SAMPLE DELIVERABLE 13

SOFTWARE BETA ON SMALL AREA ESTIMATION

Grant agreement No:	SSH - CT - 2007 - 217565
Project Acronym:	SAMPLE
Project Full title:	Small Area Methods for Poverty and Living Conditions Estimates
Funding Scheme:	Collaborative Project - Small or medium scale focused research project
Deliverable n.:	D 13
Deliverable name:	Software beta on small area estimation
WP n.:	2.2, 2.3, 2.4
Lead Beneficiary:	5
Nature:	Software
Dissemination level:	PU
Status:	final
Due delivery date:	March 31^{th}, 2010
Actual delivery date:	April 13^{th}, 2010
Project co-ordinator name:	Monica Pratesi
Title:	Associate Professor of Statistics - University of Pisa
Organization:	Department of Statistics and Mathematics Applied to Economics of the University of Pisa (UNIPI-DSMAE)
Tel.:	+39-050-2216252, +39-050-2216492
Fax:	+39-050-2216375
E-mail:	coordinator@sample-project.eu
Project website address:	www.sample-project.eu

SOFTWARE BETA ON SMALL AREA ESTIMATION

Project Acronym:	SAMPLE
Project Full title:	Small Area Methods for Poverty and Living Conditions Estimates
Grant agreement No:	SSH - CT - 2007 - 217565
WP n.:	2.2, 2.3, 2.4
Deliverable n.:	D 13
Document title:	Software beta on small area estimation
Status:	Final
Editors:	I. Molina, D. Morales, M. Pratesi and N. Tzavidis
Authors:	Nicola Salvati (salvati@ec.unipi.it), DSMAE
	Caterina Giusti (caterina.giusti@ec.unipi.it), DSMAE
	Stefano Marchetti (s.marchetti@ds.unifi.it), DSMAE
	Monica Pratesi (m.pratesi@ec.unipi.it), DSMAE
	Nikos Tzavidis (nikos.tzavidis@manchester.ac.uk), CCSR
	Isabel Molina Peralta (isabel.molina@uc3m.es), UC3M
	Domingo Morales (d.morales@umh.es), UMH
	María Dolores Esteban (md.esteban@umh.es), UMH
	Laureano Santamaría (l.santamaria@umh.es), UMH
	Yolanda Marhuenda (y.marhuenda@umh.es), UMH
	Agustín Pérez (agustin.perez@umh.es), UMH
	Maria Chiara Pagliarella (mc.pagliarella@unicas.it), UMH
	Ray Chambers (ray@uow.edu.au), CSSM
	J.N.K. Rao (jrao@math.carleton.ca), Carleton University
	Gauri Datta (gauri@stat.uga.edu), University of Georgia

Contents

Prologue

This report contains some preliminary (beta) versions of the software developments on small area estimation carried out by the partners of WP2 in the SAMPLE project. The target of the report is to give a guide for users of the developed software. For the sake of completeness, we have also paid special attention to introduce the programmed statistical methodology and to illustrate how to use the software in applications to data.

Although the presented software has already been used successfully, two remarks should be taken into account when reading this document. First one is that the software testing and the addition of further improvements are still being done. The final version will be presented in deliverable D22. Second one is that the software appearing in this report is a subset of all the software being developed in the SAMPLE project. The complete software will be presented in deliverable D22.

The manuscript is organized in 13 chapters. Chapters 1-6 contains the statistical methodology, the the software description of the R function and the examples of usage. Chapters 7-13 contains the codes of the R functions

Chapter 1 introduces the basic Fay-Herriot (FH) model and describes the corresponding software. As some of the models proposed by the SAMPLE project are in fact generalization of the FH model, this chapter will be of great interest for users. Chapter 2 introduces an area-level spatial model and describe two fitting methods and two bootstrap procedures to estimate the mean squared error (MSE) of the empirical best linear unbiased estimator. An example data set is given to illustrate how to use the programmed R functions. The software in chapters 7-8 has been programmed by the UC3M team.

Chapter 3 deals with area-level time models. Two models are presented. The second one contains time random effects following an auto-regressive process AR(1) and the first one is a simplification where these effects are independent. Theoretical descriptions are presented as well as an example of use. The software in chapters 9-10 has been programmed by the UMH team.

Chapter 4 introduces a methodology for obtaining small area estimation of a domain mean by using the M-quantile regression approach. Chapter 5 do the same but using the M-quantile geographically weighted regression approach where the regression parameters may vary between specified locations. These chapters also discuss the estimation of the domain means and the estimation of the corresponding MSEs. Both chapters present software descriptions and examples of usage. Chapter 6 introduces an estimators of the small area cumulative distribution function (cdf) by using the Chambers-Dunstan estimator of the cdf and the linear M-quantile small area model. The software in chapters 11-13 has been programmed by the UNIPI-DSMAE and the CCSR teams.

Chapter 1

Fay-Herriot model

1.1 Methodology

1.1.1 Model and small area EB estimator

Fay-Herriot (FH) models were introduced by Fay & Herriot (1979) to obtain small area estimators of median income in small places in the U.S. These models are well known in the literature of small area estimation (SAE) and are the basic tool when auxiliary data at the unit level are not available or there are confidentiality reasons preventing their use, and therefore only aggregated data at the small area level are available.

The basic Fay-Herriot model, assuming normality, is defined as

$$\begin{aligned} (i) \qquad & Y_i|\theta_i \overset{ind}{\sim} N(\theta_i, D_i),\ i=1,\ldots,m; \\ (ii) \qquad & \theta_i \overset{ind}{\sim} N(\mathbf{x}_i'\beta, A),\ i=1,\ldots,m. \end{aligned} \tag{1.1}$$

Component (ii) of (1.1) is the linking model and component (i) is the sampling model. Marginally,

$$Y_i \overset{ind}{\sim} N(\mathbf{x}_i'\beta, D_i + A),\ i=1,\ldots,m. \tag{1.2}$$

In matrix notation, (1.2) may be written as $\mathbf{Y} \sim N\{\mathbf{X}\beta, \Sigma(A)\}$, where $\mathbf{Y} = (Y_1,\ldots,Y_m)'$, $\mathbf{X} = (\mathbf{x}_1,\ldots,\mathbf{x}_m)'$ and $\Sigma(A) = diag(A+D_1,\ldots,A+D_m)$.

The best (or Bayes) estimator of θ_i under squared error loss is given by

$$\begin{aligned} \hat{\theta}_i^B = E(\theta_i|Y_i) &= Y_i - \frac{D_i}{A+D_i}(Y_i - \mathbf{x}_i'\beta) \\ &= \{1 - B_i(A)\}Y_i + B_i(A)\mathbf{x}_i'\beta, \end{aligned} \tag{1.3}$$

noting that $\theta_i|Y_i \overset{ind}{\sim} N\{\hat{\theta}_i^B, g_{1i}(A)\}$, where

$$g_{1i}(A) = D_i\{1 - B_i(A)\} \tag{1.4}$$

and $B_i(A) = D_i/(A+D_i)$. Expression (1.3) shows that $\hat{\theta}_i^B$ is a convex combination of the direct estimator Y_i and the regression synthetic estimator $\mathbf{x}_i'\beta$.

In practice, the model parameters β and A are unknown. For a given A, the maximum likelihood (ML) estimator of β is obtained from (1.2) as

$$\begin{aligned}\tilde{\beta}(A) &= \{\mathbf{X}'\Sigma^{-1}(A)\mathbf{X}\}^{-1}\mathbf{X}'\Sigma^{-1}(A)\mathbf{Y} \\ &= \left\{\sum_{j=1}^{m}(A+D_j)^{-1}\mathbf{x}_j\mathbf{x}_j'\right\}^{-1}\sum_{j=1}^{m}(A+D_j)^{-1}\mathbf{x}_jY_j. \qquad (1.5)\end{aligned}$$

Note that $\tilde{\beta}(A)$ is also the weighted least squares (WLS) estimator of β without normality assumption. Substituting $\tilde{\beta}(A)$ for β in (1.3), we get the first-step empirical best (or empirical Bayes) estimator, $\tilde{\theta}_i^{EB}(A)$, of θ_i which is also equal to the best linear unbiased prediction (BLUP) estimator of θ_i without normality assumption.

Several estimators, $\hat{A}$, of A have been proposed in the literature including moment estimators without normality assumption, ML and restricted (or residual) ML estimator (REML) estimator; see Section 1.1.2. Substituting $\hat{A}$ for A in the first-step empirical best (EB) estimator, $\tilde{\theta}_i^{EB}(A)$, we get the final EB estimator $\hat{\theta}_i^{EB}$:

$$\begin{aligned}\hat{\theta}_i^{EB} = \tilde{\theta}_i^{EB}(\hat{A}) &= \{1-B_i(\hat{A})\}Y_i + B_i(\hat{A})\mathbf{x}_i'\hat{\beta} \\ &= Y_i - B_i(\hat{A})(Y_i - \mathbf{x}_i'\hat{\beta}), \qquad (1.6)\end{aligned}$$

where $\hat{\beta} = \tilde{\beta}(\hat{A})$. Without the normality assumption, the estimator (1.6) is also the empirical BLUP (EBLUP) estimator.

1.1.2 Fitting methods

In this section, we give the formulae for an estimator of A based on moment method, ML and REML methods. A moment estimator, due to Fay and Herriot (1979), is given by $\hat{A}_{FH} = \max(0, A^*_{FH})$ with A^*_{FH} obtained iteratively as the solution of the following non-linear equation in A:

$$\sum_{j=1}^{m}(A+D_j)^{-1}\{Y_j - \mathbf{x}_j'\tilde{\beta}(A)\}^2 = m-p. \qquad (1.7)$$

This equation can be solved using an iterative method such as the Fisher-scoring algorithm. For this, let us define

$$s(A) = \sum_{j=1}^{m}(A+D_j)^{-1}\{Y_j - \mathbf{x}_j'\tilde{\beta}(A)\}^2 - m - p.$$

By a first order Taylor expansion of $s(\hat{A}_{FH})$ around the true A, we get

$$0 = s(\hat{A}_{FH}) \approx s(A) + s'(A)(\hat{A}_{FH} - A). \qquad (1.8)$$

The Fisher-scoring algorithm replaces in this equation, the derivative $s'(A)$ by its expectation $E[-s'(A)]$, which in this case is equal to

$$E[-s'(A)] = \sum_{j=1}^{m}(A+D_j)^{-1}.$$

Then, solving for $\hat{A}_{FH}$ in (1.8), we get

$$\hat{A}_{FH} = A + \left\{E[-s'(A)]\right\}^{-1} s(A).$$

This algorithm starts with an initial value $\hat{A}_{FH}^{(0)}$, and then in each iteration it updates the estimate using the updating equation

$$\hat{A}_{FH}^{(k+1)} = \hat{A}_{FH}^{(k)} + \left\{ E[-s'(A)]\big|_{A=\hat{A}_{FH}^{(k)}} \right\}^{-1} s(\hat{A}_{FH}^{(k)}),$$

In the function `fitFH`, the starting value is set to $\hat{A}_{FH}^{(0)} = \text{median}(D_i)$. It stops either when the number of iterations $k >$ MAXITER where MAXITER can be chosen by the user (default is 500), or when

$$\left| \frac{\hat{A}_{FH}^{(k+1)} - \hat{A}_{FH}^{(k)}}{\hat{A}_{FH}^{(k)}} \right| < 0.0001.$$

Convergence of the iteration is generally rapid.

Assuming normality, A and β can be estimated by ML or REML procedures. In fact, under regularity conditions, the estimators derived from these two methods (and using the Normal likelihood) remain consistent at order $O_p(m^{-1/2})$ even without the Normality assumption, for details see Jiang (1996). ML estimators of A and β are obtained by maximizing the log-likelihood, given by

$$\ell(A, \beta; \mathbf{Y}) = c - \frac{1}{2}\log|\Sigma(A)| - \frac{1}{2}(\mathbf{Y} - \mathbf{X}\beta)'\Sigma^{-1}(A)(\mathbf{Y} - \mathbf{X}\beta), \tag{1.9}$$

where c denotes a constant. Taking derivative of ℓ with respect to β and equating to zero, we obtain the maximum likelihood equation for β, which gives the WLS estimator (1.5). The maximum likelihood equation for A is obtained taking derivative of ℓ with respect to A and equating to zero, and is given by

$$\sum_{j=1}^{m} (A + D_j)^{-2} \{Y_j - \mathbf{x}_j'\tilde{\beta}(A)\}^2 = \sum_{j=1}^{m} (A + D_j)^{-1}. \tag{1.10}$$

Again, Fisher-scoring algorithm may be used to solve this equation. Let us denote

$$s_{ML}(A) = \sum_{j=1}^{m} (A + D_j)^{-2} \{Y_j - \mathbf{x}_j'\tilde{\beta}(A)\}^2 - \sum_{j=1}^{m} (A + D_j)^{-1}.$$

The Fisher information for A is obtained by taking expectation of the negative derivative of $s_{ML}(A)$, and is given by

$$I_{ML}(A) = E\left\{-s'_{ML}(A)\right\} = \frac{1}{2}\sum_{j=1}^{m} (A + D_j)^{-2}.$$

Finally, the updating equation for the ML estimator of A is

$$\hat{A}_{ML}^{(k+1)} = \hat{A}_{ML}^{(k)} + \left\{ I_{ML}(\hat{A}_{ML}^{(k)}) \right\}^{-1} s_{ML}(\hat{A}_{ML}^{(k)}).$$

Initial value of A and stopping criterion are set the same as in the FH method described before. Let $\hat{A}^*_{ML}$ be the estimate obtained in the last iteration of the algorithm. Then, the final ML estimate is $\hat{A}_{ML} = \max(0, \hat{A}^*_{ML})$.

The REML estimator of A is obtained by maximizing the joint p.d.f. of a transformation $\mathbf{F}'\mathbf{Y}$ of the data $\mathbf{Y}$, where $\mathbf{F}$ is an $m \times p$ matrix satisfying $\mathbf{F}'\mathbf{X} = \mathbf{0}$. Then, the REML estimator maximizes the following function that does not depend on β,

$$\ell_R(A;\mathbf{Y}) = c - \frac{1}{2}\log|\mathbf{F}'\Sigma(A)\mathbf{F}| - \frac{1}{2}\mathbf{Y}'\mathbf{F}\left(\mathbf{F}'\Sigma(A)\mathbf{F}\right)^{-1}\mathbf{F}'\mathbf{Y}.$$

It holds that

$$\mathbf{F}\left(\mathbf{F}'\Sigma(A)\mathbf{F}\right)^{-1}\mathbf{F}' = \mathbf{P}(A),$$

where

$$\mathbf{P}(A) = \Sigma^{-1}(A) - \Sigma^{-1}(A)\mathbf{X}\left(\mathbf{X}'\Sigma^{-1}(A)\mathbf{X}\right)^{-1}\mathbf{X}'\Sigma^{-1}(A).$$

Using this relation, we obtain

$$\ell_R(A;\mathbf{Y}) = c - \frac{1}{2}\log|\mathbf{F}'\Sigma(A)\mathbf{F}| - \frac{1}{2}\mathbf{Y}'\mathbf{P}(A)\mathbf{Y}.$$

The score is obtained by taking derivative of ℓ_R with respect to A, and is given by

$$\begin{aligned}
s_R(A) &= -\frac{1}{2}\text{trace}\{\mathbf{P}(A)\} + \frac{1}{2}\mathbf{Y}'\mathbf{P}^2(A)\mathbf{Y} \\
&= -\sum_{j=1}^{m}(A+D_j)^{-1} - \text{trace}\{\left(\mathbf{X}'\Sigma^{-1}(A)\mathbf{X}\right)^{-1}\mathbf{X}'\Sigma^{-2}(A)\mathbf{X}\} \\
&\quad + \frac{1}{2}\left\{\mathbf{Y} - \mathbf{X}\tilde{\beta}(A)\right\}'\Sigma^{-2}(A)\left\{\mathbf{Y} - \mathbf{X}\tilde{\beta}(A)\right\} \\
&= \sum_{j=1}^{m}(A+D_j)^{-1} - \sum_{j=1}^{m}\frac{\mathbf{x}_j'\left\{\sum_{k=1}^{m}(A+D_k)^{-1}\mathbf{x}_k\mathbf{x}_k'\right\}^{-1}\mathbf{x}_j}{(A+D_j)^2} - \sum_{j=1}^{m}\frac{\{Y_j - \mathbf{x}_j'\tilde{\beta}(A)\}^2}{(A+D_j)^2}.
\end{aligned}$$

The REML estimator of A is obtained by solving the non-linear equation $s_R(A) = 0$. Again, application of Fisher-scoring algorithm requires also the Fisher information for A, which is given by

$$\begin{aligned}
I_R(A) &= E\left\{-s_R'(A)\right\} = \frac{1}{2}\text{trace}\{\mathbf{P}^2(A)\} \\
&= \frac{1}{2}\text{trace}\{\Sigma(A)^{-2}\} - \text{trace}\left[\{\mathbf{X}'\Sigma^{-1}(A)\mathbf{X}\}^{-1}\mathbf{X}'\Sigma^{-3}(A)\mathbf{X}\right] \\
&\quad + \frac{1}{2}\text{trace}\left\{\left(\left[\{\mathbf{X}'\Sigma^{-1}(A)\mathbf{X}\}^{-1}\mathbf{X}'\Sigma^{-3}(A)\mathbf{X}\right]\right)^2\right\}.
\end{aligned}$$

Finally, the updating equation is

$$\hat{A}^{(k+1)}_{REML} = \hat{A}^{(k)}_{REML} + \left\{I_R(\hat{A}^{(k)}_{REML})\right\}^{-1} s_R(\hat{A}^{(k)}_{REML}).$$

Initial value $\hat{A}^{(0)}$ and stopping criterion are set the same as in the FH and ML methods. Again, if $\hat{A}^*_{REML}$ is the last value obtained in the iteration, then REML estimate is finally $\hat{A}_{REML} = \max(0, \hat{A}^*_{REML})$.

