T^d_j (totals of unplanned
domains, such that T_j = sum T^d_j (over d))
82 #                               Column (4+D) to (3 + 2*D) - N^d_j (composition of

```

```

    unplanned domains, such that  $N_j = \sum N^d_j$  (over d))
83 #
84
85 #     Consulted domains
86 cualdomplan <- as.matrix(domplan_N[, "domplan"])
87 J <- length(cualdomplan)
88
89 #     Size unplanned domains
90 D <- length(colid_D)
91
92 #     Repository totals in "unplanned domains" with three categories
93 total_domnoplan_sim <- array(NA, c(J, 3+2*D, nSim))
94
95 #-----
96 #     validation
97 #-----
98
99 #     A. Parameter alphaDP
100 if(any(alphaDP <= (0*alphaDP))){
101     stop("Error in the specification of 'alphaDP'!!!")}
102
103 #     B. Parameters alpha_D and colid_D
104 if(ncol(alpha_D) != ncol(colid_D)){
105     stop("The dimensiones of 'alpha_D' and 'colid_D' are different!!!")}
106
107 #     C. parameter alpha_D
108 if(any(alpha_D <= (0*alpha_D))){
109     stop("Error in the specification of 'alpha_D'!!!")}
110
111 #     D. Parameter nSim
112 if(nSim <= 0){
113     stop("Error in the specification of 'nSim'!!!")}
114
115 #     E. Parameter inter
116 if(inter <= 0){
117     stop("Error in the specification of 'inter'!!!")}
118
119 #     F. Total domains unplanned
120 if(sum(datos[, colid_D]) != nrow(datos)) {
121     stop("Error in total of unplanned domains")}
122
123 if(any(rowSums(datos[, colid_D]) != 1)) {
124     stop("Error in total of unplanned domains")}
125
126 #     -----
127 #     Scanning planned domains
128 #     -----
129 j <- 1
130 for(j in 1:J){
131     #     Extraction of indexes in "datos_ant"
132     P_j_ant <- which(datos_ant[, "domplan"] == cualdomplan[j])
133     #     Extraction of the data in "datos_ant"
134     datos_j_ant <- datos_ant[P_j_ant, ]
135

```

```

136 # Extraction of indexes in "data" (i.e. sample S_j)
137 P_j <- which(datos[, "domplan"] == cualdomplan[j])
138 # Extraction of the data in "data"
139 datos_j <- datos[P_j, ]
140
141 # Population size in \mathcal{P}_{j}
142 P_j <- which(domplan_N[, "domplan"] == cualdomplan[j])
143 # Extraction of the data in "data"
144 N_j <- domplan_N[P_j, "N_j"]
145
146 # domain name the planned arrangement and constant data impute "n_j"
147 sim <- 1
148 for(sim in 1:nSim){
149   total_domnoplan_sim[j, 1, sim] <- cualdomplan[j]
150   total_domnoplan_sim[j, 3, sim] <- N_j
151 }
152
153 # ``conteo``
154 conteo_sal <- conteo(datos_j, N_j)
155
156 # Estimating the size of the population out of the sample (in "Stil_j")
157 N_S_j <- conteo_sal[[2]]
158
159 # Estimating the size of the population out of the sample (in "Stil_j")
160 N_Stil_j <- conteo_sal[[3]]
161
162 # Simulating of the composition of "N_Stil_j" between domains unplanned
163 domnoplan_composicion_sim <- domnoplan_composicion(datos_j, N_j, N_S_j, N_Stil_j,
164   colid_D, alpha_D[j, ], nSim)
165
166 # Simulating data "N_j" distributed in "domnoplan1"
167 N_S_domnoplan_j <- domnoplan_composicion_sim[[1]]
168 N_Stil_domnoplan_j_sim <- domnoplan_composicion_sim[[2]]
169 total_domnoplan_sim[j, c((3+D+1):(3+2*D)), ] <- matrix(t(N_S_domnoplan_j), D, nSim) +
170   N_Stil_domnoplan_j_sim
171
172 # ``unicstar``
173 unicstar_sal <- unicstar(datos_j)
174 ystar <- unicstar_sal[[2]][, "ystar"]
175
176 # ``pesos``
177 pesos_sal <- pesosDP(unicstar_sal, alphaDP[j])
178 rho <- pesos_sal[[1]]
179 phi <- pesos_sal[[2]]
180
181 # ``g0_licitacion``
182 g0_licitacion_sal <- g0_licitacion(datos_j_ant, inter, part)
183 #g0_licitacion_sal <- g0_licitacion(datos_j_ant, inter)
184
185 # Simulating the composition of "T_Stil_j" between domains unplanned
186 domnoplan_totalcomp_sim <- domnoplan_totalcomp(datos_j, rho, ystar, phi,
187   g0_licitacion_sal, N_Stil_domnoplan_j_sim, nSim, colid_D)
188
189 # Simulating data grouped "n_j" distributed "domnoplan1"

```

```

187     T_S_domnoplan_j <- domnoplan_totalcomp_sim[[1]]
188     T_Stil_domnoplan_j <- domnoplan_totalcomp_sim[[2]]
189     T_Stil_domnoplan_j_sim <- domnoplan_totalcomp_sim[[3]]
190     total_domnoplan_sim[j,c((3+1):(3+D)),] <- matrix(t(T_S_domnoplan_j),D,nSim) +
      T_Stil_domnoplan_j_sim
191
192     # Totals for planned domains
193     total_domnoplan_sim[j,c(2),] <- matrix(sum(T_S_domnoplan_j),1,nSim) +
      T_Stil_domnoplan_j
194
195   }
196
197   #-----
198   # Output
199   #-----
200   return(total_domnoplan_sim)
201
202   #
203   # -- End of "domnoplan.R"--
204 }
205
206 domnoplan_composicion <- function(datos_j,N_j,N_S_j,N_Stil_j,colid_D,alpha_D,nSim){
207   #
208   # This function simulates Monte Carlo samples from the predictive distribution of the
209   # vector  $\mathbf{N}_j$  across the  $D$  unplanned domains in a given planned
210   # domain  $j$ .
211   #
212   # Input:
213   #   datos_j           - Data matrix with features and number of
214   #                       individuals in the sample 'S_j'
215   #   N_j               - Number of individuals in 'P_j' (jth planned
216   #                       domain)
217   #   N_S_j             - Number of individuals in the sample 'S_j'
218   #   N_Stil_j          - Number of individuals out the sample 'S_j'
219   #   colid_D           - D-dimensional matrix array with the columns in
220   #                       'datos' that correspond to the indicator variables of the planned domains (those
221   #                       indicator variables represent a portion of 'datos')
222   #   alpha_D           - D-dimensional array with positive entries for
223   #                       the parameters of the multinomial-Dirichlet component for the composition across
224   #                       unplanned domains (NOTE: this one makes reference to a single planned domain)
225   #   nSim              - Number of Monte Carlo simulated replicates of
226   #                       the predictive distribution
227   #
228   # Output:
229   #   N_S_domnoplan     - Composition of the number of individuals in
230   #                       sample 'S_j' ('N_S_j') across the 'D' unplanned domains
231   #   domnoplan_composicion_sim - (1 x D x nSim) matrix with Monte Carlo samples
232   #                       of the predictive distribution of the composition of 'Stil_j'
233   #
234   # Size of unplanned domains
235   D <- length(colid_D)
236   dim_D_j <- dim(datos_j)
237

```