The estimator of β is obtained by replacing in (1.5), A by an estimator $\hat{A}$, that is, $\hat{\beta} = \tilde{\beta}(\hat{A})$. Similarly, the EB estimator is obtained as in (1.6). Function `fitFH` delivers, together with estimates of model coefficients β, their asymptotic standard errors given by the diagonal elements of the Fisher information in each case, the Z statistics obtained by dividing the estimates by their standard errors, and the p-values of the significance tests. Since for large m, it holds

$$\hat{\beta} \sim N(\beta, I^{-1}(\beta)),$$

where $I(\beta)$ is the Fisher information, then the Z statistic for a coefficient β_j is

$$Z_j = \hat{\beta}_j / \sqrt{\hat{var}(\hat{\beta}_j)}, \quad j = 1, \ldots, p,$$

where $\hat{var}(\hat{\beta}_j)$ is the estimated asymptotic variance of $\hat{\beta}_j$, given by the j-th element in the diagonal of $I^{-1}(\hat{\beta})$. Finally, for the test

$$H_0 : \beta_j = 0 \quad \text{versus} \quad H_1 : \beta_j \neq 0,$$

p-values are obtained as

$$\text{p-value} = 2P(Z > |Z_j|),$$

where Z is a standard normal random variable.

Three different goodness of fit measures are also delivered by function `fitFH`. The first one is the estimated log-likelihood $\ell(\hat{A}, \hat{\beta}; \mathbf{Y})$, obtained by replacing the obtained estimates $\hat{A}$ and $\hat{\beta}$ in (1.9). The second is AIC, given in this case by

$$\text{AIC} = -2\ell(\hat{A}, \hat{\beta}; \mathbf{Y}) + 2(p+1).$$

Finally, the BIC is obtained as

$$\text{BIC} = -2\ell(\hat{A}, \hat{\beta}; \mathbf{Y}) + (p+1)\log(m).$$

1.1.3 Mean squared error of the EB estimator

In practical applications, the EB estimator $\hat{\theta}_i^{EB}$ should be accompanied with its estimated MSE. Under model (1.1), the MSE of the best estimator $\hat{\theta}_i^B$ is given by

$$MSE(\hat{\theta}_i^B) = E(\hat{\theta}_i^B - \theta_i)^2 = E\{V(\theta_i|Y_i)\} = E\{g_{1i}(A)\} = g_{1i}(A),$$

showing that a large reduction in MSE over $MSE(Y_i) = E[E\{(Y_i - \theta_i)^2|\theta_i\}] = E(D_i) = D_i$ is obtained when $1 - B_i(A) = A/(A + D_i)$ is small. Under normality of random effects and errors, the MSE of the EB can be decomposed as

$$\begin{aligned} \text{MSE}(\hat{\theta}_i^{EB}) &= \text{MSE}[\tilde{\theta}_i^{EB}(A)] \quad + E\{[\tilde{\theta}_i^{EB}(\hat{A}) - \tilde{\theta}_i^{EB}(A)]^2\} \\ &= [g_{1i}(A) + g_{2i}(A)] + g_{3i}(A), \end{aligned} \tag{1.11}$$

where $g_{1i}(A)$ is $O(1)$ for large m, $g_{2i}(A)$ is due to the estimation of β and is $O(m^{-1})$, and the last term measures the uncertainty of the EB estimator arising from the estimation of A and is of lower order (Prasad & Rao, 1990). The last two terms on the right hand side of (1.11) are given by

$$\begin{aligned} g_{2i}(A) &= \{B_i(A)\}^2 \mathbf{x}_i' \left\{ \sum_{j=1}^{m} (A+D_j)^{-1} \mathbf{x}_j \mathbf{x}_j' \right\}^{-1} \mathbf{x}_i \\ &=: \{B_i(A)\}^2 \tilde{h}_{ii} \end{aligned} \tag{1.12}$$

and

$$g_{3i}(A) = \{B_i(A)\}^2 (A+D_i)^{-1} \bar{V}(\hat{A}), \tag{1.13}$$

where $\bar{V}(\hat{A})$ is the asymptotic variance (as $m \to \infty$) of an estimator $\hat{A}$ of A. Note that $g_{3i}(A)$ depends on the choice of $\hat{A}$.

Using the REML estimator $\hat{A}_{REML}$, a nearly unbiased estimator of $MSE(\hat{\theta}_i^{EB})$ is given by

$$mse_{REML}(\hat{\theta}_i^{EB}) = g_{1i}(\hat{A}_{REML}) + g_{2i}(\hat{A}_{REML}) + 2g_{3i}(\hat{A}_{REML}), \tag{1.14}$$

where $g_{1i}(A)$ is given by (1.5) and

$$\bar{V}(\hat{A}_{REML}) = \frac{2}{\sum_{j=1}^{m} (A+D_j)^{-2}}; \tag{1.15}$$

see Datta and Lahiri (2000).

For the Fay–Herriot (FH) estimator $\hat{A}_{FH}$, we need the following expression for its bias to terms of order $O(m^{-1})$:

$$b_{FH}(A) = \frac{2\left[m\sum_{j=1}^{m}(A+D_j)^{-2} - \left\{\sum_{j=1}^{m}(A+D_j)^{-1}\right\}^2\right]}{\left\{\sum_{j=1}^{m}(A+D_j)^{-1}\right\}^3}. \tag{1.16}$$

Note that the bias of $\hat{A}_{REML}$ is zero if terms of order $o(m^{-1})$ are ignored.

A nearly unbiased estimator of $MSE(\hat{\theta}_i^{EB})$ using $\hat{A}_{FH}$ is given by

$$mse_{FH}(\hat{\theta}_i^{EB}) = g_{1i}(\hat{A}_{FH}) + g_{2i}(\hat{A}_{FH}) + 2g_{3i}(\hat{A}_{FH}) - b_{FH}(\hat{A}_{FH})\{B_i(\hat{A}_{FH})\}^2, \tag{1.17}$$

where, in $g_{3i}(\hat{A}_{FH})$, the asymptotic variance is

$$\bar{V}(\hat{A}_{FH}) = \frac{2m}{\left\{\sum_{j=1}^{m}(A+D_j)^{-1}\right\}^2}; \tag{1.18}$$

see Datta et al. (2005).

1.2 The Software: description of R functions

This section describes the implemented R functions that fit the basic Fay-Herriot model (1.1), give the small area EB estimates and analytical estimates of the MSE of the EB estimator. In the rest of this section we describe briefly these R functions. An example showing the use of these functions is provided in Section 1.3 and full R codes are included in Appendix 1.

1.2.1 fitFH

R function `fitFH` fits the basic Fay-Herriot model (1.1). This function is defined as

```
fitFH<-function(X,y,Dvec,method="REML",MAXITER=500)
```

Arguments of this function are:

`X`**:** matrix containing the aggregated (population) values of p auxiliary variables, with dimension $m \times p$, where m is the number of areas or sample size. The elements in first column might be equal to 1 if the model includes an intercept.

`y`**:** vector containing the direct estimates of the response variable for the m areas.

`Dvec`**:** vector containing the m sampling variances $D_1, \ldots, D_m$ of direct estimators.

`method`**:** type of fitting method, to be chosen between REML or FH methods. Default is REML method.

`MAXITER`**:** maximum number of iterations allowed in the Fisher-scoring algorithm. Default is 500 iterations.

The function returns a list with the following objects:

`convergence`**:** a logical value equal to TRUE if Fisher-scoring algorithm converges in less than MAXITER iterations.

`modelcoefficients`**:** data.frame in the shape of a table with estimated model coefficients in first column, their (asymptotic) standard errors in second column, Z statistics in third column and p-values of the significance of each coefficient in last column.

`variance`**:** estimated random effects variance A.

`goodnessoffit`**:** a vector containing three basic goodness-of-fit measures, loglikelihood, AIC and BIC.

`EBpredictor`**:** a vector of size m with the values of the EB predictor for the m areas.

1.2.2 MSE.FHmodel

R function `MSE.FHmodel` gives analytical MSE estimates of EB predictors for the m small areas, when EB predictors are obtained from the basic Fay-Herriot model (1.1). This function is defined as

```
MSE.FHmodel<-function(X,Dvec,A,method="REML")
```

Arguments of this function are:

`X`**:** matrix containing the aggregated (population) values of p auxiliary variables, with dimension $m \times p$, where m is the number of areas or sample size. The elements in first column might be equal to 1 if the model includes an intercept.

`Dvec`**:** vector containing the m sampling variances $D_1, \ldots, D_m$ of direct estimators.

`A`**:** estimated random effects variance A obtained using the fitting method specified in `method`:.

`method`**:** Type of fitting method, to be chosen between REML or FH methods. Default is REML method.

The function returns:

`mse`**:** a vector with the estimated MSEs of the EB small area estimators.

1.3 Examples of usage of R functions

This section shows how to use the R functions described in Section 1.2 to produce small area EB estimators along with their corresponding estimated MSEs, based on the basic Fay-Herriot model (1.1).

1.3.1 Example data set

We consider the data set on milk expenditure used initially by Arora and Lahiri (1997) and later by You and Chapman (2006). This data set comes from the Consumer Expenditure Survey conducted by the U.S. Bureau of the Census and is used by the Bureau of Labor Statistics to compute the monthly Consumer Price Index (CPI) numbers. Here the small areas are the $m = 43$ publication areas of the CPI throughout the U.S. and Y_i is the i-the area direct estimator of the average expenditure on fresh whole milk for the year 1989. As explanatory variables in the Fay-Herriot model, we used indicators for the 4 major areas created by You and Chapman (2006). Table 1.1 lists the full data including Small area, sample size n_i, direct estimate y_i, standard error of direct estimate y_i (SD), coefficient of variation of y_i (CV) and Major area.

Small area	n_i	y_i	SD	CV	Major area
1	191	1.099	0.163	0.148	1
2	633	1.075	0.08	0.074	1
3	597	1.105	0.083	0.075	1
4	221	0.628	0.109	0.174	1
5	195	0.753	0.119	0.158	1
6	191	0.981	0.141	0.144	1
7	183	1.257	0.202	0.161	1
8	188	1.095	0.127	0.116	2
9	204	1.405	0.168	0.12	2
10	188	1.356	0.178	0.131	2
11	149	0.615	0.1	0.163	2

12	290	1.46	0.201	0.138	2
13	250	1.338	0.148	0.111	2
14	194	0.854	0.143	0.167	2
15	184	1.176	0.149	0.127	3
16	193	1.111	0.145	0.131	3
17	218	1.257	0.135	0.107	3
18	266	1.43	0.172	0.12	3
19	214	1.278	0.137	0.107	3
20	213	1.292	0.163	0.126	3
21	196	1.002	0.125	0.125	3
22	95	1.183	0.247	0.209	3
23	195	1.044	0.14	0.134	3
24	187	1.267	0.171	0.135	3
25	479	1.193	0.106	0.089	3
26	230	0.791	0.121	0.153	4
27	186	0.795	0.121	0.152	4
28	199	0.759	0.259	0.341	4
29	238	0.796	0.106	0.133	4
30	207	0.565	0.089	0.158	4
31	165	0.886	0.225	0.254	4
32	153	0.952	0.205	0.215	4
33	210	0.807	0.119	0.147	4
34	383	0.582	0.067	0.115	4
35	255	0.684	0.106	0.155	4
36	226	0.787	0.126	0.16	4
37	224	0.44	0.092	0.209	4
38	212	0.759	0.132	0.174	4
39	211	0.77	0.1	0.13	4
40	179	0.8	0.113	0.141	4
41	312	0.756	0.083	0.11	4
42	241	0.865	0.121	0.14	4
43	205	0.64	0.129	0.202	4

Table 1.1: Data on Milk expenditure.

1.3.2 Example of R code for running function fitFH

Here we include an example of R code used to read the data set in Table 1.1 and run function `fitFH` using that data. R code includes suitable comments explaining what is each line doing.

```
# Set the Path or folder where data set and functions are.
setwd("Path")

# Read data set
```

```
data<-read.table("MilkData.txt",header=TRUE)
attach(data)

# Create the auxiliary variables, which are the indicators
# of 4 Major Areas
# Create these indicators and put them in the columns of matrix X.

m<-dim(data)[1]
M<-length(unique(MajorArea))
X<-matrix(0,nr=m,nc=M)
for (i in 1:4) {X[,i]<-as.numeric(MajorArea==i)}

# Load file where function is located
source("Fitting_FHModel.R")

# Call the function using REML method and put the output
# in the list results.
results<-fitFH(X,yi,SD^2,method="REML")

# Print function output
print(results)

# Fit function using FH method.
# Include the function output in an object call results.
# Now call the function using FH method and put the output in
# list results.
results<-fitFH(X,yi,SD^2,method="FH")
print(results)
```

1.3.3 Output of function fitFH

Output of function `fitFH` when setting

`method="REML"`

is given below:

```
-----------------------------------------------------
$convergence
[1] TRUE

$modelcoefficients
```

```
  beta.REML std.errorbeta   tvalue       pvalue
1  0.968189    0.06936237 13.95842 2.795652e-44
2  1.100970    0.07614518 14.45882 2.205456e-47
3  1.195135    0.06094029 19.61158 1.231550e-85
4  0.726888    0.04301468 16.89860 4.606911e-64

$variance
[1] 0.01855048

$goodnessoffit
   loglike        AIC        BIC
 12.677463 -15.354927  -6.548926

$EBpredictor
           [,1]
 [1,] 1.0219708
 [2,] 1.0476021
 [3,] 1.0679516
 [4,] 0.7608159
 [5,] 0.8461566
 [6,] 0.9743727
 [7,] 1.0584532
 [8,] 1.0977764
 [9,] 1.2215462
[10,] 1.1951466
[11,] 0.7852141
[12,] 1.2139470
[13,] 1.2096603
[14,] 0.9834961
[15,] 1.1864247
[16,] 1.1556980
[17,] 1.2263414
[18,] 1.2856494
[19,] 1.2363250
[20,] 1.2349603
[21,] 1.0903013
[22,] 1.1923057
[23,] 1.1216465
[24,] 1.2230299
[25,] 1.1938055
```

```
[26,] 0.7627197
[27,] 0.7649553
[28,] 0.7338445
[29,] 0.7699297
[30,] 0.6134414
[31,] 0.7695564
[32,] 0.7958257
[33,] 0.7723190
[34,] 0.6102299
[35,] 0.7001781
[36,] 0.7592790
[37,] 0.5298859
[38,] 0.7434468
[39,] 0.7548997
[40,] 0.7701921
[41,] 0.7481165
[42,] 0.8040778
[43,] 0.6810868
-------------------------------------------------------
```