```

228 # Sample Repository
229 domnoplan_composicion_sim <- array(0,c(D,nSim))
230 colnames(domnoplan_composicion_sim) <- c(1:nSim)
231
232 # Computing sample counts
233 d <- 1
234 for(d in 1:D){
235     # Creating the table with members of the group number for each unplanned domain
236     N_S_domnoplan <- datos_j[, "n_i"] * datos_j[, colid_D[d]]
237     datos_j <- cbind(datos_j, N_S_domnoplan)
238 }
239
240 # Composition of unplanned domains in the sample "S_j"
241 N_S_domnoplan <- colSums(datos_j[, c((dim_D_j[2]+1):(dim_D_j[2]+D))])
242
243 if(N_Stil_j > 0){
244     # Parameters of the Dirichlet distribution for the proportions of "domnoplan"
245     alpha_D_new <- alpha_D + N_S_domnoplan
246
247     # Predictive simulation of multinomial-Dirichlet model
248     sim <- 1
249     for(sim in 1:nSim){
250         # Sample of the composition "p_domplan_j"
251         p_domnoplan_j <- randDirichlet(alpha_D_new, 1)
252
253         # Sample of the composition "N_Stil_domplan_j"
254         N_Stil_domnoplan_j <- t(rmultinom(1, N_Stil_j, p_domnoplan_j))
255
256         # Repository
257         domnoplan_composicion_sim[, sim] <- N_Stil_domnoplan_j
258     }
259 }
260
261 # Output
262 salida <- list(N_S_domnoplan, domnoplan_composicion_sim)
263 return(salida)
264
265 #
266 # -- End of domnoplan_composicion.R --
267 }
268
269 domnoplan_g0 <- function(datos, domplan_N, alphaDP, colid_D, alpha_D, nSim, g0_licitacion_sal){
270 #
271 # Generates Monte Carlo samples of the predictive distribution of totals of a finite
272 # population
273 # segmented in planned and unplanned domains, using a predefined set of
274 #  $\{G_{j0}\}_{j=1}^J$ ,
275 # along with simulations of the predictive distribution for the composition of the
276 # population
277 # between the unplanned domains.
278 #
279 # Input:
280 #     datos      - (Mxp)-dimensional array with positive entries for S_j
281 #     domplan_N  - Matrix array with counts of individuals in each planned domain

```

```

279 #      alphaDP      -   J-dimensional array with positive entries for the parameters of
the Dirichlet process for F_j (with J being the number of planned domains)
280 #      colid_D      -   D-dimensional matrix array with the columns in 'datos' that
correspond to the indicator variables of the planned domains (those indicator variables
represent a partion of 'datos')
281 #      alpha_D      -   (JxD)-dimensional array with positive entries for the
parameters of the multinomial-Dirichlet component for the composition across unplanned
domains.
282 #                               Note: Each one of the J rows is a vector of composition for P_j
segmented across D unplanned domains
283 #      nSim         -   Number of Monte Carlo simulated replicates of the predictive
distribution
284 # g0_licitacion_sal -   (Jx1) object list, each entry is another object list itself
associated with each G_{j0} for the J planned domains. The first element for arch
G_{j0} should be the name of the chosen 'distribution'(see details below for
alternatives), the second element should be a vector object with the parameters
associated with 'distribution', and the third element should be its associated
expectation
285 #
286 #
287 #   Details:
288 #       - datos : Represents the data sample of the target population, unplanned
domains labelled.
289 #                               It should contain the following columns:
290 #                               "domplan" - Categories fot planned domains.
291 #                               "y_i"      - Actual individual measurements/outcomes (for
the moment, they must be positive) for the group of observation.
292 #                               "n_i"      - Number of individuals in the group (if the unit
of observation in the sample are individuals, then "n_i" must be equal to 1)
293 #
294 #       - domplan_N : Represents counts (or reference population) of the target
population, divided by the planned domains.
295 #                               Tagged data must be labelled by domains planned. It should
contain the following columns:
296 #                               "domplan" - Categories for planned domains.
297 #                               "N_j"      - Number of individuals in each planned domain.
298 #
299 #       - g0_licitacion_sal: Chose one and only one of the distribution:
300 #                               i)      Gamma
301 #                               ii)     Weibull
302 #                               iii)    Lognormal
303 #                               iv)     Inverse-Gaussian
304 #
305 #                               Parameterization for distribution:
306 #
307 #                               i) Gamma      with parameters theta = c(alpha>0 , beta>0) and
density function
308 #
309 #                               
$$f(x) = x^{\alpha-1} \exp\{-x/\beta\}$$

310 #
311 #                               where alpha is the shape parameter, and beta is the
scale parameter.
312 #
313 #                               ii) Weibull  with parameters theta = c(alpha>0 , beta>0) and

```

```

density function
314 #
315 #            $f(x) = (x/\beta)^{\alpha-1} \exp\{-(x/\beta)^\alpha\}$ 
316 #
317 #           where alpha is the shape parameter, and beta is the
scale parameter.
318 #
319 #           iii) Lognormal with parameters  $\theta = c(\alpha > 0, \beta > 0)$  and
density function
320 #
321 #            $f(x) = \exp\{-(\log(x) - \alpha)^2 / (2\beta^2)\}$ 
322 #
323 #           where alpha is the mean, and beta is the standard
deviation of the logarithm.
324 #
325 #           iv) Inverse-Gaussian with parameters  $\theta = c(\alpha > 0,$ 
 $\beta > 0)$  and density function
326 #
327 #            $f(x) = \dots$ 
328 #
329 #           where alpha is the shape parameter, and beta is the
scale parameter.
330 #
331 #   Output:
332 #       total_domnoplan_sim      - Matrix array of dimension "J x (3 + 2*D) x nSim"
with predictions for relevant quantities of planned and unplanned domains
333 #           Column 1 - Indicator of the planned domains
334 #           Column 2 - T_j (totals of the planned domains)
335 #           Column 3 - N_j (composition of the planned domains)
336 #           Column 4 to (4+D-1) - T^d_j (totals of unplanned
domains, such that  $T_j = \sum T^d_j$  (over d))
337 #           Column (4+D) to (3 + 2*D) - N^d_j (composition of
unplanned domains, such that  $N_j = \sum N^d_j$  (over d))
338 #
339
340 #       Consulted domains
341 cualdomplan <- as.matrix(domplan_N[, "domplan"])
342 J <- length(cualdomplan)
343
344 #       Size unplanned domains
345 D <- length(colid_D)
346
347 #       Repository totals in "unplanned domains" with three categories
total_domnoplan_sim <- array(NA, c(J, 3+2*D, nSim))
349
350 #-----
351 #   validation
352 #-----
353
354 #   A. Parameter alphaDP
355 if(any(alphaDP <= (0*alphaDP))){
356     stop("Error in the specification of 'alphaDP'!!!")
357 }
358 #   B. Parameters alpha_D and colid_D

```



```

359   if(ncol(alpha_D) != ncol(colid_D)){
360       stop("The dimensiones of 'alpha_D' and 'colid_D' are different!!!")
361   }
362   #   C.   parameter alpha_D
363   if(any(alpha_D <= (0*alpha_D))){
364       stop("Error in the specification of 'alpha_D'!!!")
365   }
366   #   D.   Parameter nSim
367   if(nSim <= 0){
368       stop("Error in the specification of 'nSim'!!!")
369   }
370   #   F. Total domains unplanned
371   if(sum(datos[,colid_D]) != nrow(datos)) {
372       stop("Error in total of unplanned domains")
373   }
374   if(any(rowSums(datos[,colid_D]) != 1)) {
375       stop("Error in total of unplanned domains")
376   }
377   #   -----
378   #   Scanning planned domains
379   #   -----
380   j <- 1
381   for(j in 1:J){
382
383       #   Extraction of indexes in "data" (i.e. sample S_j)
384       P_j <- which(datos[, "domplan"] == cualdomplan[j])
385       #   Extraction of the data in "data"
386       datos_j <- datos[P_j, ]
387
388       #   Population size in \mathcal{P}_{j}
389       P_j <- which(domplan_N[, "domplan"] == cualdomplan[j])
390       #   Extraction of the data in "data"
391       N_j <- domplan_N[P_j, "N_j"]
392
393       #   domain name the planned arrangement and constant data impute "n_j"
394       sim <- 1
395       for(sim in 1:nSim){
396           total_domnoplan_sim[j,1,sim] <- cualdomplan[j]
397           total_domnoplan_sim[j,3,sim] <- N_j
398       }
399
400       #   ``conteo``
401       conteo_sal <- conteo(datos_j, N_j)
402
403       #   Estimating the size of the population out of the sample (in "Stil_j")
404       N_S_j <- conteo_sal[[2]]
405
406       #   Estimating the size of the population out of the sample (in "Stil_j")
407       N_Stil_j <- conteo_sal[[3]]
408
409       #   Simulating of the composition of "N_Stil_j" between domains unplanned
410       domnoplan_composicion_sim <- domnoplan_composicion(datos_j, N_j, N_S_j, N_Stil_j,
411           colid_D, alpha_D[j, ], nSim)

```

```

412 # Simulating data "N_j" distributed in "domnoplan1"
413 N_S_domnoplan_j <- domnoplan_composicion_sim[[1]]
414 N_Stil_domnoplan_j_sim <- domnoplan_composicion_sim[[2]]
415 total_domnoplan_sim[j,c((3+D+1):(3+2*D)),] <- matrix(t(N_S_domnoplan_j),D,nSim) +
  N_Stil_domnoplan_j_sim
416
417 # ``unicstar``
418 unicstar_sal <- unicstar(datos_j)
419 ystar <- unicstar_sal[[2]][,"ystar"]
420
421 # ``pesos``
422 pesos_sal <- pesosDP(unicstar_sal,alphaDP[j])
423 rho <- pesos_sal[[1]]
424 phi <- pesos_sal[[2]]
425
426
427 #-----
428 # Validation ``g0_licitacion_sal``
429 #-----
430 # i) Gamma
431 # ii) Weibull
432 # iii) Log-normal
433 # iv) Inverse-Gaussian
434
435 dis_sel <- list("Gamma","Weibull","Lognormal","Inverse-Gaussian")
436
437 # Validation of distribution
438
439 if(g0_licitacion_sal[[j]][[1]] == dis_sel[[1]] || g0_licitacion_sal[[j]][[1]] ==
dis_sel[[2]] || g0_licitacion_sal[[j]][[1]] == dis_sel[[3]] || g0_licitacion_sal[[j]
]][[1]] == dis_sel[[4]]){
440
441 }else{
442   stop("Error in the specification of 'distribution'!!!")
443
444 # Validation of parameters
445 if(g0_licitacion_sal[[j]][[2]][[1]]>0 & g0_licitacion_sal[[j]][[2]][[2]]>0){
446 }else{
447   stop("Error in the specification of 'parameters'!!!")
448
449 # Simulating the composition of "T_Stil_j" between domains unplanned
450 domnoplan_totalcomp_sim <- domnoplan_totalcomp(datos_j,rho,ystar,phi,
g0_licitacion_sal[[j]],N_Stil_domnoplan_j_sim,nSim,colid_D)
451
452 # Simulating data grouped "n_j" distributed "domnoplan1"
453 T_S_domnoplan_j <- domnoplan_totalcomp_sim[[1]]
454 T_Stil_domnoplan_j <- domnoplan_totalcomp_sim[[2]]
455 T_Stil_domnoplan_j_sim <- domnoplan_totalcomp_sim[[3]]
456 total_domnoplan_sim[j,c((3+1):(3+D)),] <- matrix(t(T_S_domnoplan_j),D,nSim) +
  T_Stil_domnoplan_j_sim
457
458 # Totals for planned domains
459 total_domnoplan_sim[j,c(2),] <- matrix(sum(T_S_domnoplan_j),1,nSim) +
  T_Stil_domnoplan_j

```

```

460
461     }
462
463     #-----
464     #   Output
465     #-----
466     return(total_domnoplan_sim)
467
468     #
469     #   -- End of "domnoplan_g0.R"--
470 }
471
472 domnoplan_totalcomp <- function(datos_j,rho,ystar,phi,g0_licitacion_sal,
N_Stil_domnoplan_j_sim,nSim,colid_D){
473     #
474     #   This function simulates Monte Carlo samples from the predictive distribution of the
vector \eqn{\mathbf{T}_j} across the \eqn{D} unplanned domains in a given planned
domain \eqn{j}.
475     #
476     #   Input:
477     #       datos_j           -   Data matrix with features and number of
individuals in the sample 'S_j'
478     #       rho               -   (Ux2)-dimensional vector with weights
associated with the sample ties 'ystar'
479     #       ystar             -   Sample ties 'y^*_i' in the sample 'S_j'
480     #       phi               -   Probability weight associated with "G_{j0}"
(the continuous part of Ghat_j)
481     #       g0_licitacion_sal -   Object list with the continuous component in
'Ghat'
482     #       N_Stil_domnoplan_j_sim -   (1 x D x nSim) matrix with Monte Carlo samples
of the predictive distribution of the composition of 'Stil_j'
483     #       nSim              -   Number of Monte Carlo simulated replicates of
the predictive distribution
484     #       colid_D           -   D-dimensional matrix array with the columns in
'datos' that correspond to the indicator variables of the planned domains (those
indicator variables represent a portion of 'datos')
485     #
486     #   Output:
487     #   Object list with three entries:
488     #       T_S_domnoplan     -   Composition of 'T_j' for the planned domain 'j'
in sample 'S_j'
489     #       T_Stil_domnoplan_j -   (1xDxnSim)-dimensional array with simulated
samples of the convolution for 'T_Stil_j'
490     #       domnoplan_totalcomp_sim -   (1xDxnSim)-dimensional array with samples from
the predictive distribution of the composition of 'Stil_j'
491     #
492
493     #   Number of unplanned domains
494     D <- length(colid_D)
495     dim_D_j <- dim(datos_j)
496
497     #   Repository: Unplanned domains "T_Stil_domnoplan_d_j"
498     domnoplan_totalcomp_sim <- array(NaN,c(D,nSim))
499

```

```

500 # Repository: Non-sampled total the domain j - "T_Stil_domnoplan_j"
501 T_Stil_domnoplan_j <- array(NA,nSim)
502 rownames(T_Stil_domnoplan_j) <- c("T_Stil_domnoplan_j")
503 colnames(T_Stil_domnoplan_j) <- c(1:nSim)
504
505 # Computing sample totals
506 d <- 1
507 for(d in 1:D){
508   # vector with number of members of the group for each unplanned domains
509   T_S_domnoplan <- datos_j[, "n_i"] * datos_j[, "y_i"] * datos_j[, colid_D[d]]
510   datos_j <- cbind(datos_j, T_S_domnoplan)
511 }
512
513 # Composition of the total across domains on the sample unplanned "S_j"
514 T_S_domnoplan <- colSums(datos_j[, c((dim_D_j[2]+1):(dim_D_j[2]+D))])
515
516 # Simulations
517 sim <- 1
518 d <- 1
519 for(sim in 1:nSim){
520   for(d in 1:D){
521     # Extract the auxiliary size N_Stil_domnoplan1_d
522     N_d_aux <- N_Stil_domnoplan_j_sim[d,sim]
523
524     if(N_d_aux > 0){
525       # simulating a sample of size "N_d_aux" of "ghat"
526       y_d_aux <- ghat_simulacion(rho, ystar, phi, g0_licitacion_sal, N_d_aux)
527
528       # Simulation subtotal "T_Stil_domnoplan1_d"
529       domnoplan_totalcomp_sim[d,sim] <- sum(y_d_aux)
530     }else{
531       # Sum of the subtotal "T_Stil_domnoplan1_d" zero
532       domnoplan_totalcomp_sim[d,sim] <- 0
533     }
534   }
535   T_Stil_domnoplan_j[sim] <- sum(domnoplan_totalcomp_sim[,sim])
536 }
537
538 # Output
539 salida <- list(T_S_domnoplan, T_Stil_domnoplan_j, domnoplan_totalcomp_sim)
540 return(salida)
541
542 #
543 # -- END of domnoplan_totalcomp.R --
544 }
545
546 domplan <- function(datos, datos_ant, domplan_N, alphaDP, inter, part, nSim){
547 #
548 # Generates Monte Carlo samples of the predictive distribution of totals of a finite
549 # population segmented in planned domains.
550 #
551 # Input:
552 #   datos      - p-dimensional vector with positive entries
553 #   datos_ant  - Reference data for calibration of G_0

```

```

553 #      domplan_N    -   Free dimension matrix array represents counts
554 #      alphaDP      -   J-dimensional positive parameter for the Dirichlet process
555 #      inter        -   Tuning parameter for model comparison and selection
556 #      part         -   Number of partitions for predictive cross-validation
557 #      nSim         -   Number of simulated replicates
558 #
559 #   Details:
560 #       - datos : Represents the data sample of the target population, unplanned
561 #       domains labelled.
562 #           It should contain the following columns:
563 #           "domplan" - Planned domains categories.
564 #           "y_i"     - Actual measurements of individual positive and
565 #           the sample group.
566 #           "n_i"     - Number of members in the group (if the unit of
567 #           observation in the sample are individuals, then "n_i" must be equal to 1)
568 #       - datos_ant : Represents the data of reference used to calibrate G_{0}.
569 #       The data must be labelled by domains planned. It should contain
570 #       the following columns:
571 #       "domplan" - Planned categories of domains
572 #       "y_i"     - Positive real and individual measurements of
573 #       each group in the sample (when the units of observation are groups, "y_i" should be a
574 #       per capita measurement)
575 #
576 #       - domplan_N : Represents counts (or population frame) of the target
577 #       population, divided by the planned domains.
578 #       Tagged data must be labelled by domains planned. It should
579 #       contain the following columns:
580 #       "domplan" - Planned domains categories
581 #       "N_j"     - Number of individuals in each population
582 #       planned domain
583 #
584 #   Output:
585 #       total_domplan_sim - "(J x 3 x nSim)" Matrix array dimension with the
586 #       predictions of the planned domains
587 #       Column 1 - Indicator of the planned domains
588 #       Column 2 - T_j (totals of the planned domains)
589 #       Column 3 - N_j (composition of the planned domains)
590 #
591 #   Definition consulted domains
592 #   cualdomplan <- as.matrix(domplan_N[, "domplan"])
593 #   J <- length(cualdomplan)
594 #
595 #   Repository totals in "unplanned domains" with three categories
596 #   total_domplan_sim <- array(NA, c(J, 3, nSim))
597 #
598 #-----
599 #   Validation
600 #-----
601 #
602 #   A. Parameter alphaDP
603 #   if(any(alphaDP <= (0*alphaDP))){

```