Output of function `fitFH` when setting

```
method="FH"
```

is given below:

```
-------------------------------------------------------
$convergence
[1] TRUE

$modelcoefficients
    beta.FH std.errorbeta   tvalue       pvalue
1 0.9679012    0.06695896 14.45514 2.326569e-47
2 1.0973513    0.07400814 14.82744 9.737434e-50
3 1.1946922    0.05925359 20.16236 2.096386e-90
4 0.7257494    0.04150620 17.48532 1.853544e-68

$variance
[1] 0.01642027

$goodnessoffit
```

```
  loglike       AIC       BIC
 12.76205 -15.52410  -6.71810

$EBpredictor
           [,1]
 [1,] 1.0179759
 [2,] 1.0449639
 [3,] 1.0644808
 [4,] 0.7706920
 [5,] 0.8525124
 [6,] 0.9738262
 [7,] 1.0508569
 [8,] 1.0961652
 [9,] 1.2105053
[10,] 1.1856404
[11,] 0.7975687
[12,] 1.2021499
[13,] 1.2004587
[14,] 0.9889713
[15,] 1.1867450
[16,] 1.1579920
[17,] 1.2242232
[18,] 1.2786804
[19,] 1.2335659
[20,] 1.2318601
[21,] 1.0959551
[22,] 1.1922126
[23,] 1.1259974
[24,] 1.2206948
[25,] 1.1936875
[26,] 0.7602435
[27,] 0.7623581
[28,] 0.7322880
[29,] 0.7674591
[30,] 0.6173102
[31,] 0.7649969
[32,] 0.7893148
[33,] 0.7693760
[34,] 0.6128615
[35,] 0.7009616
```

```
[36,] 0.7568908
[37,] 0.5371932
[38,] 0.7418816
[39,] 0.7532513
[40,] 0.7675187
[41,] 0.7470595
[42,] 0.7993630
[43,] 0.6831609
-----------------------------------------------------
```

1.3.4 Example of R code for running function MSE.FHmodel

Here we include an example of R code used to read the data set in Table 1.1, fit the Fay-Herriot model to that data set using function `fitFH` which gives also the small area EB estimators, and finally run function `MSE.FHmodel` to obtain analytical MSE estimators of EB predictors. R code includes suitable comments explaining what is each line doing.

```
# Set path where data set and functions are
setwd("Path")

# Read data set
data<-read.table("MilkData.txt",header=TRUE)
attach(data)

# The auxiliary variables are indicators of 4 Major Areas
# Create these indicators

m<-dim(data)[1]
M<-length(unique(MajorArea))
X<-matrix(0,nr=m,nc=M)
for (i in 1:4) {X[,i]<-as.numeric(MajorArea==i)}

# Load files where R functions are
source("Fitting_FHModel.R")
source("MSE_FHModel.R")

# Fit FH model using FH method
results<-fitFH(X,yi,SD^2,method="FH")

# Compute estimated MSEs of EB estimators
mse<-MSE.FHmodel(X,SD^2,results$variance,method="REML")
```

```
mse
```

1.3.5 Output of function MSE.FHmodel

Output of function `MSE.FHmodel` when setting

```
method="REML"
```

is given below:

```
------------------------------------------------------------
 [1] 0.012757016 0.005314467 0.005632201 0.008323471 0.009283520
 [6] 0.011178151 0.014867661 0.010252709 0.013470878 0.014094867
[11] 0.007558331 0.015325273 0.012038943 0.011640323 0.011466957
[16] 0.011181888 0.010423673 0.012914352 0.010580560 0.012385543
[21] 0.009599980 0.015890239 0.010810965 0.012857861 0.007864537
[26] 0.008855177 0.008855177 0.015041525 0.007569376 0.005975211
[31] 0.014211799 0.013571948 0.008691546 0.003833361 0.007569376
[36] 0.009253152 0.006264329 0.009709449 0.007020470 0.008185874
[41] 0.005391054 0.008855177 0.009484220
------------------------------------------------------------
```

Chapter 2

Area-level spatial model

2.1 Methodology

2.1.1 Model and small area EB estimator

The Fay-Herriot model defined as (1.1) can be also expressed as

$$\mathbf{y} = \mathbf{X}\beta + \mathbf{v} + \mathbf{e}, \tag{2.1}$$

where $\mathbf{y} = (y_1, \ldots, y_m)^T$ is the vector of direct estimators for the m small areas, $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_m)^T$ is a $m \times p$ matrix containing in the columns the values of p explanatory variables for the m areas, $\mathbf{v} = (v_1, \ldots, v_m)^T$ is a vector of independent and identically distributed variables (called random effects) with $\mathbf{v} \sim N(\mathbf{0}, \sigma_v^2 I_m)$ and $\mathbf{e} = (e_1, \ldots, e_m)^T$ is the vector of independent sampling errors, independent of $\mathbf{v}$, with $\mathbf{e} \sim N(\mathbf{0}, \Psi)$, where the covariance matrix $\Psi = \text{diag}(\psi_1, \ldots, \psi_m)$ is known. Here, the target quantity is the vector $\theta = \mathbf{X}\beta + \mathbf{v} = (\theta_1, \ldots, \theta_m)^T$, which usually contains the true means of the target variable in the m areas.

Model (2.1) can be extended to allow for spatially correlated area effects as follows. Let $\mathbf{v}$ be the result of a SAR process with unknown autoregression parameter ρ and proximity matrix $\mathbf{W}$ (Anselin, 1988; Cressie, 1993), i.e.,

$$\mathbf{v} = \rho\mathbf{W}\mathbf{v} + \mathbf{u}. \tag{2.2}$$

We assume that the matrix $(\mathbf{I}_m - \rho\mathbf{W})$ is non-singular. Then $\mathbf{v}$ can be expressed as

$$\mathbf{v} = (\mathbf{I}_m - \rho\mathbf{W})^{-1}\mathbf{u}. \tag{2.3}$$

Here, $\mathbf{u} = (u_1, \ldots, u_m)^T$ is a vector with mean $\mathbf{0}$ and covariance matrix $\sigma_u^2\mathbf{I}_m$, where $\mathbf{I}_m$ denotes the $m \times m$ identity matrix and σ_u^2 is an unknown parameter. We consider that the proximity matrix $\mathbf{W}$ is defined in row standardized form; that is, $\mathbf{W}$ is row stochastic. Then, $\rho \in (-1, 1)$ is called spatial autocorrelation parameter (Banerjee et al., 2004). Hereafter, the vector of variance components will be denoted $\omega = (\omega_1, \omega_2)^T = (\sigma_u^2, \rho)^T$. Equation (2.3) implies that $\mathbf{v}$ has mean vector $\mathbf{0}$ and covariance matrix equal to

$$\mathbf{G}(\omega) = \sigma_u^2[(\mathbf{I}_m - \rho\mathbf{W})^T(\mathbf{I}_m - \rho\mathbf{W})]^{-1}. \tag{2.4}$$

Since $\mathbf{e}$ is independent of $\mathbf{v}$, the covariance matrix of $\mathbf{y}$ is equal to

$$\mathbf{V}(\omega) = \mathbf{G}(\omega) + \Psi.$$

Combining (2.1) and (2.3), the model is

$$\mathbf{y} = \mathbf{X}\beta + (\mathbf{I}_m - \rho\mathbf{W})^{-1}\mathbf{u} + \mathbf{e} \tag{2.5}$$

Under model (2.5), the Spatial BLUP of the quantity of interest $\theta_i = \mathbf{x}_i^T\beta + v_i$ is

$$\tilde{\theta}_i(\omega) = \mathbf{x}_i^T\tilde{\beta}(\omega) + \mathbf{b}_i^T\mathbf{G}(\omega)\mathbf{V}^{-1}(\omega)[\mathbf{y} - \mathbf{X}\tilde{\beta}(\omega)], \tag{2.6}$$

where $\tilde{\beta}(\omega) = [\mathbf{X}^T\mathbf{V}^{-1}(\omega)\mathbf{X}]^{-1}\mathbf{X}^T\mathbf{V}^{-1}(\omega)\mathbf{y}$ is the generalized least squares estimator of the regression parameter β and $\mathbf{b}_i^T$ is the $1 \times m$ vector $(0,\ldots,0,1,0,\ldots,0)$ with 1 in the i-th position. The Spatial BLUP $\tilde{\theta}_i(\omega)$ depends on the unknown vector of variance components $\omega = (\sigma_u^2, \rho)^T$. The two stage estimator $\tilde{\theta}_i(\hat{\omega})$ obtained by replacing ω in expression (2.6) by a consistent estimator $\hat{\omega} = (\hat{\sigma}_u^2, \hat{\rho})^T$ is called Spatial EBLUP (Singh et al., 2005; Petrucci & Salvati, 2006).

2.1.2 Fitting methods

A maximum likelihood estimator (MLE) of $\omega = (\sigma_u^2, \rho)^T$ is obtained maximizing the log-likelihood of ω given the data vector $\mathbf{y}$,

$$\ell(\omega;\mathbf{y}) = c - \frac{1}{2}\log|\mathbf{V}(\omega)| - \frac{1}{2}(\mathbf{y} - \mathbf{X}\beta)^T\mathbf{V}^{-1}(\omega)(\mathbf{y} - \mathbf{X}\beta),$$

where c denotes a constant. In practice, an iterative algorithm such as the Fisher-scoring algorithm must be applied to maximize the likelihood. Let $\mathbf{S}(\omega) = (S_{\sigma_u^2}, S_\rho)^T$ be the scores or derivatives of the log-likelihood with respect to σ_u^2 and ρ, and let $I(\omega)$ be the Fisher information matrix obtained from $\ell(\omega;\mathbf{y})$, with elements

$$I(\omega) = \begin{pmatrix} I_{\sigma_u^2,\sigma_u^2} & I_{\sigma_u^2,\rho^2} \\ I_{\rho,\sigma_u^2} & I_{\rho,\rho} \end{pmatrix}.$$

Then the Fisher-scoring algorithm starts with an initial estimate $\omega^{(0)} = (\sigma_u^{2(0)}, \rho^{(0)})^T$ and then at each iteration k, this estimate is updated with the equation

$$\omega^{(k+1)} = \omega^{(k)} + I^{-1}(\omega^{(k)})\mathbf{S}(\omega^{(k)}).$$

The ML equation for β obtained by equating the corresponding score to zero yields

$$\tilde{\beta}(\omega) = [\mathbf{X}^T\mathbf{V}^{-1}(\omega)\mathbf{X}]^{-1}\mathbf{X}^T\mathbf{V}^{-1}(\omega)\mathbf{y}. \tag{2.7}$$

Let us denote

$$\mathbf{C}(\rho) = (\mathbf{I}_m - \rho\mathbf{W})^T(\mathbf{I}_m - \rho\mathbf{W})$$

and

$$\mathbf{P}(\omega) = \mathbf{V}^{-1}(\omega) - \mathbf{V}^{-1}(\omega)\mathbf{X}\left[\mathbf{X}^T\mathbf{V}^{-1}(\omega)\mathbf{X}\right]^{-1}\mathbf{X}^T\mathbf{V}^{-1}(\omega).$$

Then the derivative of $\mathbf{C}(\rho)$ with respect to ρ is

$$\frac{\partial \mathbf{C}(\rho)}{\partial \rho} = -\mathbf{W} - \mathbf{W}^T + 2\rho \mathbf{W}^T\mathbf{W}$$

and the derivatives of $\mathbf{V}(\omega)$ with respect to σ_u^2 and ρ are respectively given by

$$\frac{\partial \mathbf{V}(\omega)}{\partial \sigma_u^2} = \mathbf{C}^{-1}(\rho), \quad \frac{\partial \mathbf{V}(\omega)}{\partial \rho} = -\sigma_u^2 \mathbf{C}^{-1}(\rho)\frac{\partial \mathbf{C}(\rho)}{\partial \rho}\mathbf{C}^{-1}(\rho) \triangleq \mathbf{A}(\omega).$$

The scores associated to σ_u^2 and ρ, after replacing (2.7), are given by

$$\begin{aligned} S_{\sigma_u^2} &= -\frac{1}{2}\operatorname{trace}\left\{\mathbf{V}^{-1}(\omega)\mathbf{C}^{-1}(\rho)\right\} + \frac{1}{2}\mathbf{y}^T\mathbf{P}(\omega)\mathbf{C}^{-1}(\rho)\mathbf{P}(\omega)\mathbf{y}, \\ S_{\rho} &= -\frac{1}{2}\operatorname{trace}\left\{\mathbf{V}^{-1}(\omega)\mathbf{A}^{-1}(\omega)\right\} + \frac{1}{2}\mathbf{y}^T\mathbf{P}(\omega)\mathbf{A}(\omega)\mathbf{P}(\omega)\mathbf{y}. \end{aligned}$$

The elements of the Fisher information matrix are

$$\begin{aligned} I_{\sigma_u^2,\sigma_u^2} &= \frac{1}{2}\operatorname{trace}\left\{\mathbf{V}^{-1}(\omega)\mathbf{C}^{-1}(\rho)\mathbf{V}^{-1}(\omega)\mathbf{C}^{-1}(\rho)\right\}, \\ I_{\sigma_u^2,\rho} = I_{\rho,\sigma_u^2} &= \frac{1}{2}\operatorname{trace}\left\{\mathbf{V}^{-1}(\omega)\mathbf{A}(\omega)\mathbf{V}^{-1}(\omega)\mathbf{C}^{-1}(\rho)\right\}, \\ I_{\rho,\rho} &= \frac{1}{2}\operatorname{trace}\left\{\mathbf{V}^{-1}(\omega)\mathbf{A}(\omega)\mathbf{V}^{-1}(\omega)\mathbf{A}(\omega)\right\}. \end{aligned}$$

In the function `fitSpatialFH`, the starting value of σ_u^2 is set to $\sigma_u^{2(0)} = \text{median}(\psi_i)$. For ρ, we take $\rho^{(0)} = 0.5$. The algorithm stops either when the number of iterations $k >$ MAXITER where MAXITER can be chosen by the user (default is 500), or when

$$\max\left\{\left|\frac{\sigma_u^{2(k+1)} - \sigma_u^{2(k)}}{\sigma_u^{2(k)}}\right|, \left|\frac{\rho^{(k+1)} - \rho^{(k)}}{\rho^{(k)}}\right|\right\} < 0.0001.$$

A restricted maximum likelihood estimator (RMLE) of ω is obtained by maximizing the restricted likelihood, which is the joint p.d.f. of a transformation of the response $\mathbf{y}$, that eliminates the vector of coefficients β. Let $\mathbf{F}$ be an $m \times p$ matrix satisfying $\mathbf{F}^T\mathbf{X} = \mathbf{0}$. Then, the restricted log-likelihood is the logarithm of the joint p.d.f. of the transformed data $\mathbf{F}^T\mathbf{y}$ and is given by

$$\ell_R(\omega;\mathbf{y}) = c - \frac{1}{2}\log|\mathbf{F}^T\mathbf{V}(\omega)\mathbf{F}| - \frac{1}{2}\mathbf{y}^T\mathbf{F}(\mathbf{F}^T\mathbf{V}(\omega)\mathbf{F})^{-1}\mathbf{F}^T\mathbf{y},$$

where

$$\mathbf{F}\left[\mathbf{F}^T\mathbf{V}(\omega)\mathbf{F}\right]^{-1}\mathbf{F}^T = \mathbf{P}(\omega),$$

so that the restricted log-likelihood becomes

$$\ell_R(\omega;\mathbf{y}) = c - \frac{1}{2}\log|\mathbf{F}^T\mathbf{V}(\omega)\mathbf{F}| - \frac{1}{2}\mathbf{y}^T\mathbf{P}(\omega)\mathbf{y}.$$

Using the following properties of the matrix $\mathbf{P}(\omega)$,

$$\mathbf{P}(\omega)\mathbf{V}(\omega)\mathbf{P}(\omega) = \mathbf{P}(\omega), \quad \frac{\partial \mathbf{P}(\omega)}{\partial \omega_j} = -\mathbf{P}(\omega)\frac{\partial \mathbf{V}(\omega)}{\partial \omega_j}\mathbf{P}(\omega),$$

we obtain the scores corresponding to this restricted log-likelihood,

$$S^R_{\sigma^2_u} = -\frac{1}{2}\operatorname{trace}\left\{\mathbf{P}(\omega)\mathbf{C}^{-1}(\rho)\right\} + \frac{1}{2}\mathbf{y}^T\mathbf{P}(\omega)\mathbf{C}^{-1}(\rho)\mathbf{P}(\omega)\mathbf{y},$$
$$S^R_{\rho} = -\frac{1}{2}\operatorname{trace}\left\{\mathbf{P}(\omega)\mathbf{A}(\omega)\right\} + \frac{1}{2}\mathbf{y}^T\mathbf{P}(\omega)\mathbf{A}(\omega)\mathbf{P}(\omega)\mathbf{y},$$

Finally, the elements of the Fisher information obtained from ℓ_R are

$$I^R_{\sigma^2_u,\sigma^2_u} = \frac{1}{2}\operatorname{tr}\{\mathbf{P}(\omega)\mathbf{C}^{-1}(\rho)\mathbf{P}(\omega)\mathbf{C}^{-1}(\rho)\},$$
$$I^R_{\sigma^2_u,\rho} = I^R_{\rho,\sigma^2_u} = \frac{1}{2}\operatorname{tr}\{\mathbf{P}(\omega)\mathbf{A}(\omega)\mathbf{P}(\omega)\mathbf{C}^{-1}(\rho)\},$$
$$I^R_{\rho,\rho} = \frac{1}{2}\operatorname{tr}\{\mathbf{P}(\omega)\mathbf{A}(\omega)\mathbf{P}(\omega)\mathbf{A}(\omega)\}.$$

Starting values and stopping criterion are set the same as in the case of ML estimates.

2.1.3 Mean squared error of the Spatial EBLUP

Again, under normality of random effects and errors, the MSE of the Spatial EBLUP can be decomposed as

$$\begin{array}{rcccl} \mathrm{MSE}[\tilde{\theta}_i(\hat{\omega})] = & \mathrm{MSE}[\tilde{\theta}_i(\omega)] & + & E\{[\tilde{\theta}_i(\hat{\omega}) - \tilde{\theta}_i(\omega)]^2\} \\ = & [g_{1i}(\omega) + g_{2i}(\omega)] & + & g_{3i}(\omega), \end{array} \tag{2.8}$$

where the first two terms on the right hand side are easily calculated due to the linearity of the Spatial BLUP $\tilde{\theta}_i(\omega)$ in the data vector $\mathbf{y}$. They are given by

$$g_{1i}(\omega) = \mathbf{b}_i^T[\mathbf{G}(\omega) - \mathbf{G}(\omega)\mathbf{V}^{-1}(\omega)\mathbf{G}(\omega)]\mathbf{b}_i, \tag{2.9}$$
$$g_{2i}(\omega) = \mathbf{b}_i^T[\mathbf{I}_m - \mathbf{G}(\omega)\mathbf{V}^{-1}(\omega)]\mathbf{X}(\mathbf{X}^T\mathbf{V}^{-1}(\omega)\mathbf{X})^{-1}\mathbf{X}^T[\mathbf{I}_m - \mathbf{V}^{-1}(\omega)\mathbf{G}(\omega)]\mathbf{b}_i. \tag{2.10}$$

However, for the last term $g_{3i}(\omega) = E\{[\tilde{\theta}_i(\hat{\omega}) - \tilde{\theta}_i(\omega)]^2\}$, an exact analytical expression does not exist due to the non-linearity of the EBLUP $\tilde{\theta}_i(\hat{\omega})$ in $\mathbf{y}$. Under the basic Fay-Herriot model (2.1) with independent random effects v_i (diagonal covariance matrix $\mathbf{V}$), Prasad & Rao (1990) obtained an approximation up to $o(m^{-1})$ terms of $g_{3i}(\omega)$ through Taylor linearization, see Section 1.1.3. Their formula can be taken as a naive approximation of the true $g_{3i}(\omega)$ under model (2.1)–(2.2). Straightforward application of this formula to model (2.1)–(2.2) yields

$$g^{PR}_{3i}(\omega) = \operatorname{trace}\left\{\mathbf{L}_i(\omega)\mathbf{V}(\omega)\mathbf{L}_i^T(\omega)\, I^{-1}(\omega)\right\},$$

where

$$\mathbf{L}_i(\omega) = \begin{pmatrix} \mathbf{b}_i^T\left[\mathbf{C}^{-1}(\rho)\mathbf{V}^{-1}(\omega) - \sigma^2_u\mathbf{C}^{-1}(\rho)\mathbf{V}^{-1}(\omega)\mathbf{C}^{-1}(\rho)\mathbf{V}^{-1}(\omega)\right] \\ \mathbf{b}_i^T\left[\mathbf{A}(\omega)\mathbf{V}^{-1}(\omega) - \sigma^2_u\mathbf{C}^{-1}(\rho)\mathbf{V}^{-1}(\omega)\mathbf{A}(\omega)\mathbf{V}^{-1}(\omega)\right] \end{pmatrix}.$$

Then the full MSE can be approximated by

$$\text{MSE}^{PR}[\tilde{\theta}_i(\hat{\omega})] = g_{1i}(\omega) + g_{2i}(\omega) + g_{3i}^{PR}(\omega). \tag{2.11}$$

Singh et al. (2005) arrived to the same formula (2.11) for the true MSE under a Fay-Herriot model with random effects following a SAR process. However, this formula is not accounting for the extra uncertainty of the Spatial EBLUP $\tilde{\theta}_i(\hat{\omega})$ due to the estimation of the autocorrelation parameter ρ.

As to MSE estimation, when $\hat{\omega}$ is obtained by REML method, Singh et al. (2005) derived the following MSE estimator

$$\text{mse}^{SSK}[\tilde{\theta}_i(\hat{\omega})] = g_{1i}(\hat{\omega}) + g_{2i}(\hat{\omega}) + 2g_{3i}^{PR}(\hat{\omega}) - g_{4i}(\hat{\omega}). \tag{2.12}$$

Here, $g_{4i}(\omega)$ is given by

$$g_{4i}(\omega) = \frac{1}{2}\sum_{k=1}^{2}\sum_{\ell=1}^{2} \mathbf{b}_i^T \Psi \mathbf{V}^{-1}(\omega) \frac{\partial^2 \mathbf{V}(\omega)}{\partial\omega_k \partial\omega_\ell} \mathbf{V}^{-1}(\omega) \Psi \, I_{k\ell}^{-1}(\omega) \mathbf{b}_i.$$

When $\hat{\omega}$ is obtained by ML, their estimator is

$$\text{mse}_{ML}^{SSK}[\tilde{\theta}_i(\hat{\omega})] = g_{1i}(\hat{\omega}) + g_{2i}(\hat{\omega}) + 2g_{3i}^{PR}(\hat{\omega}) - g_{4i}(\hat{\omega}) - \mathbf{b}_{ML}^T(\hat{\omega}) \nabla g_{1i}(\hat{\omega}), \tag{2.13}$$

where $\nabla g_{1i}(\omega) = \partial g_{1i}(\omega)/\partial\omega$ is the gradient of $g_{1i}(\omega)$ and $\mathbf{b}_{ML}(\hat{\omega})$ is the bias of the ML estimator $\hat{\omega}$ up to order $o(m^{-1})$. This bias is equal to $\mathbf{b}_{ML}(\hat{\omega}) = I^{-1}(\hat{\omega})\mathbf{h}(\hat{\omega})/2$ with $\mathbf{h}(\hat{\omega}) = (h_1(\hat{\omega}), h_2(\hat{\omega}))^T$ and

$$h_k(\omega) = \text{trace}\left\{ \left[\mathbf{X}^T \mathbf{V}^{-1}(\omega)\mathbf{X}\right]^{-1} \frac{\partial\left[\mathbf{X}^T \mathbf{V}^{-1}(\omega)\mathbf{X}\right]}{\partial\omega_k} \right\}, \quad k = 1, 2.$$

2.1.4 Parametric bootstrap mean squared error

Here we propose to use the parametric bootstrap of González-Manteiga et al. (2008) extended to the FH model with spatial correlation (2.1)–(2.2). The final MSE estimate obtained by this procedure is expected to be consistent if the model parameter estimates are consistent. This could be seen by the method of imitation as in González-Manteiga et al. (2008), using the asymptotic formula of the MSE obtained by Singh et al. (2005). This extended parametric bootstrap works as follows:

1) Fit model (2.5) to the initial data $\mathbf{y} = (y_1, \ldots, y_m)^T$, obtaining estimates $\hat{\omega} = (\hat{\sigma}_u^2, \hat{\rho})^T$ and $\hat{\beta} = \tilde{\beta}(\hat{\omega})$.

2) Generate a vector $\mathbf{t}_1^*$ whose elements are m independent copies of a $N(0,1)$. Construct bootstrap vectors $\mathbf{u}^* = \hat{\sigma}_u \mathbf{t}_1^*$ and $\mathbf{v}^* = (\mathbf{I}_m - \hat{\rho}\mathbf{W})^{-1}\mathbf{u}^*$, and calculate the bootstrap quantity of interest $\theta^* = \mathbf{X}\hat{\beta} + \mathbf{v}^*$, by regarding $\hat{\beta}$ and $\hat{\omega}$ as the true values of the parameters.

3) Generate a vector $\mathbf{t}_2^*$ with m independent copies of a $N(0,1)$, independently of the generation of $\mathbf{t}_1^*$, and construct the vector of random errors $\mathbf{e}^* = \Psi^{1/2}\mathbf{t}_2^*$.

4) Obtain bootstrap data $\mathbf{y}^*$ directly applying the model, $\mathbf{y}^* = \theta^* + \mathbf{e}^* = \mathbf{X}\hat{\beta} + \mathbf{v}^* + \mathbf{e}^*$.

5) Regarding $\hat{\beta}$ and $\hat{\omega}$ as the true values of β and ω, fit model (2.5) to bootstrap data $\mathbf{y}^*$, obtaining estimates of the "true" $\hat{\beta}$ and $\hat{\omega}$ based on bootstrap data $\mathbf{y}^*$: first, calculate the estimator of $\hat{\beta}$ calculated at the "true" $\hat{\omega}$,

$$\tilde{\beta}^*(\hat{\omega}) = \left[\mathbf{X}^T\mathbf{V}^{-1}(\hat{\omega})\mathbf{X}\right]^{-1}\mathbf{X}^T\mathbf{V}^{-1}(\hat{\omega})\mathbf{y}^*;$$

next, obtain the estimator $\hat{\omega}^*$ based on $\mathbf{y}^*$, and finally, the estimator of $\hat{\beta}$ calculated at $\hat{\omega}^*$, that is, $\tilde{\beta}^*(\hat{\omega}^*)$.

6) Calculate the bootstrap Spatial BLUP from bootstrap data $\mathbf{y}^*$ and regarding $\hat{\omega}$ as the true value of ω,

$$\tilde{\theta}_i^*(\hat{\omega}) = \mathbf{x}_i^T\tilde{\beta}^*(\hat{\omega}) + \mathbf{b}_i^T\mathbf{G}(\hat{\omega})\mathbf{V}(\hat{\omega})^{-1}[\mathbf{y}^* - \mathbf{X}\tilde{\beta}^*(\hat{\omega})].$$

Calculate also the bootstrap Spatial EBLUP using $\hat{\omega}^*$ in place of the "true" $\hat{\omega}$,

$$\tilde{\theta}_i^*(\hat{\omega}^*) = \mathbf{x}_i^T\tilde{\beta}^*(\hat{\omega}^*) + \mathbf{b}_i^T\mathbf{G}(\hat{\omega}^*)\mathbf{V}^{-1}(\hat{\omega}^*)[\mathbf{y}^* - \mathbf{X}\tilde{\beta}^*(\hat{\omega}^*)].$$

7) Repeat steps 2)–6) B times. In b-th bootstrap replication, let $\theta_i^{*(b)}$ be the quantity of interest for i-th area, $\hat{\omega}^{*(b)}$ the bootstrap estimate of ω, $\tilde{\theta}_i^{*(b)}(\hat{\omega})$ the bootstrap Spatial BLUP and $\tilde{\theta}_i^{*(b)}(\hat{\omega}^{*(b)})$ the bootstrap Spatial EBLUP for i-th area.

8) A parametric bootstrap estimator of $g_{3i}(\omega)$ is

$$g_{3i}^{PB}(\hat{\omega}) = B^{-1}\sum_{b=1}^{B}\left[\tilde{\theta}_i^{*(b)}(\hat{\omega}^{*(b)}) - \tilde{\theta}_i^{*(b)}(\hat{\omega})\right]^2.$$

Similarly, a naive parametric bootstrap estimator of the full MSE is

$$\text{mse}^{naPB}[\tilde{\theta}_i(\hat{\omega})] = B^{-1}\sum_{b=1}^{B}\left[\tilde{\theta}_i^{*(b)}(\hat{\omega}^{*(b)}) - \theta_i^{*(b)}\right]^2. \tag{2.14}$$

Another MSE estimate can be obtained as in Pfeffermann & Tiller (2006), by adding the analytical estimates $g_{1i}(\hat{\omega})$ and $g_{2i}(\hat{\omega})$, the bootstrap estimate $g_{3i}^{PB}(\hat{\omega})$, and a bootstrap bias correction of $g_{1i}(\hat{\omega}) + g_{2i}(\hat{\omega})$. The final estimator obtained in this way is

$$\text{mse}^{bcPB}[\tilde{\theta}_i(\hat{\omega})] = 2\left[g_{1i}(\hat{\omega}) + g_{2i}(\hat{\omega})\right] - B^{-1}\sum_{b=1}^{B}\left[g_{1i}(\hat{\omega}^{*(b)}) + g_{2i}(\hat{\omega}^{*(b)})\right] + g_{3i}^{PB}(\hat{\omega}). \tag{2.15}$$

2.1.5 Nonparametric bootstrap

This section describes a nonparametric bootstrap for MSE estimation, in which bootstrap random effects $\{u_1^*,\ldots,u_m^*\}$ and bootstrap random errors $\{e_1^*,\ldots,e_m^*\}$ are obtained by resampling respectively from the empirical distribution of predicted random effects $\{\hat{u}_1,\ldots,\hat{u}_m\}$ and of residuals $\{\hat{r}_1,\ldots,\hat{r}_m\}$, where $r_i = y_i - \tilde{\theta}_i(\hat{\omega})$, $i = 1,\ldots,m$, both previously standardized. This method avoids the need of distributional assumptions; therefore, it is expected to be more robust to non-normality of any of the random components of the model.

Under model (2.1)–(2.2), the BLUPs of $\mathbf{u}$ and $\mathbf{v}$ are respectively

$$\tilde{\mathbf{v}}(\omega) = \mathbf{G}(\omega)\mathbf{V}^{-1}(\omega)[\mathbf{y} - \mathbf{X}\tilde{\beta}(\omega)], \quad \tilde{\mathbf{u}}(\omega) = (\mathbf{I} - \rho\mathbf{W})\tilde{\mathbf{v}}(\omega),$$

and the covariance matrix of $\tilde{\mathbf{u}}(\omega)$ is

$$\mathbf{V}_{\mathbf{u}}(\omega) = (\mathbf{I} - \rho\mathbf{W})\mathbf{G}(\omega)\mathbf{P}(\omega)\mathbf{G}(\omega)(\mathbf{I} - \rho\mathbf{W}^T).$$

Moreover, consider the vector of residuals

$$\tilde{\mathbf{r}}(\omega) = \mathbf{y} - \mathbf{X}\tilde{\beta}(\omega) - \tilde{\mathbf{v}}(\omega) = (y_1 - \tilde{\theta}_1(\omega), \ldots, y_m - \tilde{\theta}_m(\omega))^T.$$

It is easy to see that the covariance matrix of $\tilde{\mathbf{r}}(\omega)$ is

$$\mathbf{V}_{\mathbf{r}}(\omega) = \Psi\mathbf{P}(\omega)\Psi.$$

The covariance matrices $\mathbf{V}_{\mathbf{u}}(\omega)$ and $\mathbf{V}_{\mathbf{r}}(\omega)$ are not diagonal; hence, the elements of the vectors $\tilde{\mathbf{u}}(\omega)$ and $\tilde{\mathbf{r}}(\omega)$ are correlated. Indeed, both $\tilde{\mathbf{u}}(\omega)$ and $\tilde{\mathbf{r}}(\omega)$ lie in a space of dimension $m - p$. Since the methods that resample from the empirical distribution work well under an ideally *iid* setup, when applying these methods, a previous standardization step is crucial. Here we propose to transform both $\hat{\mathbf{u}} = \tilde{\mathbf{u}}(\hat{\omega})$ and $\hat{\mathbf{r}} = \tilde{\mathbf{r}}(\hat{\omega})$ to make them as close as possible to vectors with uncorrelated and unit variance elements. We describe the standardization method only for $\hat{\mathbf{u}}$, since for $\hat{\mathbf{r}}$ the process is analogous. Let us consider the estimated covariance matrix $\hat{\mathbf{V}}_{\mathbf{u}} = \mathbf{V}_{\mathbf{u}}(\hat{\omega})$. The method works by carrying out the spectral decomposition of $\hat{\mathbf{V}}_{\mathbf{u}}$,

$$\hat{\mathbf{V}}_{\mathbf{u}} = \mathbf{Q}_{\mathbf{u}}\Delta_{\mathbf{u}}\mathbf{Q}_{\mathbf{u}}^T,$$

where $\Delta_{\mathbf{u}}$ is a diagonal matrix with the $m - p$ non-zero eigenvalues of $\hat{\mathbf{V}}_{\mathbf{u}}$ and $\mathbf{Q}_{\mathbf{u}}$ is the matrix with the corresponding eigenvectors in the columns. Then we take the matrix $\hat{\mathbf{V}}_{\mathbf{u}}^{-1/2} = \mathbf{Q}_{\mathbf{u}}\Delta_{\mathbf{u}}^{-1/2}\mathbf{Q}_{\mathbf{u}}^T$. Squaring this matrix gives a generalized inverse of $\hat{\mathbf{V}}_{\mathbf{u}}$. With the obtained square root, we transform $\hat{\mathbf{u}}$ as

$$\hat{\mathbf{u}}^S = \hat{\mathbf{V}}_{\mathbf{u}}^{-1/2}\hat{\mathbf{u}}.$$

The covariance matrix of $\hat{\mathbf{u}}^S$ is then $Var(\hat{\mathbf{u}}^S) = \mathbf{Q}_{\mathbf{u}}\mathbf{Q}_{\mathbf{u}}^T$, which is close to an identity matrix. Observe that in the transformation

$$\hat{\mathbf{u}}^S = \mathbf{Q}_{\mathbf{u}}\Delta_{\mathbf{u}}^{-1/2}\mathbf{Q}_{\mathbf{u}}^T\hat{\mathbf{u}},$$

the vector $\mathbf{Q}_{\mathbf{u}}^T\hat{\mathbf{u}}$ contains the coordinates of $\hat{\mathbf{u}}$ in its principal components, which are uncorrelated with covariance matrix $\Delta_{\mathbf{u}}$. Then multiplying by $\Delta_{\mathbf{u}}^{-1/2}$, these coordinates are standardized to have unit variance. Finally, this standardized vector in the space of the principal components is returned to the original space by multiplying by $\mathbf{Q}_{\mathbf{u}}$. Thus, the transformed vector $\hat{\mathbf{u}}^S$ contains the coordinates of the vector $\Delta_{\mathbf{u}}^{-1/2}\mathbf{Q}_{\mathbf{u}}^T\hat{\mathbf{u}}$, with standard elements, in the original space. The eigenvalues, which are the variances of the uncorrelated principal components, collect better the variability than the diagonals of $\hat{\mathbf{V}}_{\mathbf{u}}$. Indeed, simulations were also carried out standardizing simply by taking $\hat{u}_i^S = \hat{u}_i/\sqrt{v_{ii}}$, where v_{ii} is the i-the diagonal element of $\hat{\mathbf{V}}_{\mathbf{u}}$, but the resulting nonparametric bootstrap did not work well.