```

596     stop("Error in the specification of 'alphaDP'!!!")
597
598   #   B.   Parameter nSim
599   if(nSim <= 0){
600     stop("Error in the specification of 'nSim'!!!")
601
602   #   C.   Parameter inter
603   if(inter <= 0){
604     stop("Error in the specification of 'inter'!!!")
605
606   #   -----
607   #   Scanning planned domains
608   #   -----
609   j <- 1
610   for(j in 1:J){
611     #   Extraction of indexes in "datos_ant"
612     P_j_ant <- which(datos_ant[, "domplan"] == cualdomplan[j])
613     #   Extraction of the data in "datos_ant"
614     datos_j_ant <- datos_ant[P_j_ant, ]
615
616     #   Extraction of indexes in "data" (ie sample S_j)
617     P_j <- which(datos[, "domplan"] == cualdomplan[j])
618     #   Extraction of the data in "data"
619     datos_j <- datos[P_j, ]
620
621     #   Population size of  $P_{j}$ 
622     P_j <- which(domplan_N[, "domplan"] == cualdomplan[j])
623     #   Extraction of the data in "data"
624     N_j <- domplan_N[P_j, "N_j"]
625
626     #   Name the planned domain grooming and constant data impute "n_j"
627     for(sim in 1:nSim){
628       total_domplan_sim[j, 1, sim] <- cualdomplan[j]
629       total_domplan_sim[j, 3, sim] <- N_j
630     }
631
632     #   ``conteo``
633     conteo_sal <- conteo(datos_j, N_j)
634
635     #   Estimating the size of the population outside of the sample (in "Stil_j")
636     N_S_j <- conteo_sal[[2]]
637
638     #   Estimating the size of the population outside of the sample (in "Stil_j")
639     N_Stil_j <- conteo_sal[[3]]
640
641     if(N_Stil_j > 0){
642       #   ``unicstar``
643       unicstar_sal <- unicstar(datos_j)
644       ystar <- unicstar_sal[[2]][, "ystar"]
645
646       #   ``pesos``
647       pesos_sal <- pesosDP(unicstar_sal, alphaDP[j])
648       rho <- pesos_sal[[1]]
649       phi <- pesos_sal[[2]]

```

```

650
651     # ``g0_licitacion''
652     g0_licitacion_sal <- g0_licitacion(datos_j_ant,inter,part)
653     #g0_licitacion_sal <- g0_licitacion(datos_j_ant,inter)
654
655     # Simulation the T_Stil_j
656     domplan_total_sim <- domplan_total(datos_j,rho,ystar,phi,g0_licitacion_sal,
        N_Stil_j,nSim)
657
658     # Simulating data grouped "n_j" distributed "domnoplan1"
659     T_S_domplan_j <- domplan_total_sim[[1]]
660     T_Stil_domplan_j <- domplan_total_sim[[2]]
661     total_domplan_sim[j,2,] <- matrix(t(T_S_domplan_j),1,nSim) + T_Stil_domplan_j
662   }else if(N_Stil_j ==0){
663     # Calculating the total "T_S_j" compositional in S_j (within the sample)
664     T_S_j_gpo <- datos_j[, "n_i"] * datos_j[, "y_i"]
665
666     # Composition of the total between domains on the sample unplanned "S_j"
667     T_S_domplan_j <- sum(T_S_j_gpo)
668
669     # Simulating data store "n_j" distributed "domnoplan1"
670     total_domplan_sim[j,2,] <- matrix(t(T_S_domplan_j),1,nSim)
671   }
672
673 }
674
675 #-----
676 # Output
677 #-----
678 return(total_domplan_sim)
679
680 #
681 # -- END of "domplan.R"--
682 }
683
684 domplan_g0<- function(datos,domplan_N,alphaDP,nSim,g0_licitacion_sal){
685   #
686   # Generates Monte Carlo samples of the predictive distribution of totals of a finite
        population segmented in planned domains, using a predefined set of
        \eqn{(G_{j0})_{j=1}^J}.
687   #
688   # Input:
689   #   datos           - (Mxp)-dimensional array with positive entries for {S}_j
690   #   domplan_N       - Matrix array with counts of individuals in each planned
        domain
691   #   alphaDP         - J-dimensional array with positive entries for the
        parameters of the Dirichlet process for F_j (with J being the number of planned domains)
692   #   nSim             - Number of Monte Carlo simulated replicates of the
        predictive distribution
693   #   g0_licitacion_sal - (Jx1) object list, each entry is another object list
        itself associated with each G_{j0} for the J planned domains. The first element for
        arch G_{j0} should be the name of the chosen distribution(see details below for
        alternatives), the second element should be a vector object with the parameters
        associated with distribution, and the third element should be its associated expectation.

```

```

694 #
695 #   Details:
696 #       - datos : Represents the data sample of the target population, unplanned
domains labelled.
697 #           It should contain the following columns:
698 #               "domplan" - Categories fot planned domains.
699 #               "y_i"      - Actual individual measurements/outcomes (for
the moment, they must be positive) for the group of observation.
700 #               "n_i"      - Number of individuals in the group (if the unit
of observation in the sample are individuals, then "n_i" must be equal to 1)
701 #
702 #       - domplan_N : Represents counts (or reference population) of the target
population, divided by the planned domains.
703 #           Tagged data must be labelled by domains planned. It should
contain the following columns:
704 #               "domplan" - Categories for planned domains.
705 #               "N_j"      - Number of individuals in each planned domain.
706 #
707 #       - g0_licitacion_sal: Chose one and only one of the distribution:
708 #           i)      Gamma
709 #           ii)     Weibull
710 #           iii)    Lognormal
711 #           iv)     Inverse-Gaussian
712 #
713 #           Parameterization for distribution:
714 #
715 #           i) Gamma      with parameters theta = c(alpha>0 , beta>0) and
density function
716 #
717 #                $f(x) = x^{\alpha-1} \exp\{-x/\beta\}$ 
718 #
719 #               where alpha is the shape parameter, and beta is the
scale parameter.
720 #
721 #           ii) Weibull  with parameters theta = c(alpha>0 , beta>0) and
density function
722 #
723 #                $f(x) = (x/\beta)^{\alpha-1} \exp\{-(x/\beta)^\alpha\}$ 
724 #
725 #               where alpha is the shape parameter, and beta is the
scale parameter.
726 #
727 #           iii) Lognormal with parameters theta = c(alpha>0 , beta>0) and
density function
728 #
729 #                $f(x) = \exp\{-(\log(x)-\alpha)^2/(2*\beta^2)\}$ 
730 #
731 #               where alpha is the mean, and beta is the standard
deviation of the logarithm.
732 #
733 #           iv) Inverse-Gaussian with parameters theta = c(alpha>0 ,
beta>0) and density function
734 #
735 #               f(x)= .....

```



```

736 #
737 #                               where alpha is the shape parameter, and beta is the
scale parameter.
738 #
739 #
740 #   Output:
741 #       total_domplan_sim   - "(J x 3 x nSim)" Matrix array dimension with the
predictions of the planned domains
742 #                               Column 1 - Indicator of the planned domains
743 #                               Column 2 - T_j (totals of the planned domains)
744 #                               Column 3 - N_j (composition of the planned domains)
745 #
746
747 #   Definition consulted domains
748 cualdomplan <- as.matrix(domplan_N[, "domplan"])
749 J <- length(cualdomplan)
750 #   Repository totals in "unplanned domains" with three categories
751 total_domplan_sim <- array(NA, c(J, 3, nSim))
752
753 #-----
754 #   Validation
755 #-----
756
757 #   A.   Parameter alphaDP
758 if(any(alphaDP <= (0*alphaDP))){
759     stop("Error in the specification of 'alphaDP'!!!")}
760
761 #   B.   Parameter nSim
762 if(nSim <= 0){
763     stop("Error in the specification of 'nSim'!!!")}
764
765 #   -----
766 #   Scanning planned domains
767 #   -----
768 j <- 1
769 for(j in 1:J){
770
771     #   Extraction of indexes in "data" (ie sample S_j)
772     P_j <- which(datos[, "domplan"] == cualdomplan[j])
773     #   Extraction of the data in "data"
774     datos_j <- datos[P_j, ]
775
776     #   Population size of  $P_j$ 
777     P_j <- which(domplan_N[, "domplan"] == cualdomplan[j])
778     #   Extraction of the data in "data"
779     N_j <- domplan_N[P_j, "N_j"]
780
781     #   Name the planned domain grooming and constant data impute "n_j"
782     for(sim in 1:nSim){
783         total_domplan_sim[j, 1, sim] <- cualdomplan[j]
784         total_domplan_sim[j, 3, sim] <- N_j
785     }
786
787     #   ``conteo''

```

```

788     conteo_sal <- conteo(datos_j,N_j)
789
790     # Estimating the size of the population outside of the sample (in "Stil_j")
791     N_S_j <- conteo_sal[[2]]
792
793     # Estimating the size of the population outside of the sample (in "Stil_j")
794     N_Stil_j <- conteo_sal[[3]]
795
796     if(N_Stil_j >0){
797         # ``unicstar''
798         unicstar_sal <- unicstar(datos_j)
799         ystar <- unicstar_sal[[2]][,"ystar"]
800
801         # ``pesos''
802         pesos_sal <- pesosDP(unicstar_sal,alphaDP[j])
803         rho <- pesos_sal[[1]]
804         phi <- pesos_sal[[2]]
805
806         #-----
807         # Validation ``g0_licitacion_sal''
808         #-----
809         # i) Gamma
810         # ii) Weibull
811         # iii) Log-normal
812         # iv) Inverse-Gaussian
813
814         dis_sel <- list("Gamma","Weibull","Lognormal","Inverse-Gaussian")
815
816         # Validation of distribution
817
818         if(g0_licitacion_sal[[j]][[1]] == dis_sel[[1]] || g0_licitacion_sal[[j]][[1]] ==
819         dis_sel[[2]] || g0_licitacion_sal[[j]][[1]] == dis_sel[[3]] || g0_licitacion_sal[[j]
820         ][[1]] == dis_sel[[4]]){
821         }else{
822             stop("Error in the specification of 'distribution'!!!")
823
824         # Validation of parameters
825         if(g0_licitacion_sal[[j]][[2]][[1]]>0 & g0_licitacion_sal[[j]][[2]][[2]]>0){
826         }else{
827             stop("Error in the specification of 'parameters'!!!")
828
829         # Simulation the T_Stil_j
830         domplan_total_sim <- domplan_total(datos_j,rho,ystar,phi,g0_licitacion_sal[[j]],
831         N_Stil_j,nSim)
832
833         # Simulating data grouped "n_j" distributed "domnoplan1"
834         T_S_domplan_j <- domplan_total_sim[[1]]
835         T_Stil_domplan_j <- domplan_total_sim[[2]]
836         total_domplan_sim[j,2,] <- matrix(t(T_S_domplan_j),1,nSim) + T_Stil_domplan_j
837     }else if(N_Stil_j ==0){
838         # Calculating the total "T_S_j" compositional in S_j (within the sample)
839         T_S_j_gpo <- datos_j[, "n_i"] * datos_j[, "y_i"]

```

```

838      # Composition of the total between domains on the sample unplanned "S_j"
839      T_S_domplan_j <- sum(T_S_j_gpo)
840
841      # Simulating data store "n_j" distributed "domnoplan1"
842      total_domplan_sim[j,2,] <- matrix(t(T_S_domplan_j),1,nSim)
843      }
844
845  }
846
847  #-----
848  # Output
849  #-----
850  return(total_domplan_sim)
851
852  #
853  # -- END of "domplan_g0.R"--
854  }
855
856  domplan_total <- function(datos_j,rho,ystar,phi,g0_licitacion_sal,N_Stil_j,nSim){
857  #
858  # Simulates Monte Carlo samples of size ' $\{nSim\}$ ' of the final distribution for
859  # the total for the given planned domain ' $\mathcal{P}_j$ '.
860  # Input:
861  #      datos_j          - Data matrix with features and number of
862  #                        individuals in the sample 'S_j'
863  #      rho              - (Ux2)-dimensional vector with weights
864  #                        associated with the sample ties 'ystar'
865  #      ystar            - Sample ties ' $y^*_i$ ' in the sample 'S_j'
866  #      phi              - Probability weight associated with ' $G_{j0}$ '
867  #                        (the continuous part of Ghat_j)
868  #      g0_licitacion_sal - Object list with the continuous component in
869  #                        'Ghat'
870  #      N_Stil_sal       - Composition of individuals out of 'S_j'
871  #      nSim             - Number of Monte Carlo simulated replicates of
872  #                        the predictive distribution
873  #
874  # Output:
875  # Object list with two entries:
876  #      T_S_j            - Composition of 'T_j' across planned domains in
877  #                        the sample 'S_j'
878  #      domplan_total_sim - (1 x nSim) matrix with samples from the
879  #                        predictive distribution of the total 'T_Stil_j'
880  #
881  # Repository: Total non-sampling domain j -- "T_Stil_domnoplan_j"
882  T_Stil_domnoplan_j <- array(NA,c(1,nSim))
883  rownames(T_Stil_domnoplan_j) <- c("T_Stil_domnoplan_j")
884  colnames(T_Stil_domnoplan_j) <- c(1:nSim)
885
886  # Computing the total "T_S_j" compositional in "S_j" (within the sample)
887  # Vector with the totals of each group
888  T_S_j_gpo <- datos_j[, "n_i"] * datos_j[, "y_i"]
889  datos_j <- cbind(datos_j, T_S_j_gpo)

```

```

884
885 # Composition of the total between domains on the sample unplanned "S_j"
886 T_S_j <- sum(datos_j[, "T_S_j_gpo"])
887
888 # Simulations
889 sim <- 1
890 for(sim in 1:nSim){
891   # Computing the sample of size "N_Stil_j" of "ghat"
892   y_i_aux <- ghat_simulacion(rho, ystar, phi, g0_licitacion_sal, N_Stil_j)
893
894   # Simulation subtotal "T_Stil_j"
895   T_Stil_domnplan_j[sim] <- sum(y_i_aux)
896 }
897
898 # Output
899 salida <- list(T_S_j, T_Stil_domnplan_j)
900 return(salida)
901
902 #
903 # -- END of domplan_total.R --
904 }
905
906 g0_licitacion <- function(datos_ant, inter, part){
907   # This function computes the prior elicitation of  $G_{\{j0\}}$  using reference data
   for the planned domain  $\{j\}$ .
908   #  $G_{\{j0\}}$  is used by the SSM as the baseline function. The distribution is
   elicited using a predictive cross-validation
909   # procedure for model comparison and selection among the following alternatives:
   Gamma, Weibull, Lognormal and Inverse-Gaussian.
910   # The function also computes the expectation of the chose distribution.
911   #
912   # input:
913   #       datos_ant - Reference data for calibration of G_0
914   #       inter    - Tuning parameter for model comparison and selection
915   #
916   #
917   # output: List object:
918   #       'distribution' - String object for 'distribution'
919   #                       i) Gamma
920   #                       ii) Weibull
921   #                       iii) Log-normal
922   #                       iv) Inverse-Gaussian
923   #
924   #       theta      - Vector of parameters associated with 'distribution'
925   #       mu         - Expected value of 'distribution'
926   #
927
928   y <- datos_ant[, "y_i"]
929   n <- length(y)
930   n1 <- n/20
931   # Vector with assessments of the 20 groups with missing observations
932   g0LogN20 <- rep(NA, 20)
933   g0Gam20 <- rep(NA, 20)
934   g0Wei20 <- rep(NA, 20)

```