The final nonparametric bootstrap procedure works by replacing steps 2) and 3) of the parametric bootstrap by the new steps 2') and 3') below:

2') With the estimates $\hat{\omega} = (\hat{\sigma}_u^2, \hat{\rho})^T$ and $\hat{\beta} = \tilde{\beta}(\hat{\omega})$ obtained in step 1), calculate predictors of $\mathbf{v}$ and $\mathbf{u}$ as follows

$$\hat{\mathbf{v}} = \mathbf{G}(\hat{\omega})\mathbf{V}(\hat{\omega})^{-1}(\mathbf{y} - \mathbf{X}\hat{\beta}), \quad \hat{\mathbf{u}} = (\mathbf{I} - \hat{\rho}\mathbf{W})\hat{\mathbf{v}} = (\hat{u}_1, \ldots, \hat{u}_m)^T.$$

Then take $\hat{\mathbf{u}}^S = \hat{\mathbf{V}}_{\mathbf{u}}^{-1/2}\hat{\mathbf{u}} = (\hat{u}_1^S, \ldots, \hat{u}_m^S)^T$, where $\hat{\mathbf{V}}_{\mathbf{u}}^{1/2}$ is the square root of the generalized inverse of $\hat{\mathbf{V}}_{\mathbf{u}}$ obtained by the spectral decomposition. It is convenient to re-scale the elements $\hat{u}_i^S$ so that they have sample mean exactly equal to zero and sample variance $\hat{\sigma}_u^2$. This is achieved by the transformation

$$\hat{u}_i^{SS} = \frac{\hat{\sigma}_u(\hat{u}_i^S - m^{-1}\sum_{j=1}^m \hat{u}_j^S)}{\sqrt{m^{-1}\sum_{d=1}^m (\hat{u}_d^S - m^{-1}\sum_{j=1}^m \hat{u}_j^S)^2}}, \quad i = 1, \ldots, m.$$

Construct the vector $\mathbf{u}^* = (u_1^*, \ldots, u_m^*)^T$, whose elements are obtained by extracting a simple random sample with replacement of size m from the set $\{\hat{u}_1^{SS}, \ldots, \hat{u}_m^{SS}\}$. Then obtain $\mathbf{v}^* = (\mathbf{I} - \hat{\rho}\mathbf{W})^{-1}\mathbf{u}^*$ and calculate the bootstrap quantity of interest $\theta^* = \mathbf{X}\hat{\beta} + \mathbf{v}^* = (\theta_1^*, \ldots, \theta_m^*)^T$

3') Compute the vector of residuals $\hat{\mathbf{r}} = \mathbf{y} - \mathbf{X}\hat{\beta} - \hat{\mathbf{v}} = (\hat{r}_1, \ldots, \hat{r}_m)^T$. Standardize the residuals by $\hat{\mathbf{r}}^S = \hat{\mathbf{V}}_{\mathbf{r}}^{-1/2}\hat{\mathbf{r}} = (\hat{r}_1^S, \ldots, \hat{r}_m^S)^T$, where $\hat{\mathbf{V}}_{\mathbf{r}} = \Psi\mathbf{P}(\hat{\omega})\Psi$ is the estimated covariance matrix and $\hat{\mathbf{V}}_{\mathbf{r}}^{-1/2}$ is a root square of the generalized inverse derived from the spectral decomposition of $\hat{\mathbf{V}}_{\mathbf{r}}$. Again, re-standardize these values

$$\hat{r}_i^{SS} = \frac{\hat{r}_i^S - m^{-1}\sum_{j=1}^m \hat{r}_j^S}{\sqrt{m^{-1}\sum_{d=1}^m (\hat{r}_d^S - m^{-1}\sum_{j=1}^m \hat{r}_j^S)^2}}, \quad i = 1, \ldots, m.$$

Construct $\mathbf{r}^* = (r_1^*, \ldots, r_m^*)^T$ by extracting a simple random sample with replacement of size m from the set $\{\hat{r}_1^{SS}, \ldots, \hat{r}_m^{SS}\}$. Then take $\mathbf{e}^* = (e_1^*, \ldots, e_m^*)^T$, where $e_i^* = \psi_i^{1/2} r_i^*$, $i = 1, \ldots, m$.

This procedure yields naive and bias-corrected nonparametric bootstrap estimators analogous to (2.14) and (2.15). They are respectively denoted as $\text{mse}^{naNPB}[\tilde{\theta}_i(\hat{\omega})]$ and $\text{mse}^{bcNPB}[\tilde{\theta}_i(\hat{\omega})]$.

2.2 The Software: description of R functions

This section describes the implemented R functions that fit the area-level spatial model with random effects following a SAR process as defined in (2.5), give the spatial EBLUP small area estimators and provide analytical, parametric bootstrap and nonparametric bootstrap MSE estimators. A brief description of these R functions is given in the rest of this section. An example showing the use of these functions is provided in the next section and full R codes are included in Appendix 2.

2.2.1 fitSpatialFH

R function `fitSpatialFH` fits the area-level spatial model with random effects following a SAR process, as defined in (2.5), using REML fitting method. The function is defined as

```
fitSpatialFH<-function(X,y,Dvec,W,method="REML",MAXITER=500)
```

Arguments of this function are:

`X`: matrix containing the aggregated (population) values of p auxiliary variables, with dimension $m \times p$, where m is the number of areas or sample size. The elements in first column might be equal to 1 if the model includes an intercept.

`y`: vector containing the direct estimates of the response variable for the m areas.

`Dvec`: vector containing the m sampling variances $D_1, \ldots, D_m$ of direct estimators.

`W`: $m \times m$ proximity matrix with rows adding to one.

`method`: type of fitting method, to be chosen between REML or ML methods. Default is REML method.

`MAXITER`: maximum number of iterations allowed to the Fisher-scoring algorithm. Default is 500 iterations.

The function returns a list with the following objects:

`convergence`: a logical value equal to TRUE if Fisher-scoring algorithm converges in less than MAXITER iterations.

`modelcoefficients`: data.frame in the shape of a table with the estimated model coefficients in first column, their asymptotic standard errors in second column, the Z statistics in third column and the p-values of the significance of each coefficient in last column.

`variance`: estimated random effects variance σ_u^2.

`spatialcorr`: estimated spatial correlation parameter ρ.

`goodnessoffit`: a vector containing three different goodness-of-fit measures, namely the loglikelihood, the AIC and the BIC.

`EBpredictor`: a vector of size m with the values of the EB predictor for the m areas.

2.2.2 MSE.SpatialFHmodel

R function `MSE.SpatialFHmodel` gives the analytical MSE estimates of the Spatial EBLUPs for the m small areas obtained from the Spatial Fay-Herriot model (2.5). This function is defined as

```
MSE.SpatialFHmodel<-function(X,Dvec,A,rho,W,method="REML")
```

Arguments of this function are:

`X`: matrix containing the aggregated (population) values of p auxiliary variables, with dimension $m \times p$, where m is the number of areas or sample size. The elements in first column might be equal to 1 if the model includes an intercept.

`Dvec`: vector containing the m sampling variances $D_1, \ldots, D_m$ of direct estimators.

`A`: estimated random effects variance σ_u^2 obtained using the fitting method specified in `method`.

`rho`: estimated spatial autocorrelation parameter obtained using the fitting method specified in `method`.

`W`: proximity matrix with rows adding to one.

`method`: type of fitting method. Currently only REML method is available.

The function returns:

`mse`: a vector with the analytical MSE estimates of the Spatial EBLUP small area estimators.

2.2.3 PBMSE.SpatialFHmodel

R function `PBMSE.SpatialFHmodel` gives parametric bootstrap MSE estimates of the Spatial EBLUPs for the m small areas obtained from the Spatial Fay-Herriot model (2.5). This function is defined as

```
PBMSE.SpatialFHmodel<-function(X,Dvec,beta,A,rho,W,n.boot,method="REML")
```

Arguments of this function are:

`X`: matrix containing the aggregated (population) values of p auxiliary variables, with dimension $m \times p$, where m is the number of areas or sample size. The elements in first column might be equal to 1 if the model includes an intercept.

`Dvec`: vector containing the m sampling variances $D_1, \ldots, D_m$ of direct estimators.

`beta`: estimated regression coefficients β obtained using the fitting method specified in `method`.

`A`: estimated random effects variance σ_u^2 obtained using the fitting method specified in `method`.

`rho`: estimated spatial autocorrelation parameter obtained using the fitting method specified in `method`.

`W`: proximity matrix with rows adding to one.

`n.boot`: number of bootstrap replicates.

`method`: type of fitting method. Currently only REML method is available.

The function returns a data frame containing the following two vectors:

`PBmse`: naive parametric bootstrap MSE estimates of the Spatial EBLUP small area estimators.

`bcPBmse`: bias-corrected parametric bootstrap MSE estimates of the Spatial EBLUP small area estimators.

2.3 Examples of usage of R functions

This section shows how to use the R functions described in Section 1.2 to produce small area EB estimators along with their corresponding estimated MSE, based on the basic Fay-Herriot model (1.1).

2.3.1 Example data set

The example data set used to show the use of functions `fitSpatialFH` and `MSE.SpatialFHmodel` has been obtained drawing a sample from the Italian Agricultural Census of year 2000, see Molina et al. (2008) for a description of the data generation. The small areas are $m = 274$ municipalities of Tuscany and a proximity matrix was obtained from the neighborhood structure of these municipalities, with one assigned to the pairs of municipalities that share a common edge. Data set for the first ten municipalities is shown below. The response variable with the direct estimates for the m municipalities is grapehect, the auxiliary variables are area and workdays, and var is the sampling error of direct estimators.

```
       grapehect       area  workdays          var
1     30.9477582  203.93775  73.95884 2.153054e+01
2     57.2161416  187.22505 148.21444 2.014299e+02
3     73.7540672  590.73016 171.34921 2.818218e+00
4     66.2420341  318.32333 105.86333 2.138089e+01
5     36.9317990  217.25789  87.83813 6.424393e+01
6     78.5339314 1562.02985 202.68657 1.629171e-01
7     50.4567217  101.47826  93.48161 1.473757e+01
8     41.9724571  147.48728  85.71624 1.950657e+01
9    111.5708810  274.27101 233.63077 3.293872e+02
10    10.2326214  145.89855  32.67029 6.659302e-03
...
```

The proximity matrix for the first 10 municipalities is

```
     V1        V2        V3        V4        V5        V6        V7
1  0.00 0.3333333 0.3333333 0.0000000 0.0000000 0.0000000 0.0000000
2  0.25 0.0000000 0.0000000 0.2500000 0.0000000 0.0000000 0.2500000
3  0.50 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
4  0.00 0.3333333 0.0000000 0.0000000 0.3333333 0.0000000 0.3333333
5  0.00 0.0000000 0.0000000 0.1428571 0.0000000 0.1428571 0.1428571
6  0.00 0.0000000 0.0000000 0.0000000 0.5000000 0.0000000 0.0000000
7  0.00 0.2000000 0.0000000 0.2000000 0.2000000 0.0000000 0.0000000
8  0.20 0.2000000 0.2000000 0.0000000 0.0000000 0.0000000 0.2000000
9  0.00 0.0000000 0.0000000 0.0000000 0.1111111 0.1111111 0.0000000
10 0.00 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
```

```
          V8        V9       V10 ...
1  0.3333333 0.0000000 0.0000000
2  0.2500000 0.0000000 0.0000000
3  0.5000000 0.0000000 0.0000000
4  0.0000000 0.0000000 0.0000000
5  0.0000000 0.1428571 0.0000000
6  0.0000000 0.5000000 0.0000000
7  0.2000000 0.0000000 0.0000000
8  0.0000000 0.0000000 0.0000000
9  0.0000000 0.0000000 0.1111111
10 0.0000000 0.2500000 0.0000000
...
```

2.3.2 Example of R code for running function fitSpatialFH

Here we include an example of R code used to read the data set from the Italian Agriculture Census listed in Section 2.3.1 and run function `fitFH` using that data. R code includes suitable comments explaining what is each line doing.

```
# Set the Path or folder where data set and functions are.
setwd("Path")

# Read data set
data<-read.table("SimulSpatData_ItAgricSurvey2000.txt",header=TRUE)
attach(data)

# Load file where function is located
source("Fitting_SpatialFHModel.R")

# Read file with proximity matrix and assign it to a matrix called prox.
prox<-as.matrix(read.table("ProximitySpatData_ItAgricSurvey2000.txt"))

# Define matrix with area-level auxiliary variables
X<-as.matrix(cbind(area,workdays))

# Fit function using REML method and put the output in the list resultsSp.

resultsSp<-fitSpatialFH(X,grapehect,var,prox,method="ML")
print(resultsSp)
```

2.3.3 Output of function fitSpatialFH

Output of function `fitSpatialFH` when setting

```
method="ML"
```

is given below. Spatial EB predictors are given only for the first 10 small areas for brevity.

```
-----------------------------------------------------
$convergence
[1] TRUE

$modelcoefficients
               Bstim std.errorbeta    tvalue       pvalue
area     -0.01241993   0.002095302 -5.927514 3.075547e-09
workdays  0.49861050   0.012732816 39.159485 0.000000e+00

$variance
[1] 67.69242

$spatialcorr
[1] 0.6550672

$EBpredictor
            [,1]
V1    31.2591747
V2    71.9504581
V3    73.8753880
V4    62.2488593
V5    39.6819534
V6    78.5365740
V7    50.1555221
V8    41.2015557
V9   109.2337422
V10   10.2332637
...
-----------------------------------------------------
```

2.3.4 Example of R code for running function MSE.SpatialFHmodel

Here we include an example of R code used to read the data set from the Italian Agriculture Census in Section 2.3.1, fit the Spatial Fay-Herriot model to that data set using function `fitSpatialFH` which

gives also the small area EB estimators, and finally run function `MSE.SpatialFHmodel` to obtain analytical MSE estimates of Spatial EBLUPs for the $m = 274$ municipalities. R code includes suitable comments explaining what is each line doing.

```
# Set the Path or folder where data set and functions are.
setwd("Path")

# Read data set
data<-read.table("SimulSpatData_ItAgricSurvey2000.txt",header=TRUE)
attach(data)

# Read proximity matrix and assign it to a matrix called prox
prox<-as.matrix(read.table("ProximitySpatData_ItAgricSurvey2000.txt"))

# Create matrix with area-level auxiliary variables
X<-as.matrix(cbind(area,workdays))

# Load files where functions are
source("Fitting_SpatialFHModel.R")
source("MSE_SpatialFHModel.R")

# Fit the Spatial Fay-Herriot model with REML method
results<-fitSpatialFH(X,grapehect,var,prox)

# Obtain analytical MSE estimates of Spatial EBLUPs
mse<-MSE.SpatialFHmodel(X,var,results$variance,results$spatialcorr,prox)
mse
```

2.3.5 Output of function MSE.SpatialFHmodel

Output of function `MSE.SpatialFHmodel` when setting

```
method="REML"
```

is given below for the first 10 small areas:

```
------------------------------------------------------------------------
[1] 1.661451e+01 5.180644e+01 2.721105e+00 1.692434e+01 3.133056e+01
[6] 1.626345e-01 1.226893e+01 1.508771e+01 4.901339e+01 6.658713e-03 ...
------------------------------------------------------------------------
```

2.3.6 Example of R code for running function PBMSE.SpatialFHmodel

Here we include an example of R code used to read the data set from the Italian Agriculture Census in Section 2.3.1, fit the Spatial Fay-Herriot model to that data set using function `fitSpatialFH` which gives also the small area EB estimators, and finally run function `PBMSE.SpatialFHmodel` to obtain parametric bootstrap MSE estimates of Spatial EBLUPs for the $m = 274$ municipalities. R code includes suitable comments explaining what is each line doing.

```
# Set the Path or folder where data set and functions are.
setwd("Path")

# Read data set
data<-read.table("SimulSpatData_ItAgricSurvey2000.txt",header=TRUE)
attach(data)

# Read proximity matrix
prox<-as.matrix(read.table("ProximitySpatData_ItAgricSurvey2000.txt"))

# Create the matrix with area-level auxiliary variables
X<-as.matrix(cbind(area,workdays))

# Load files with needed functions
source("Fitting_SpatialFHModel.R")
source("PBMSE_SpatialFHModel.R")

# Fit the Spatial Fay-Herriot model by REML method
results<-fitSpatialFH(X,grapehect,var,prox)

# Take the estimated model coefficients
coef<-results$modelcoefficients$beta

# Take the estimated random effects variance
A<-results$variance

# Take the estimated spatial autocorrelation parameter
rho<-results$spatialcorr

# Obtain the naive and bias-corrected parametric bootstrap mse estimates
PBmse<-PBMSE.SpatialFHmodel(X,var,coef,A,rho,prox,100)
PBmse$PBmse
```

2.3.7 Output of function PBMSE.SpatialFHmodel

Output of function `MSE.SpatialFHmodel` when setting

`method="REML"`

is given below for the first 10 small areas:

```
------------------------------------------------------------------
[1] 1.88266e+01 4.48179e+01 2.33316e+00 1.65272e+01 3.86748e+01
[6] 1.58263e-01 1.11911e+01 1.24805e+01 4.63717e+01 4.57704e-03 ...
------------------------------------------------------------------
```

Chapter 3

Area-level time models

3.1 Area-level model with independent time effects

3.1.1 The methodology

Consider the model

$$y_{dt} = \mathbf{x}_{dt}\beta + u_{dt} + e_{dt}, \quad d = 1,\dots,D, \quad t = 1,\dots,m_d, \tag{3.1}$$

where y_{dt} is a direct estimator of the indicator of interest for area d and time instant t, and $\mathbf{x}_{dt}$ is a vector containing the aggregated (population) values of p auxiliary variables. The index d is used for domains and the index t for time instants. We assume that the vectors u_{dt}'s are $N(0,\sigma_u^2)$, the errors e_{dt}'s are independent $N(0,\sigma_{dt}^2)$ with known variances σ_{dt}^2, and the u_{dt}'s are independent of the e_{dt}'s. Model (3.1) can be alternatively written in the form

$$\mathbf{y} = \mathbf{X}\beta + \mathbf{Z}\mathbf{u} + \mathbf{e}, \tag{3.2}$$

where $\mathbf{y} = \underset{1\le d\le D}{\text{col}}(\mathbf{y}_d)$, $\mathbf{y}_d = \underset{1\le t\le m_d}{\text{col}}(y_{dt})$, $\mathbf{u} = \underset{1\le d\le D}{\text{col}}(\mathbf{u}_d)$, $\mathbf{u}_d = \underset{1\le t\le m_d}{\text{col}}(u_{dt})$, $\mathbf{e} = \underset{1\le d\le D}{\text{col}}(\mathbf{e}_d)$, $\mathbf{e}_d = \underset{1\le t\le m_d}{\text{col}}(e_{dt})$, $\mathbf{X} = \underset{1\le d\le D}{\text{col}}(\mathbf{X}_d)$, $\mathbf{X}_d = \underset{1\le t\le m_d}{\text{col}}(\mathbf{x}_{dt})$, $\mathbf{x}_{dt} = \underset{1\le i\le p}{\text{col}'}(x_{dti})$, $\beta = \underset{1\le i\le p}{\text{col}}(\beta_i)$, $\mathbf{Z} = \mathbf{I}_M$, $M = \sum_{d=1}^{D} m_d$. We assume that $\mathbf{u} \sim N(\mathbf{0},\mathbf{V}_u)$ and $\mathbf{e} \sim N(\mathbf{0},\mathbf{V}_e)$ are independent with covariance matrices

$$\mathbf{V}_u = \sigma_u^2\mathbf{I}_M, \quad \mathbf{I}_M = \underset{1\le d\le D}{\text{diag}}(\mathbf{I}_{m_d}), \quad \mathbf{V}_e = \underset{1\le d\le D}{\text{diag}}(\mathbf{V}_{ed}), \quad \mathbf{V}_{ed} = \underset{1\le t\le m_d}{\text{col}}(\sigma_{dt}^2),$$

and known variances σ_{dt}^2. The BLUE of β and the BLUP of $\mathbf{u}$ are

$$\widehat{\beta} = (\mathbf{X}'\mathbf{V}^{-1}\mathbf{X})^{-1}\mathbf{X}'\mathbf{V}^{-1}\mathbf{y} \qquad \text{and} \qquad \widehat{\mathbf{u}} = \mathbf{V}_u\mathbf{Z}'\mathbf{V}^{-1}(\mathbf{y} - \mathbf{X}\widehat{\beta}),$$

where

$$var(\mathbf{y}) = \mathbf{V} = \sigma_u^2 \underset{1\le d\le D}{\text{diag}}(\mathbf{I}_{m_d}) + \mathbf{V}_e = \underset{1\le d\le D}{\text{diag}}(\sigma_u^2\mathbf{I}_{m_d} + \mathbf{V}_{ed}) = \underset{1\le d\le D}{\text{diag}}(\mathbf{V}_d).$$

The estimator $\widehat{\beta}$ and the predictor $\widehat{\mathbf{u}}$ are calculated by applying the formulas

$$\widehat{\beta} = \left(\sum_{d=1}^{D}\mathbf{X}_d'\mathbf{V}_d^{-1}\mathbf{X}_d\right)^{-1}\left(\sum_{d=1}^{D}\mathbf{X}_d'\mathbf{V}_d^{-1}\mathbf{y}_d\right) \qquad \text{and} \qquad \widehat{\mathbf{u}} = \sigma_u^2 \underset{1\le d\le D}{\text{col}}\left(\mathbf{V}_d^{-1}(\mathbf{y}_d - \mathbf{X}_d\widehat{\beta})\right).$$

The Henderson 3 estimators of σ_u^2 is

$$\widehat{\sigma}_{uH}^2 = \frac{\mathbf{y}'\mathbf{P}_2\mathbf{y} - (M-p)}{\mathrm{tr}\{\mathbf{P}_2\}},$$

where $\mathbf{Q}_2 = \sum_{d=1}^{D}\left(\mathbf{X}_d'\mathbf{V}_{ed}^{-1}\mathbf{X}_d\right)^{-1}$ and

$$\begin{aligned}
\mathbf{P}_2 &= \underset{1\le d\le D}{\mathrm{diag}}\,(\mathbf{V}_{ed}^{-1}) - \underset{1\le d\le D}{\mathrm{col}}(\mathbf{V}_{ed}^{-1}\mathbf{X}_d)\mathbf{Q}_2 \underset{1\le d\le D}{\mathrm{col}'}(\mathbf{X}_d'\mathbf{V}_{ed}^{-1}),\\
\mathrm{tr}\{\mathbf{P}_2\} &= \sum_{d=1}^{D}\sum_{t=1}^{m_d}\sigma_{dt}^{-2} - \sum_{d=1}^{D}\mathrm{tr}\{\mathbf{X}_d'\mathbf{V}_{ed}^{-2}\mathbf{X}_d\mathbf{Q}_2\},\\
\mathbf{y}'\mathbf{P}_2\mathbf{y} &= \sum_{d=1}^{D}\sum_{t=1}^{m_d}\sigma_{dt}^{-2}y_{dt}^2 - \left(\sum_{d=1}^{D}\mathbf{y}_d'\mathbf{V}_{ed}^{-1}\mathbf{X}_d\right)\mathbf{Q}_2\left(\sum_{d=1}^{D}\mathbf{y}_d'\mathbf{V}_{ed}^{-1}\mathbf{X}_d\right)'.
\end{aligned}$$

The REML estimators are calculated by using the Fisher-scoring algorithm with the updating formula

$$\theta^{k+1} = \theta^k + F^{-1}(\theta^k)S(\theta^k),$$

where $\theta = \sigma_u^2$. The Henderson 3 estimator $\widehat{\sigma}_{uH}^2$ is used as seed of the Fisher-scoring algorithm. The REML score and Fisher amount of information are

$$S = S(\theta) = -\frac{1}{2}\mathrm{tr}(\mathbf{P}) + \frac{1}{2}\mathbf{y}'\mathbf{P}^2\mathbf{y} \quad \text{and} \quad F = F(\theta) = \frac{1}{2}\mathrm{tr}(\mathbf{P}^2),$$

where $\mathbf{Q} = \left(\sum_{d=1}^{D}\mathbf{X}_d'\mathbf{V}_d^{-1}\mathbf{X}_d\right)^{-1}$ and

$$\begin{aligned}
\mathbf{P} &= \underset{1\le d\le D}{\mathrm{diag}}\,(\mathbf{V}_d^{-1}) - \underset{1\le d\le D}{\mathrm{col}}(\mathbf{V}_d^{-1}\mathbf{X}_d)\mathbf{Q}\underset{1\le d\le D}{\mathrm{col}'}(\mathbf{X}_d'\mathbf{V}_d^{-1}),\\
\mathrm{tr}(\mathbf{P}) &= \sum_{d=1}^{D}\mathrm{tr}(\mathbf{V}_d^{-1}) - \sum_{d=1}^{D}\mathrm{tr}(\mathbf{X}_d'\mathbf{V}_d^{-2}\mathbf{X}_d\mathbf{Q}),\\
\mathrm{tr}(\mathbf{P}^2) &= \sum_{d=1}^{D}\mathrm{tr}(\mathbf{V}_d^{-2}) - 2\sum_{d=1}^{D}\mathrm{tr}(\mathbf{X}_d'\mathbf{V}_d^{-3}\mathbf{X}_d\mathbf{Q})\\
&+ \mathrm{tr}\left\{\left(\sum_{d=1}^{D}\mathbf{X}_d'\mathbf{V}_d^{-2}\mathbf{X}_d\right)\mathbf{Q}\left(\sum_{d=1}^{D}\mathbf{X}_d'\mathbf{V}_d^{-2}\mathbf{X}_d\right)\mathbf{Q}\right\},\\
\mathbf{y}'\mathbf{P}^2\mathbf{y} &= \sum_{d=1}^{D}\mathbf{y}_d'\mathbf{V}_d^{-2}\mathbf{y}_d - 2\left(\sum_{d=1}^{D}\mathbf{y}_d'\mathbf{V}_d^{-1}\mathbf{X}_d\right)\mathbf{Q}\left(\sum_{d=1}^{D}\mathbf{X}_d'\mathbf{V}_d^{-2}\mathbf{y}_d\right)\\
&+ \left(\sum_{d=1}^{D}\mathbf{y}_d'\mathbf{V}_d^{-1}\mathbf{X}_d\right)\mathbf{Q}\left(\sum_{d=1}^{D}\mathbf{X}_d'\mathbf{V}_d^{-2}\mathbf{X}_d\right)\mathbf{Q}\left(\sum_{d=1}^{D}\mathbf{y}_d'\mathbf{V}_d^{-1}\mathbf{X}_d\right)'.
\end{aligned}$$

The REML estimator of β is

$$\widehat{\beta}_{REML} = (\mathbf{X}'\widehat{\mathbf{V}}^{-1}\mathbf{X})^{-1}\mathbf{X}'\widehat{\mathbf{V}}^{-1}\mathbf{y}.$$

The asymptotic distributions of the REML estimators of σ_u^2 and β are

$$\hat{\sigma}_u^2 \sim N(\theta, F^{-1}(\sigma_u^2)), \quad \hat{\beta} \sim N_p(\beta, (\mathbf{X}'\mathbf{V}^{-1}\mathbf{X})^{-1}).$$

Asymptotic confidence intervals at the level $1-\alpha$ for σ_u^2 and β_i are

$$\hat{\sigma}_u^2 \pm z_{\alpha/2}\, \nu^{1/2}, \quad \hat{\beta}_i \pm z_{\alpha/2}\, q_{ii}^{1/2},\ i = 1, \ldots, p,$$

where $\hat{\sigma}_u^2 = \sigma_u^{2,(\kappa)}$, $\nu = F^{-1}(\sigma_u^{2,(\kappa)})$, $(\mathbf{X}'\mathbf{V}^{-1}(\sigma_u^{2,(\kappa)})\mathbf{X})^{-1} = (q_{ij})_{i,j=1,\ldots,p}$, κ is the final iteration of the Fisher-scoring algorithm and z_α is the α-quantile of the standard normal distribution $N(0,1)$. Observed $\hat{\beta}_i = \beta_0$, the p-value for testing the hypothesis $H_0 : \beta_i = 0$ is

$$p = 2P_{H_0}(\hat{\beta}_i > |\beta_0|) = 2P(N(0,1) > |\beta_0|/\sqrt{q_{ii}}).$$

We are interested in predicting $\mu_{dt} = \mathbf{x}_{dt}\beta + u_{dt}$ with the EBLUP $\widehat{\mu}_{dt} = \mathbf{x}_{dt}\widehat{\beta} + \widehat{u}_{dt}$. If we do not take into account the error e_{dt}, this is equivalent to predict $y_{dt} = \mathbf{a}'\mathbf{y}$, where $\mathbf{a} = \underset{1\le\ell\le D}{\text{col}}\,(\underset{1\le k\le m_\ell}{\text{col}}\,(\delta_{d\ell}\delta_{tk}))$ is a vector having one "1" in the cell $t + \sum_{\ell=1}^{d-1} m_\ell$ and "0"'s in the remaining cells. The total $\overline{Y}_{dt}$ is estimated with $\widehat{\overline{Y}}_{dt}^{eblup} = \widehat{\mu}_{dt}$. The mean squared error of $\widehat{\overline{Y}}_{dt}^{eblup}$ is

$$MSE(\widehat{\overline{Y}}_{dt}^{eblup}) = g_1(\sigma_u^2) + g_2(\sigma_u^2) + g_3(\sigma_u^2),$$

and the estimator of $MSE(\widehat{\overline{Y}}_{dt}^{eblup})$ is

$$mse(\widehat{\overline{Y}}_{dt}^{eblup}) = g_1(\hat{\sigma}_u^2) + g_2(\hat{\sigma}_u^2) + 2g_3(\hat{\sigma}_u^2),$$

where

$$\begin{aligned}
g_1(\sigma_u^2) &= \frac{\sigma_u^2\sigma_{dt}^2}{\sigma_u^2+\sigma_{dt}^2}, \\
g_2(\sigma_u^2) &= \left[\mathbf{a}_d'\mathbf{X}_d - \sigma_u^2\mathbf{a}_d'\mathbf{V}_{ed}^{-1}\mathbf{X}_d + \sigma_u^4\mathbf{a}_d'\mathbf{V}_d^{-1}\mathbf{V}_{ed}^{-1}\mathbf{X}_d\right]\mathbf{Q} \\
&\cdot \left[\mathbf{X}_d'\mathbf{a}_d - \sigma_u^2\mathbf{X}_d'\mathbf{V}_{ed}^{-1}\mathbf{a}_d + \sigma_u^4\mathbf{X}_d'\mathbf{V}_{ed}^{-1}\mathbf{V}_d^{-1}\mathbf{a}_d\right], \quad \mathbf{a}_d = \underset{1\le k\le m_d}{\text{col}}\,(\delta_{tk}), \\
g_3(\sigma_u^2) &= qF^{-1}(\sigma_u^2), \quad q = \frac{1}{\sigma_u^2+\sigma_{dt}^2} - \frac{2\sigma_u^2}{\left(\sigma_u^2+\sigma_{dt}^2\right)^2} + \frac{\sigma_u^4}{\left(\sigma_u^2+\sigma_{dt}^2\right)^3}
\end{aligned}$$

and F is the REML Fisher amount of information calculated by the updating equation of the Fisher-scoring algorithm.