```

935     g0IGau20 <- rep(NA,20)
936     # Vector with assessments of each observation using the estimated parameters with the
    observations that were omitted
937     g0LogNN <- rep(NA,n)
938     g0GamN  <- rep(NA,n)
939     g0Wein  <- rep(NA,n)
940     g0IGauN <- rep(NA,n)
941     # initial parameters
942     sigmaLN0c <- rnorm(1,0,1)
943     muLN0c    <- rnorm(1,0,1)
944     nu1G0c    <- rnorm(1,0,1)
945     nu2G0c    <- rnorm(1,0,1)
946     nu1W0c    <- rnorm(1,0,1)
947     nu2W0c    <- rnorm(1,0,1)
948     nu1IG0c   <- rnorm(1,0,1)
949     nu2IG0c   <- rnorm(1,0,1)
950     # Randomization group
951     y <- sample(y)
952     # Vector with 20 parameters and parameter groups of the entire group
953     muLN0 <- rep(NA,21)
954     sigLN0 <- rep(NA,21)
955     shG0 <- rep(NA,21)
956     scG0 <- rep(NA,21)
957     shW0 <- rep(NA,21)
958     scW0 <- rep(NA,21)
959     shIG0 <- rep(NA,21)
960     scIG0 <- rep(NA,21)
961
962     i <- 1
963     for (i in 1:20)
964     {
965         # Ranges to form 20 groups
966         i5 <- round((i-1)*n1)+1
967         i6 <- round(i*n1)
968         i7 <- i6+1
969         if (i6 <= n)
970         {
971             for (j in (i5:i6))
972             {
973                 # Missing observations according to previous ranks
974                 y[j] <- NA
975             }
976         }else{for (j in i7:n){y[j] <- NA}}
977         y <- na.omit(y)
978
979         # Calculations prior to the estimation of Weibull
    parameters
980         nOmiOrd <- length(sort(y))
981         x00 <- c(1:nOmiOrd)
982         x0 <- log(log(1/(1-(x00/(nOmiOrd+1))))))
983         x1 <- (1/nOmiOrd)*sum(x0)
984         y1 <- mean(log(sort(y)))
985
986         # ESTIMATES IN EACH GROUP:

```

```

987
988   # Mu de X ~ Normal, where
989   X=log(Y)
990   muLN0[i] <- sum(log(y))/length(y)
991   sigLN0[i] <- sqrt(sum((log(y)-muLN0[i])^2)/length(y))
992   # Form parameter of Gamma
993   shG0[i] <- 0.5/(log(mean(y))-mean(log(y)))
994   # Scale parameter of Gamma
995   scG0[i] <- mean(y)/shG0[i]
996   # Form parameter of Weibull
997   shW0[i] <- ((nOmiOrd*(sum((log(sort(y))*(x0)))-(sum(x0)*sum(log(sort(y)))))/((
1000   nOmiOrd*sum(log(sort(y))^2))-((sum(log(sort(y)))^2))
1001   # Scale parameter of Weibull
1002   scW0[i] <- exp(y1-(x1/shW0[i]))
1003   # Parameter "mu" of the Inverse Gaussian (not a location parameter)
1004   shIG0[i] <- mean(y)
1005   # Parameter "sigma" of the Inverse Gaussian (not a scale parameter)
1006   scIG0[i] <- (sum((1/y)-(1/shIG0[i])))/length(y)
1007
1008   y <- datos_ant[, "y_i"]
1009
1010   g0LogNPaso <- rep(NA,n)
1011   g0GamPaso <- rep(NA,n)
1012   g0WeiPaso <- rep(NA,n)
1013   g0IGauPaso <- rep(NA,n)
1014
1015   for (j in (i5:i6))
1016   {
1017     # Conditional densities
1018     g0LogNPaso[j] <- (pnorm(log(y[j]+inter),mean=muLN0[i],sd=sigLN0[i])-pnorm(log(
1019     max(y[j]-inter,0),mean=muLN0[i],sd=sigLN0[i]))/(2*inter)
1020     g0GamPaso[j] <- (pgamma(y[j]+inter,shape=shG0[i],scale=scG0[i])-pgamma(max(y[j]
1021     ]-inter,0),shape=shG0[i],scale=scG0[i]))/(2*inter)
1022     g0WeiPaso[j] <- (pweibull(y[j]+inter,shape=shW0[i],scale=scW0[i])-pweibull(max(
1023     y[j]-inter,0),shape=shW0[i],scale=scW0[i]))/(2*inter)
1024     g0IGauPaso[j] <- (pinvgauss(y[j]+inter,shIG0[i],scIG0[i])-pinvgauss(max(y[j]-
1025     inter,0.000001),shIG0[i],scIG0[i]))/(2*inter)
1026     # Vectors with assessments of the "n" observations
1027     g0LogNN[j] <- g0LogNPaso[j]
1028     g0GamN[j] <- g0GamPaso[j]
1029     g0WeiN[j] <- g0WeiPaso[j]
1030     g0IGauN[j] <- g0IGauPaso[j]
1031     # Vectors with assessments of the observations of the group "i" (20 groups)
1032     g0LogNsinNA <- na.omit(g0LogNPaso)
1033     g0GamsinNA <- na.omit(g0GamPaso)
1034     g0WeisinNA <- na.omit(g0WeiPaso)
1035     g0IGausinNA <- na.omit(g0IGauPaso)
1036   }
1037   # Vectors with the sum of the logarithms of the evaluations of the observations of
1038   the group "i"
1039   g0LogN20[i] <- sum(log(g0LogNsinNA))
1040   g0Gam20[i] <- sum(log(g0GamsinNA))
1041   g0Wei20[i] <- sum(log(g0WeisinNA))

```

```

1034     g0IGau20[i] <- sum(log(g0IGausinNA))
1035   }
1036   # Average of the 20 sums of logarithms
1037   criLN <- mean(g0LogN20)
1038   criG  <- mean(g0Gam20)
1039   criW  <- mean(g0Wei20)
1040   criIG <- mean(g0IGau20)
1041
1042   # ESTIMATE FOR THE WHOLE SAMPLE :
1043
1044   # Previous calculations for Weibull parameter
estimation
1045   x00 <- c(1:n)
1046   x0  <- log(log(1/(1-(x00/(n+1))))))
1047   x1  <- (1/n)*sum(x0)
1048   y1  <- mean(log(y))
1049
1050   # Mu of Y ~ Lognormal (mu_{y})
1051   muLN0[21] <- sum(log(y))/length(y)
1052   # Sigma of Y ~ Lognormal (sigma_{y})
1053   sigLN0[21] <- sqrt(sum((log(y)-muLN0[21])^2)/length(y))
1054   # Form parameter of Gamma
1055   shG0[21] <- 0.5/(log(mean(y))-mean(log(y)))
1056   # Scale parameter Gamma
1057   scG0[21] <- mean(y)/shG0[21]
1058   #Form parameter of Weibull
1059   shW0[21] <- ((n*(sum((log(sort(y))*(x0))))-(sum(x0)*sum(log(sort(y)))))/((n*sum(log(
sort(y))^2))-((sum(log(sort(y)))^2))
1060   # Scale parameter of Weibull
1061   scW0[21] <- exp(y1-(x1/shW0[21]))
1062   # Parameter "mu" of the Inverse Gaussian (not a location parameter)
1063   shIG0[21] <- mean(y)
1064   # Parameter "sigma" of the Inverse Gaussian (not a scale parameter)
1065   scIG0[21] <- (sum((1/y)-(1/shIG0[21])))/length(y)
1066
1067   # COMPARISON CRITERIA:
1068
1069   # Including all criteria
1070   #MaxCri <- max(criLN, criG, criW, criIG)
1071   # We removed the Gamma distribution criteria
1072   MaxCri <- max(criLN, criW, criIG)
1073   # Selected Distribution
1074   DenSel <- NA
1075   # Parameters of the selected distribution
1076   ParSel <- 0
1077   # Expectation of selected distribution
1078   MuSel <- 0
1079   if((MaxCri==criLN)) {DenSel="Lognormal"
1080   ParSel <- c(muLN0[21],sigLN0[21])
1081   MuSel <- exp(muLN0[21]+(0.5*(sigLN0[21]^2)))}else{if((MaxCri==criW)) {DenSel="Weibull"
1082   ParSel <- c(shW0[21],scW0[21])
1083   MuSel <- shG0[21]*scG0[21]}else{if(MaxCri==criG) {DenSel="Gamma"
1084   ParSel <- c(shG0[21],scG0[21])
1085   MuSel <- shG0[21]*scG0[21]}else{if(MaxCri==criIG) {DenSel="Inverse-Gaussian"

```