3.1.2 The Software: description of R functions

This section describes the R functions that have been implemented for fitting the area-level model with independent time effects (3.1). An example of how to use these functions is given in the next section and the related codes are listed in Appendix 3.1.

The developed R software contains a series of functions that return, as final output, the EBLUP estimates of interest. We recall that R functions are objects with the form

$$name \ <- \ function\ (arg_1, arg_2, ...)\ \{expression\}.$$

R functions allows to define a dependent variable *name* as output of a given procedure, when inputs variables are *arguments*. The *expression* within curly brackets contains the needed calculations to obtain *name* from *arguments*. The function codes appearing in *expression* are listed in Appendix A.

A brief descriptions of programmed R functions is given in the next subsections. The functions can be used for calculating the Henderson 3 and the REML variance estimates, the β estimate, the $\mathbf{u}$ predictor, the EBLUPs and the MSEs of EBLUPs.

H3area

Function **H3area** calculates the unbiased Henderson 3 estimator of σ_u^2 and has the form

$$H3.area \ <- \ function\ (X,\ ydt,\ D,\ md,\ sigma2edt).$$

The arguments are:

X: matrix containing the aggregated (population) values of p auxiliary variables, with dimension $M \times p$. First column elements should be equal to 1 if the model includes intercept.

ydt: vector containing the direct estimates of the dependent variable for area d and time instant t, with size M.

D: total number of domains.

md: vector containing the time instants totals m_d within each domain, with size D.

sigma2edt: vector containing the known error variances σ_{ed}^2, with size M.

The function returns the value of the Henderson 3 estimate of $\hat{\sigma}_u^2$.

REMLarea.indep

Function **REMLarea.indep** calculates the estimate of σ_u^2 and the Fisher amount of information F for the Restricted Maximum Likelihood (REML) method. The function is

$$REMLarea.indep \ <- \ function\ (X, ydt, D, md, sigma2edt, sigma.0, MAXITER = 500).$$

The arguments are:

X: matrix containing the aggregated (population) values of p auxiliary variables, with dimension $M \times p$. First column elements should be equal to 1 if the model includes intercept.

ydt: vector containing the direct estimates of the dependent variable for area d and time instant t, with size M.

D: total number of domains.

md: vector containing the time instants totals m_d within each domain, with size D.

sigma2edt: vector containing the error variances σ^2_{ed}, with size M.

sigma.0: Henderson 3 estimate of σ^2_u obtained as output of **H3area**. It is used as seed of the Fisher-scoring algorithm.

MAXITER: maximum number of iterations in the Fisher-scoring algorithm. Default value is 500.

The function returns a list of three elements. First element **sigma.f** is the REML estimate of σ^2_u, second element **Fsig** is the estimated Fisher amount of information F and third element **Q** is the inverse matrix appearing in the expression of $\hat{\beta}$.

BETA.U.area.indep

Function **BETA.U.area.indep** calculates the estimator $\hat{\beta}$ and the predictor $\hat{\mathbf{u}}$. The function is

$$BETA.U.area.indep <- function\ (X,\ ydt,\ D,\ md,\ sigma2edt,\ sigmau).$$

The arguments are:

X: matrix containing the aggregated (population) values of p auxiliary variables, with dimension $M \times p$. First column elements should be equal to 1 if the model includes intercept.

ydt: vector containing the direct estimates of the variable of interest for area d and time instant t, with size M.

D: total number of domains.

md: vector containing the time instants totals m_d within each domain, with size D.

sigma2edt: vector containing the error variances σ^2_{ed}, with size M.

sigmau: estimated value of σ^2_u, calculated by the function **REMLarea.indep**.

The function returns an array with the following elements:

beta: vector containing the estimated regression parameters $\widehat{\beta}$, with size p.

u: vector containing the predicted random effects $\widehat{\mathbf{u}}$, with size M.

mse.area.indep

Function **mse.area.indep** calculates the estimator of the Mean Squared Error (MSE) of the EBLUP $\widehat{\mu}_{dt} = \mathbf{x}_{dt}\widehat{\beta} + \widehat{u}_{dt}$. The function is

$$mse.area.indep \ <- \ function\ (X,\ D,\ md,\ sigma2edt,\ sigmau,\ Fsig).$$

The arguments are:

X: matrix containing the aggregated (population) values of p auxiliary variables, with dimension $M \times p$. First column elements should be equal to 1 if the model includes intercept.

D: total number of domains.

md: vector containing the time instants totals m_d within each domain, with size D.

sigma2edt: vector containing the error variances σ^2_{ed}, with size M.

sigmau: estimated value of σ^2_u, calculated by the function **REMLarea.indep**.

Fsig: estimated Fisher amount of information, calculated by the function **REMLarea.indep**.

The function returns a vector containing the MSE estimates $mse(\widehat{\overline{Y}}^{eblup}_{dt})$, with size M.

Interval.indep

Function **Interval.indep** calculates the asymptotic confidence intervals for σ^2_u and β_i. The function is

$$Interval.indep \ <- \ function\ (fit,\ conf = 0.95).$$

The arguments are:

fit: returned object, obtained by applying the function **REMLarea.indep**.

conf: interval confidence level $1-\alpha$. Default value is 0.95.

This function returns the semi-lengths **sigma.std.err** and **beta.std.err** of the asymptotic confidence intervals for σ^2_u and β_i respectively.

pvalue

Function **pvalue** calculates the asymptotic p-value of test statistics $\hat{\beta}_i$ for the null hypothesis $H_0 : \beta_i = 0$. The function is

$$pvalue \ <- \ function\ (beta0,\ fit).$$

The arguments are:

beta0: observed value of $\hat{\beta}_i$, calculated by the function **BETA.U.area.indep**.

fit: returned object, obtained by applying the function **REMLarea.indep**.

This function returns the vector **pval** containing the asymptotic p-values for hypotheses $H_0 : \beta_i = 0$, $i = 1, \ldots, p$, with size p.

3.2 Area-level model with correlated time effects

3.2.1 The methodology

Consider the model

$$y_{dt} = \mathbf{x}_{dt}\beta + u_{dt} + e_{dt}, \quad d = 1,\dots,D, \quad t = 1,\dots,m_d, \tag{3.3}$$

where y_{dt} is a direct estimator of the indicator of interest for area d and time instant t, and $\mathbf{x}_{dt}$ is a vector containing the aggregated (population) values of p auxiliary variables. The index d is used for domains and the index t for time instants. We further assume that the random vectors $(u_{d1},\dots,u_{dm_d})$, $d = 1,\dots,D$, follow i.i.d. AR(1) processes with variance and auto-correlation parameters σ_u^2 and ρ respectively, the errors e_{dt}'s are independent $N(0,\sigma_{dt}^2)$ with known variances σ_{dt}^2, and the u_{dt}'s are independent of the e_{dt}'s.

In matrix notation the model is

$$\mathbf{y} = \mathbf{X}\beta + \mathbf{Z}\mathbf{u} + \mathbf{e}, \tag{3.4}$$

where $\mathbf{y} = \underset{1\le d\le D}{\text{col}}(\mathbf{y}_d)$, $\mathbf{y}_d = \underset{1\le t\le m_d}{\text{col}}(y_{dt})$, $\mathbf{u} = \underset{1\le d\le D}{\text{col}}(\mathbf{u}_d)$, $\mathbf{u}_d = \underset{1\le t\le m_d}{\text{col}}(u_{dt})$, $\mathbf{e} = \underset{1\le d\le D}{\text{col}}(\mathbf{e}_d)$, $\mathbf{e}_d = \underset{1\le t\le m_d}{\text{col}}(e_{dt})$, $\mathbf{X} = \underset{1\le d\le D}{\text{col}}(\mathbf{X}_d)$, $\mathbf{X}_d = \underset{1\le t\le m_d}{\text{col}}(\mathbf{x}_{dt})$, $\mathbf{x}_{dt} = \underset{1\le i\le p}{\text{col}'}(x_{dti})$, $\beta = \underset{1\le i\le p}{\text{col}}(\beta_i)$, $\mathbf{Z} = \mathbf{I}_{M\times M}$ and $M = \sum_{d=1}^{D} m_d$. In this notation, $\mathbf{u} \sim N(\mathbf{0},\mathbf{V}_u)$ and $\mathbf{e} \sim N(\mathbf{0},\mathbf{V}_e)$ are independent with covariance matrices

$$\mathbf{V}_u = \sigma_u^2\Omega(\rho), \quad \Omega(\rho) = \underset{1\le d\le D}{\text{diag}}\,(\Omega_d(\rho)), \quad \mathbf{V}_e = \underset{1\le d\le D}{\text{diag}}\,(\mathbf{V}_{ed}), \quad \mathbf{V}_{ed} = \underset{1\le t\le m_d}{\text{diag}}\,(\sigma_{dt}^2),$$

where the σ_{dt}^2 are known and

$$\Omega_d = \Omega_d(\rho) = \frac{1}{1-\rho^2}\begin{pmatrix} 1 & \rho & \dots & \rho^{m_d-2} & \rho^{m_d-1} \\ \rho & 1 & \ddots & & \rho^{m_d-2} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \rho^{m_d-2} & & \ddots & 1 & \rho \\ \rho^{m_d-1} & \rho^{m_d-2} & \dots & \rho & 1 \end{pmatrix}_{m_d\times m_d}.$$

If the variance components are known, then the BLUE of β and the BLUP of $\mathbf{u}$ are

$$\widehat{\beta} = (\mathbf{X}'\mathbf{V}^{-1}\mathbf{X})^{-1}\mathbf{X}'\mathbf{V}^{-1}\mathbf{y} \qquad \text{and} \qquad \widehat{\mathbf{u}} = \mathbf{V}_u\mathbf{Z}'\mathbf{V}^{-1}(\mathbf{y} - \mathbf{X}\widehat{\beta}),$$

where

$$var(\mathbf{y}) = \mathbf{V} = \sigma_u^2 \underset{1\le d\le D}{\text{diag}}\,(\Omega_d(\rho)) + \mathbf{V}_e = \underset{1\le d\le D}{\text{diag}}\,(\sigma_u^2\Omega_d(\rho) + \mathbf{V}_{ed}) = \underset{1\le d\le D}{\text{diag}}\,(\mathbf{V}_d).$$

The estimator $\widehat{\beta}$ and the predictor $\widehat{\mathbf{u}}$ are calculated by applying the formulas

$$\widehat{\beta} = \left(\sum_{d=1}^{D}\mathbf{X}_d'\mathbf{V}_d^{-1}\mathbf{X}_d\right)^{-1}\left(\sum_{d=1}^{D}\mathbf{X}_d'\mathbf{V}_d^{-1}\mathbf{y}_d\right), \quad \widehat{\mathbf{u}} = \sigma_u^2 \underset{1\le d\le D}{\text{col}}\left(\Omega_d(\rho)\mathbf{V}_d^{-1}(\mathbf{y}_d - \mathbf{X}_d\widehat{\beta})\right).$$

If the variance components are unknown, their REML estimators are calculated by using the Fisher-scoring algorithm with the updating formula

$$\theta^{k+1} = \theta^k + \mathbf{F}^{-1}(\theta^k)\mathbf{S}(\theta^k),$$

where $\theta = (\theta_1, \theta_2) = (\sigma_u^2, \rho)$. As seeds we use $\rho = 0$ and $\sigma_u^{2(0)} = \widehat{\sigma}_{uH}^2$, where $\widehat{\sigma}_{uH}^2$ is the Henderson 3 estimator of σ_u^2 under the model restricted to $\rho = 0$. The REML scores and components of the Fisher information matrix are

$$S_a = -\frac{1}{2}\text{tr}(\mathbf{P}\mathbf{V}_a) + \frac{1}{2}\mathbf{y}'\mathbf{P}\mathbf{V}_a\mathbf{P}\mathbf{y}, \quad F_{ab} = \frac{1}{2}\text{tr}(\mathbf{P}\mathbf{V}_a\mathbf{P}\mathbf{V}_b), \quad a,b = 1,2.$$

where $\mathbf{V}_1 = \frac{\partial \mathbf{V}}{\partial \sigma_u^2} = \underset{1\le d\le D}{\text{diag}}\,(\Omega_d(\rho))$, $\mathbf{V}_2 = \frac{\partial \mathbf{V}}{\partial \rho} = \sigma_u^2 \underset{1\le d\le D}{\text{diag}}\,(\dot{\Omega}_d(\rho))$, $\mathbf{Q} = \left(\sum_{d=1}^D \mathbf{X}_d'\mathbf{V}_d^{-1}\mathbf{X}_d\right)^{-1}$,

$$\begin{aligned}
\mathbf{P} &= \underset{1\le d\le D}{\text{diag}}\,(\mathbf{V}_d^{-1}) - \underset{1\le d\le D}{\text{col}}\,(\mathbf{V}_d^{-1}\mathbf{X}_d)\mathbf{Q}\,\underset{1\le d\le D}{\text{col}}'\,(\mathbf{X}_d'\mathbf{V}_d^{-1}),\\
\mathbf{P}\mathbf{V}_a &= \underset{1\le d\le D}{\text{diag}}\,(\mathbf{V}_d^{-1}\mathbf{V}_{ad}) - \underset{1\le d\le D}{\text{col}}\,(\mathbf{V}_d^{-1}\mathbf{X}_d)\mathbf{Q}\,\underset{1\le d\le D}{\text{col}}'\,(\mathbf{X}_d'\mathbf{V}_d^{-1}\mathbf{V}_{ad}),\\
\text{tr}(\mathbf{P}\mathbf{V}_a) &= \sum_{d=1}^D \text{tr}(\mathbf{V}_d^{-1}\mathbf{V}_{ad}) - \sum_{d=1}^D \text{tr}(\mathbf{X}_d'\mathbf{V}_d^{-1}\mathbf{V}_{ad}\mathbf{V}_d^{-1}\mathbf{X}_d\mathbf{Q}),\\
\text{tr}(\mathbf{P}\mathbf{V}_a\mathbf{P}\mathbf{V}_b) &= \sum_{d=1}^D \text{tr}(\mathbf{V}_d^{-1}\mathbf{V}_{ad}\mathbf{V}_d^{-1}\mathbf{V}_{bd}) - 2\sum_{d=1}^D \text{tr}(\mathbf{X}_d'\mathbf{V}_d^{-1}\mathbf{V}_{ad}\mathbf{V}_d^{-1}\mathbf{V}_{bd}\mathbf{V}_d^{-1}\mathbf{X}_d\mathbf{Q})\\
&+ \text{tr}\left\{\left(\sum_{d=1}^D \mathbf{X}_d'\mathbf{V}_d^{-1}\mathbf{V}_{ad}\mathbf{V}_d^{-1}\mathbf{X}_d\right)\mathbf{Q}\left(\sum_{d=1}^D \mathbf{X}_d'\mathbf{V}_d^{-1}\mathbf{V}_{bd}\mathbf{V}_d^{-1}\mathbf{X}_d\right)\mathbf{Q}\right\}.\\
\mathbf{y}'\mathbf{P}\mathbf{V}_a\mathbf{P}\mathbf{y} &= \sum_{d=1}^D \mathbf{y}_d'\mathbf{V}_d^{-1}\mathbf{V}_{ad}\mathbf{V}_d^{-1}\mathbf{y}_d - \left(\sum_{d=1}^D \mathbf{y}_d'\mathbf{V}_d^{-1}\mathbf{V}_{ad}\mathbf{V}_d^{-1}\mathbf{X}_d\right)\mathbf{Q}\left(\sum_{d=1}^D \mathbf{y}_d'\mathbf{V}_d^{-1}\mathbf{X}_d\right)'\\
&- \left(\sum_{d=1}^D \mathbf{y}_d'\mathbf{V}_d^{-1}\mathbf{X}_d\right)\mathbf{Q}\left(\sum_{d=1}^D \mathbf{X}_d'\mathbf{V}_d^{-1}\mathbf{V}_{ad}\mathbf{V}_d^{-1}\mathbf{y}_d\right)\\
&+ \left(\sum_{d=1}^D \mathbf{y}_d'\mathbf{V}_d^{-1}\mathbf{X}_d\right)\mathbf{Q}\left(\sum_{d=1}^D \mathbf{X}_d'\mathbf{V}_d^{-1}\mathbf{V}_{ad}\mathbf{V}_d^{-1}\mathbf{X}_d\right)\mathbf{Q}\left(\sum_{d=1}^D \mathbf{y}_d'\mathbf{V}_d^{-1}\mathbf{X}_d\right)'.
\end{aligned}$$