```

1086   ParSel <- c(shIG0[21],scIG0[21])
1087   MuSel <- shIG0[21]}else{DenSel="Adjustment is not available"}}}}
1088
1089   g0_licitacion_sal <- list(DenSel,ParSel,MuSel)
1090
1091   #   Output:
1092   salida <- g0_licitacion_sal
1093   return(salida)
1094
1095   #   END of g0_licitacion.R
1096   }
1097
1098   g0_simulacion <- function(g0_licitacion_sal,nSim){
1099   #
1100   #   This function computes Monte Carlo samples from a given continuous \eqn{G_{j0}}.
1101   #
1102   #   Input:
1103   #       g0_licitacion_sal   -   Object list with three elements (produced with
1104   #                               'g0_licitacion'):
1105   #                               a) String object for 'distribution'
1106   #                                   i) Gamma
1107   #                                   ii) Weibull
1108   #                                   iii) Log-normal
1109   #                                   iv) Inverse-Gaussian
1110   #                               b) theta           -   Vector of parameters associated
1111   #                               c) mu             -   Expected value of
1112   #                               'distribution'
1113   #       nSim                -   Number of Monte Carlo simulations
1114   #
1115   #   Output:
1116   #       g0_simulacion_sal   -   (nSim x 1) dimensional array with simulated data.
1117   #
1118   #   Simulations case-particularl
1119   if(g0_licitacion_sal[[1]]=='Lognormal'){
1120     g0_simulacion_sal <- as.matrix(rlnorm(nSim, meanlog = g0_licitacion_sal[[2]][1],
1121     sdlog = g0_licitacion_sal[[2]][2]))
1122     colnames(g0_simulacion_sal) <- c("ind_sim")
1123   }else if(g0_licitacion_sal[[1]]=='Gamma'){
1124     g0_simulacion_sal <- as.matrix(rgamma(nSim, shape = g0_licitacion_sal[[2]][1], scale
1125     = g0_licitacion_sal[[2]][2]))
1126     colnames(g0_simulacion_sal) <- c("ind_sim")
1127   }else if(g0_licitacion_sal[[1]]=='Weibull'){
1128     g0_simulacion_sal <- as.matrix(rweibull(nSim, shape = g0_licitacion_sal[[2]][1],
1129     scale = g0_licitacion_sal[[2]][2]))
1130     colnames(g0_simulacion_sal) <- c("ind_sim")
1131   }else if (g0_licitacion_sal[[1]]=='Inverse-Gaussian'){
1132     g0_simulacion_sal <- as.matrix(rinvgauss(nSim, g0_licitacion_sal[[2]][1],
1133     g0_licitacion_sal[[2]][2]))
1134     colnames(g0_simulacion_sal) <- c("ind_sim")
1135   }
1136   }

```



```

1133 # Output
1134 salida <- g0_simulacion_sal
1135 return(salida)
1136
1137 #
1138 # -- END of g0_simulacion.R --
1139 }
1140
1141 #
1142 # Generalized inverse Gaussian distribution
1143 #
1144 # Package:    rmutil
1145 # Version:    1.0
1146 # Title:      Utilities for Nonlinear Regression and Repeated Measurements Models
1147 # Author:     Jim Lindsey <jlindsey@luc.ac.be>
1148 #
1149
1150 pginvgauss <- function(q, m, s, f){
1151 # Generalized inverse Gaussian distribution - Distribution
1152 #
1153 # Package:    rmutil
1154 # Version:    1.0
1155 # Title:      Utilities for Nonlinear Regression and Repeated Measurements Models
1156 # Author:     Jim Lindsey <jlindsey@luc.ac.be>
1157 if(any(q<=0)){stop("q must contain positive values")}
1158 if(any(m<=0)){stop("m must be positive")}
1159 if(any(s<=0)){stop("s must be positive")}
1160 len <- max(length(q),length(m),length(s))
1161 if(length(q)!=len){
1162     if(length(q)==1)q <- rep(q,len)
1163     else stop("length of q incorrect")}
1164 if(length(m)!=len){
1165     if(length(m)!=1)stop("m has incorrect length")
1166     else m <- rep(m,len)}
1167 if(length(s)!=len){
1168     if(length(s)!=1)stop("s has incorrect length")
1169     else s <- rep(s,len)}
1170 if(length(f)!=len){
1171     if(length(f)!=1)stop("f has incorrect length")
1172     else f <- rep(f,len)}
1173 z <- .C("pginvgauss",
1174     as.double(q),
1175     as.double(m),
1176     as.double(s),
1177     as.double(f),
1178     len=as.integer(len),
1179     eps=as.double(1.0e-6),
1180     pts=as.integer(5),
1181     max=as.integer(16),
1182     err=integer(1),
1183     res=double(len),
1184     DUP=FALSE,
1185     PACKAGE="rmutil")
1186 if(z$serr==1){warning("Unable to allocate memory for integration")}

```

```

1187   if(z$err==2){warning("Division by zero in integration")}
1188   else if(z$err==3){warning("No convergence in integration")}
1189   return(z$res)
1190 }
1191
1192 dginvgauss <- function(y, m, s, f, log=FALSE){
1193   # Generalized inverse Gaussian distribution - Density
1194   #
1195   # Package:    rmutil
1196   # Version:    1.0
1197   # Title:      Utilities for Nonlinear Regression and Repeated Measurements Models
1198   # Author:     Jim Lindsey <jlindsey@luc.ac.be>
1199   if(any(y<=0)){stop("y must contain positive values")}
1200   if(any(m<=0)){stop("m must be positive")}
1201   if(any(s<=0)){stop("s must be positive")}
1202   tmp <- (f-1)*log(y)-(1/y+y/m^2)/(2*s)-f*log(m)-log(2*besselK(1/(s*m),abs(f)))
1203   if(!log)tmp <- exp(tmp)
1204   return(tmp)
1205 }
1206
1207 rginvgauss <- function(n=1, m, s, f){
1208   # Generalized inverse Gaussian distribution - Simulation
1209   #
1210   # Package:    rmutil
1211   # Version:    1.0
1212   # Title:      Utilities for Nonlinear Regression and Repeated Measurements Models
1213   # Author:     Jim Lindsey <jlindsey@luc.ac.be>
1214   tmp <- qginvgauss(runif(n),m=m,s=s,f=f)
1215   return(tmp)
1216 }
1217
1218 ghat_simulacion <- function(rho,ystar,phi,g0_licitacion_sal,N_Stil_j){
1219   #
1220   # This function simulates Monte Carlo samples from 'Ghat_j'.
1221   #
1222   # Input:
1223   #   rho                - Ux2 dimensional vector with weights associated with
1224   #   'ystar'
1225   #   ystar              - Sample ties 'y^*_i' in 'S_j'
1226   #   phi                - Probability weight associated with 'G_{j0}' (the
1227   #   continuous part of 'Ghat_j')
1228   #   g0_licitacion_sal  - Object list with three elements (produced with
1229   #   'g0_licitacion'):
1230   #       a) String object for 'distribution'
1231   #           i) Gamma
1232   #           ii) Weibull
1233   #           iii) Log-normal
1234   #           iv) Inverse-Gaussian
1235   #       b) theta        - Parameters associated with
1236   #   'distribution'
1237   #       c) mu            - Expected value of
1238   #   'distribution'
1239   #   N_Stil_j           - Number of samples to simulate (number of
1240   #   individuals in 'Stil_j') out of the sample

```

```

1235 #
1236 #   Output:
1237 #       ghat_simulacion_sal - (N_Stil_j x 1) dimensional vector with simulated data
1238 #
1239
1240 #   Repository
1241 ghat_simulacion_sal <- matrix(NaN,N_Stil_j,1)
1242 colnames(ghat_simulacion_sal) <- c("ghat_simulacion")
1243 #   Simulation
1244 sim <- 1
1245 if(N_Stil_j > 0){
1246   for(sim in 1:N_Stil_j){
1247     #   Simulated component "g_0"
1248     #   1) Discrete "ystar"
1249     #   0) Continuous "g_0"
1250     ghat_comp_sim <- rbinom(1,1,sum(rho))
1251
1252     #   Simulation "Y_jl"
1253     if( ghat_comp_sim == 1){
1254       #   Simulating the discrete component
1255       ghat_simulacion_sal[sim] <- gstar_simulacion(rho,ystar,1)
1256     }else{
1257       #   Simulating the discrete component
1258       ghat_simulacion_sal[sim] <- g0_simulacion(g0_licitacion_sal,1)
1259     }
1260   }
1261 }else{warning("The sample size is incorrect.")}
1262 #   Output
1263 salida <- ghat_simulacion_sal
1264 return(salida)
1265 #   -- END of ghat_simulacion.R --
1266 }
1267
1268 gstar_simulacion <- function(rho,ystar,nSim){
1269 #
1270 #   This function generates Monte Carlo samples from the discrete component of
1271 #   \eqn{\hat{G}_j} for the Dirichlet process prior in the planned domain \eqn{j}.
1272 #
1273 #   Input:
1274 #       rho - Ux2 dimensional vector with weights associated with
1275 #       'ystar'
1276 #       ystar - Sample ties 'y^*_i' in 'S_j'
1277 #       phi - Probability weight associated with 'G_{j0}' (the
1278 #       continuous part of 'Ghat_j')
1279 #
1280 #   Output:
1281 #       gstar_simulacion_sal - (nSim x 1) dimensional vector with simulated data
1282 #
1283 #   Normalising the weights "rho"
1284 rho_norm <- rho / sum(rho)
1285 #-----
1286 #   Verification:

```

```

1286  unicstar <- cbind(ystar, rho_norm)
1287
1288  #-----
1289  # Generating the final sample discrete component (with replacement)
1290  gstar_simulacion_sal <- sample(unicstar[, "ystar"], nSim, replace = TRUE, prob = unicstar
1291  [, "rho"])
1292
1293  # output
1294  salida <- gstar_simulacion_sal
1295  return(salida)
1296
1297  # -- END of gstar_simulacion.R --
1298  }
1299
1300  #
1301  # Inverse Gaussian distribution
1302  #
1303  # Package:    rmutil
1304  # Version:    1.0
1305  # Title:      Utilities for Nonlinear Regression and Repeated Measurements Models
1306  # Author:     Jim Lindsey <jlindsey@luc.ac.be>
1307  #
1308
1309  pinvgauss <- function(q, m, s){
1310  # Inverse Gaussian distribution - Probability
1311  #
1312  # Package:    rmutil
1313  # Version:    1.0
1314  # Title:      Utilities for Nonlinear Regression and Repeated Measurements Models
1315  # Author:     Jim Lindsey <jlindsey@luc.ac.be>
1316  if(any(q<=0)){stop("q must contain positive values")}
1317  if(any(m<=0)){stop("m must be positive")}
1318  if(any(s<=0)){stop("s must be positive")}
1319  t <- q/m
1320  v <- sqrt(q*s)
1321  out <- pnorm((t-1)/v)+exp(2/(m*s))*pnorm(-(t+1)/v)
1322  return(out)
1323  }
1324
1325  dinvgauss <- function(y, m, s, log=FALSE){
1326  # Inverse Gaussian distribution - Density
1327  #
1328  # Package:    rmutil
1329  # Version:    1.0
1330  # Title:      Utilities for Nonlinear Regression and Repeated Measurements Models
1331  # Author:     Jim Lindsey <jlindsey@luc.ac.be>
1332  if(any(y<=0)){stop("y must contain positive values")}
1333  if(any(m<=0)){stop("m must be positive")}
1334  if(any(s<=0)){stop("s must be positive")}
1335  tmp <- -(y-m)^2/(2*y*s*m^2)-(log(2*pi*s)+3*log(y))/2
1336  if(!log){tmp <- exp(tmp)}
1337  return(tmp)
1338  }

```

```

1339
1340   rinvgauss <- function(n=1, m, s){
1341     #   Inverse Gaussian distribution - Simulation
1342     #
1343     #   Package:      rmutil
1344     #   Version:      1.0
1345     #   Title:        Utilities for Nonlinear Regression and Repeated Measurements Models
1346     #   Author:       Jim Lindsey <jlindsey@luc.ac.be>
1347     temp <- qinvgauss(runif(n),m=m,s=s)
1348     return(temp)
1349   }
1350
1351   pesosDP <- function(unicos_sal,alphaDP){
1352     #
1353     #   This function computes the weights associated with the predictive distribution
1354     #   \eqn{\hat{G}_j} for a given planned domain \eqn{j},
1355     #   using the sample ties \eqn{(y^{*}_j)}, under the Dirichlet processes prior.
1356     #
1357     #   Input:
1358     #       unicos_sal - (U x 2) matrix with sample ties and associated frequencies
1359     #                   (produced with 'unicos')
1360     #       alphaDP    - Positive scalar for the Dirichlet process
1361     #
1362     #   Details:
1363     #       The matrix with sample ties 'unicos_sal' must include a column named 'm_k' for
1364     #       the frequencies of the sample ties.
1365     #
1366     #   Output:
1367     #       rho        - (U x 2) dimensional vector with weights associated with 'ystar'
1368     #                   (the sample ties)
1369     #       phi        - Probability weight associated with 'G_{j0}' (the continuous part of
1370     #                   'Ghat_j')
1371     #
1372     # computing "rho" and "phi"
1373     unicstar <- unicos_sal[[2]]
1374     U <- unicos_sal[[1]]
1375     rho <- NaN * unicstar[, "m"]
1376     M <- sum(unicstar[, "m"])
1377     for(l in 1:U){
1378       rho[l] <- unicstar[l, "m"]/(M+alphaDP)
1379     }
1380     phi <- as.matrix(alphaDP/(M+alphaDP))
1381     rho <- as.matrix(rho)
1382     colnames(rho) <- c("rho")
1383     colnames(phi) <- c("phi")
1384
1385     #   Output
1386     salida <- list(rho,phi)
1387     return(salida)
1388   }
1389
1390   #
1391   #   -- END of pesosDP.R --
1392   }

```

```

1388
1389 randDirichlet <- function(alpha,n){
1390   #
1391   #   This function simulates samples from the Dirichlet distribution with (px1) vector
1392   #   parameter \eqn{\alpha}.
1393   #
1394   #   Input:
1395   #       alpha    -   p-dimensional vector with positive entries
1396   #       n        -   Number of simulated replicates
1397   #
1398   #   Output:
1399   #       randDir  -   (Nx p) matrix with 'n' simulated replicates
1400   #
1401   #   References:
1402   #       -   "Non-Uniform Random Variate Generation", Berlin: Springer-Verlag. Devrog, Luz
1403   #           (1986)
1404   #
1405   #   Dimension "p"
1406   p <- length(alpha)
1407   #
1408   #   Repository
1409   randDir <- matrix(NaN,n,p)
1410   pAux <- NaN * rep(1:p)
1411   #
1412   #   Simulation
1413   for(i in 1:n)
1414   {
1415     for(k in 1:p)
1416     {
1417       pAux[k] <- rgamma(1, shape = alpha[k], scale = 1)
1418     }
1419     randDir[i,] <- pAux / sum(pAux)
1420   }
1421   #
1422   #   Output
1423   salida <- randDir
1424   return(salida)
1425   #
1426   #   -- END of randDirichlet.R --
1427 }
1428
1429 unicstar <- function(datos_j){
1430   #
1431   #   This function identifies the sample ties \eqn{(y^{*}_k)} in \eqn{\mathcal{S}_j} and
1432   #   computes their associated frequencies.
1433   #
1434   #   Input:
1435   #       datos_j    -   Data matrix with features and number of individuals in the
1436   #                       sample 'S_j'
1437   #
1438   #   Output. The function 'unicstar' produces an object list with two elements:
1439   #       U           -   Number of sample ties in 'S_j'

```

```
1438 #      unicstar      -      Data matrix with sample ties 'y_i' and associated frequencies
1439 #
1440
1441 y_i <- as.data.frame(datos_j)
1442 y_i <- as.data.frame(datos_j[, "y_i"])
1443 colnames(y_i) <- c("y_i")
1444
1445 #      Identifying sample ties
1446 ystar <- sort(as.matrix(unique(y_i)))
1447 unicstar <- cbind(ystar, ystar)
1448 colnames(unicstar) <- c("ystar", "m")
1449 dimUnicos <- dim(unicstar)
1450
1451 #      Computing frequencies
1452 k <- 1
1453 for(k in 1:dimUnicos[1]){
1454     ids <- which(datos_j[, "y_i"] == unicstar[k, "ystar"])
1455     unicstar[k, "m"] <- sum(datos_j[ids, "n_i"])
1456 }
1457 #      Counting the number sample ties
1458 U <- dimUnicos[1]
1459
1460 #      Output
1461 salida <- list(U, unicstar)
1462 return(salida)
1463
1464 #
1465 #      --      END of unicstar.R      --
1466 }
1467
```