Finally, the derivative of matrix $\Omega_d(\rho)$ with respect to ρ is

$$\dot{\Omega}_d(\rho) = \frac{1}{1-\rho^2}\begin{pmatrix} 0 & 1 & \dots & \dots & (m_d-1)\rho^{m_d-2} \\ 1 & 0 & \ddots & & (m_d-2)\rho^{m_d-3} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ (m_d-2)\rho^{m_d-3} & & \ddots & 0 & 1 \\ (m_d-1)\rho^{m_d-2} & \dots & \dots & 1 & 0 \end{pmatrix} + \frac{2\rho\Omega_d(\rho)}{(1-\rho^2)^2}.$$

The REML estimator of β is calculated by applying the formula

$$\widehat{\beta}_{REML} = (\mathbf{X}'\widehat{\mathbf{V}}^{-1}\mathbf{X})^{-1}\mathbf{X}'\widehat{\mathbf{V}}^{-1}\mathbf{y}.$$

The asymptotic distributions of the REML estimators of θ and β are

$$\hat{\theta} \sim N_2(\theta, \mathbf{F}^{-1}(\theta)), \quad \hat{\beta} \sim N_p(\beta, (\mathbf{X}'\mathbf{V}^{-1}\mathbf{X})^{-1}).$$

Asymptotic confidence intervals at the level $1-\alpha$ for θ_a and β_i are

$$\hat{\theta}_a \pm z_{\alpha/2}\, \nu_{aa}^{1/2},\ a = 1,2, \quad \hat{\beta}_i \pm z_{\alpha/2}\, q_{ii}^{1/2},\ i = 1,\ldots,p,$$

where $\hat{\theta} = \theta^{\kappa}$, $\mathbf{F}^{-1}(\theta^{\kappa}) = (\nu_{ab})_{a,b=1,2}$, $(\mathbf{X}'\mathbf{V}^{-1}(\theta^{\kappa})\mathbf{X})^{-1} = (q_{ij})_{i,j=1,\ldots,p}$, κ is the final iteration of the Fisher-scoring algorithm and z_α is the α-quantile of the standard normal distribution $N(0,1)$. Observed $\hat{\beta}_i = \beta_0$, the p-value for testing the hypothesis $H_0 : \beta_i = 0$ is

$$p = 2P_{H_0}(\hat{\beta}_i > |\beta_0|) = 2P(N(0,1) > |\beta_0|/\sqrt{q_{ii}}).$$

We are interested in predicting $\mu_{dt} = \mathbf{x}_{dt}\beta + u_{dt}$ with the EBLUP $\widehat{\mu}_{dt} = \mathbf{x}_{dt}\widehat{\beta} + \widehat{u}_{dt}$. If we do not take into account the error, e_{dt}, this is equivalent to predict $y_{dt} = \mathbf{a}'\mathbf{y}$, where $\mathbf{a} = \underset{1\le\ell\le D}{\text{col}}\big(\underset{1\le k\le m_\ell}{\text{col}}(\delta_{d\ell}\delta_{tk})\big)$ is a vector having one 1 in the position $t + \sum_{\ell=1}^{d-1} m_\ell$ and 0's in the remaining cells. To estimate $\overline{Y}_{dt}$ we use $\widehat{\overline{Y}}_{dt}^{eblup} = \widehat{\mu}_{dt}$. The mean squared error of $\widehat{\overline{Y}}_{dt}^{eblup}$ is

$$MSE(\widehat{\overline{Y}}_{dt}^{eblup}) = g_1(\theta) + g_2(\theta) + g_3(\theta),$$

and the estimator of $MSE(\widehat{\overline{Y}}_{dt}^{eblup})$ is

$$mse(\widehat{\overline{Y}}_{dt}^{eblup}) = g_1(\hat{\theta}) + g_2(\hat{\theta}) + 2g_3(\hat{\theta}),$$

where $\theta = (\sigma_u^2, \rho)$, $\mathbf{a}_d = \underset{1\le k\le m_d}{\text{col}}(\delta_{tk})$. The expressions for g_1-g_3 are

$$\begin{aligned}
g_1(\theta) &= \sigma_u^2 \mathbf{a}_d'\Omega_d\mathbf{a}_d - \sigma_u^4 \mathbf{a}_d'\Omega_d\mathbf{V}_d^{-1}\Omega_d\mathbf{a}_d,\\
g_2(\theta) &= \left[\mathbf{a}_d'\mathbf{X}_d - \sigma_u^2\mathbf{a}_d'\Omega_d\mathbf{V}_{ed}^{-1}\mathbf{X}_d + \sigma_u^4\mathbf{a}_d'\Omega_d\mathbf{V}_d^{-1}\Omega_d\mathbf{V}_{ed}^{-1}\mathbf{X}_d\right]\mathbf{Q}\\
&\cdot \left[\mathbf{X}_d'\mathbf{a}_d - \sigma_u^2\mathbf{X}_d'\mathbf{V}_{ed}^{-1}\Omega_d\mathbf{a}_d + \sigma_u^4\mathbf{X}_d'\mathbf{V}_{ed}^{-1}\Omega_d\mathbf{V}_d^{-1}\Omega_d\mathbf{a}_d\right],\\
g_3(\theta) &\approx \text{tr}\left\{\begin{pmatrix} q_{11} & q_{12}\\ q_{21} & q_{22}\end{pmatrix}\begin{pmatrix} F_{11} & F_{12}\\ F_{21} & F_{22}\end{pmatrix}^{-1}\right\},
\end{aligned}$$

where F_{ab} is the element of the REML Fisher information matrix.

$$\begin{aligned}
q_{11} &= \mathbf{a}_d'\Omega_d\mathbf{V}_d^{-1}\Omega_d\mathbf{a}_d - 2\sigma_u^2\mathbf{a}_d'\Omega_d\mathbf{V}_d^{-1}\Omega_d\mathbf{V}_d^{-1}\Omega_d\mathbf{a}_d + \sigma_u^4\mathbf{a}_d'\Omega_d\mathbf{V}_d^{-1}\Omega_d\mathbf{V}_d^{-1}\Omega_d\mathbf{V}_d^{-1}\Omega_d\mathbf{a}_d,\\
q_{12} &= \sigma_u^2\mathbf{a}_d'\Omega_d\mathbf{V}_d^{-1}\dot{\Omega}_d\mathbf{a}_d - \sigma_u^4\mathbf{a}_d'\Omega_d\mathbf{V}_d^{-1}\dot{\Omega}_d\mathbf{V}_d^{-1}\Omega_d\mathbf{a}_d - \sigma_u^4\mathbf{a}_d'\Omega_d\mathbf{V}_d^{-1}\Omega_d\mathbf{V}_d^{-1}\dot{\Omega}_d\mathbf{a}_d\\
&+ \sigma_u^6\mathbf{a}_d'\Omega_d\mathbf{V}_d^{-1}\Omega_d\mathbf{V}_d^{-1}\dot{\Omega}_d\mathbf{V}_d^{-1}\Omega_d\mathbf{a}_d,\\
q_{22} &= \sigma_u^4\mathbf{a}_d'\dot{\Omega}_d\mathbf{V}_d^{-1}\dot{\Omega}_d\mathbf{a}_d - 2\sigma_u^6\mathbf{a}_d'\Omega_d\mathbf{V}_d^{-1}\dot{\Omega}_d\mathbf{V}_d^{-1}\dot{\Omega}_d\mathbf{a}_d + \sigma_u^8\mathbf{a}_d'\Omega_d\mathbf{V}_d^{-1}\dot{\Omega}_d\mathbf{V}_d^{-1}\dot{\Omega}_d\mathbf{V}_d^{-1}\Omega_d\mathbf{a}_d.
\end{aligned}$$

3.2.2 The Software: description of R functions

This section describes the R functions that have been implemented for fitting the area-level model with time correlated effects (3.3). A brief descriptions of programmed R functions is given in the next subsections and the related codes are listed in Appendix B. The function can be used for calculating the REML variance estimates, the β estimate, the **u** predictor, the EBLUPs and the MSEs of EBLUPs.

REMLarea.autocorr

Function **REMLarea.autocorr** calculates the estimate of σ_u^2, the correlation coefficient ρ and the Fisher information matrix F for the Restricted Maximum Likelihood (REML) method. The function is

$$REMLarea.autocorr <- function\ (X, ydt, D, md, sigma2edt, sigma.0, MAXITER = 500).$$

The arguments are:

X: matrix containing the aggregated (population) values of p auxiliary variables, with dimension $M \times p$. First column elements should be equal to 1 if the model includes intercept.

ydt: vector containing the direct estimates of the dependent variable for area d and time instant t, with size M.

D: total number of domains.

md: vector containing the time instants totals m_d within each domain, with size D.

sigma2edt: vector containing the error variances σ_{ed}^2, with size M.

sigma.0: Henderson 3 estimate of σ_u^2 obtained as output of **H3area** (see Section 3.1.2). It is used as seed of the Fisher-scoring algorithm.

MAXITER: maximum number of iterations for the Fisher-scoring algorithm. Default value is 500.

The function returns a list of three elements. First element **theta.f** is the vector containing the REML estimates of σ_u^2 and ρ, second element **Fsig** is the estimated Fisher information matrix F and third element **Q** is the inverse matrix appearing in the expression of $\hat{\beta}$.

BETA.U.area.autocorr

Function **BETA.U.area.autocorr** calculates the estimator $\hat{\beta}$ and the predictor $\hat{\mathbf{u}}$. The function is

$$BETA.U.area.autocorr <- function\ (X,\ ydt,\ D,\ md,\ sigma2edt,\ sigmau,\ rho).$$

The arguments are:

X: matrix containing the aggregated (population) values of p auxiliary variables, with dimension $M \times p$. First column elements should be equal to 1 if the model includes intercept.

ydt: vector containing the direct estimates of the variable of interest for area d and time instant t, with size M.

D: total number of domains.

md: vector containing the time instants totals m_d within each domain, with size D.

sigma2edt: vector containing the error variances σ^2_{ed}, with size M.

sigmau: estimated value of σ^2_u, calculated by the function **REMLarea.autocorr**.

rho: estimated value of ρ, calculated by the function **REMLarea.autocorr**.

The function returns an array with the following elements:

beta: vector containing the estimated regression parameters $\widehat{\beta}$, with size p.

u: vector containing the predicted random effects $\widehat{\mathbf{u}}$, with size M.

mse.area.autocorr

Function **mse.area.autocorr** calculates the estimator of the Mean Squared Error (MSE) of the EBLUP $\widehat{\mu}_{dt} = \mathbf{x}_{dt}\widehat{\beta} + \widehat{u}_{dt}$. The function is

$$mse.area.autocorr <- function\ (X,\ D,\ md,\ sigma2edt,\ sigmau,\ rho,\ Fsig).$$

The arguments are:

X: matrix containing the aggregated (population) values of p auxiliary variables, with dimension $M \times p$. First column elements should be equal to 1 if the model includes intercept.

D: total number of domains.

md: vector containing the time instants totals m_d within each domain, with size D.

sigma2edt: vector containing the error variances σ^2_{ed}, with size M.

sigmau: estimated value of σ^2_u, calculated by the function **REMLarea.autocorr**.

rho: estimated value of ρ, calculated by the function **REMLarea.autocorr**.

Fsig: estimated Fisher amount of information, calculated by the function **REMLarea.autocorr**.

The function returns a vector containing the MSE estimates $mse(\widehat{\overline{Y}}^{eblup}_{dt})$, with size M.

Interval.autocorr

Function **Intervalo.autocorr** calculates the asymptotic confidence intervals for σ_u^2 and β_i. The function is

$$Interval.autocorr \ <- \ function\ (fit,\ conf = 0.95).$$

The arguments are:

fit: returned object, obtained by applying the function **REMLarea.autocorr**.

conf: interval confidence level $1-\alpha$. Default value is 0.95.

This function returns the semi-lengths **sigma.std.err** and **beta.std.err** of the asymptotic confidence intervals for σ_u^2 and β_i respectively.

3.3 Examples of usage of R functions

This section demonstrates how the R routines described in chapters 3.1 and 3.2 can be applied to produce EBLUP estimates with their corresponding mean squared errors.

3.3.1 Example data set

Table 3.3.1.1 presents the data sets used in the example. There are 10 domains (areas), 3 time periods, 2 independent variables $X1$ and $X2$ for each domain and time period. Dependent variables is labeled by Y and the error variances by Var. There are are 30 observations. The file *dataExample.txt* contains the data, which should be sorted by domains and time periods.

Domain	*Time*	*ones*	$X1$	$X2$	Y	*Var*
11	1	1	0.414586518	0.188617372	0.07099622	0.000675053
11	2	1	0.410214326	0.196059524	0.072288837	0.000751033
11	3	1	0.430158306	0.203894195	0.082660682	0.001154806
12	1	1	0.416790574	0.191078693	0.146031284	0.001563715
12	2	1	0.39922138	0.186915951	0.082499785	0.000930011
12	3	1	0.430940532	0.187417706	0.078653447	0.001043772
21	1	1	0.419196102	0.148285289	0.301060532	0.001080062
21	2	1	0.402663325	0.148051767	0.241819175	0.000959909
21	3	1	0.399746159	0.144933539	0.237411987	0.001252906
22	1	1	0.385341527	0.150625093	0.305820376	0.001155839
22	2	1	0.381030859	0.154456747	0.263558025	0.001045748
22	3	1	0.371354065	0.151396774	0.285240927	0.00138958
31	1	1	0.405546324	0.163096412	0.179595403	0.000279276
31	2	1	0.418451155	0.160096774	0.171986758	0.000268098
31	3	1	0.422061454	0.152057299	0.160000387	0.000288642
32	1	1	0.38297778	0.172748583	0.179514009	0.000248195
32	2	1	0.392681888	0.163776148	0.183061881	0.000272611
32	3	1	0.401865322	0.159065292	0.188520565	0.000326651

Table 3.3.1.1. Data set *dataExample*.

Domain	*Time*	*ones*	$X1$	$X2$	Y	*Var*
41	1	1	0.449063901	0.127355842	0.353368205	0.001370052
41	2	1	0.452458366	0.123193508	0.249208975	0.001059773
41	3	1	0.45310808	0.129271701	0.317849199	0.001248913
42	1	1	0.438922319	0.131555109	0.335358187	0.001132905
42	2	1	0.426280132	0.140372377	0.322904936	0.001211021
42	3	1	0.441941803	0.12645344	0.353882106	0.001397947
51	1	1	0.375257147	0.168546499	0.208500731	0.002511227
51	2	1	0.384939766	0.152016046	0.329334406	0.00405833
51	3	1	0.389536232	0.170682902	0.335023497	0.00482337
52	1	1	0.350720286	0.151162941	0.204206531	0.002439803
52	2	1	0.373013185	0.133152685	0.389675949	0.004361985
52	3	1	0.381826881	0.145456107	0.452922541	0.005475034

Table 3.3.1.1. Data set *dataExample* (continuation).

3.3.2 Example of R code

An R code for reading the data file and applying the above described functions is needed. The file *Example.R* contains this code and, for this example, is located in the folder *C:/Areatimemodel*. It is important to take care on where to put this file and all the function *R* files. Note that under *Windows system*, the folder of the file is set by default installation. Otherwise the user can type the complete path. But under *Linux* the folder is the same than the one used to execute R.

The R file *Example.R* contains a program with the instructions for fitting the area level model to data in file *dataExample.txt*. First step is to open the *R* files containing all the above described R functions, i.e. *H3.R, REMLindep.R, REMLautocorr.R, EstimationBETAindep.R, EstimationBETAautocorr.R, EstimationMSEindep.R, EstimationMSEautocorr.R, ICindep.R, ICautocorr.R* and *pvalue*. Second step is to read the data file *dataExample.txt*. Third step is to run the application. The program creates several *txt* files in folder *C:/Areatimemodel*. The new files contain the output of the program in what follows the code in *Example.R* is listed.

```
#########################################################################
###
###                          Area-level time models
###                               SAMPLE project
###
### Author: Agustin Perez Martin
### File name: Example.R
### Updated: November 25th, 2009
###
#########################################################################

### Establishing the folder where data and routine files are located.
setwd("C:/Areatimemodel/")
```

```
### Call functions: H3area, REMLarea.indep, BETA.U.area.indep,
###                 mse.area.indep, Interval.indep,
###                 REMLarea.autocorr, BETA.U.area.autocorr,
###                 mse.area.autocorr, Interval.autocorr and p-value
source("H3.R")
source("REMLindep.R")
source("EstimationBETAindep.R")
source("EstimationMSEindep.R")
source("ICindep.R")
source("REMLautocorr.R")
source("EstimationBETAautocorr.R")
source("EstimationMSEautocorr.R")
source("ICautocorr.R")
source("pvalue.R")


### Reading data
data <- read.table(file = "dataExample.txt", header = T)

X <- as.matrix(data[,3:(ncol(data)-2)])
ydt <- data[,ncol(data)-1]
D <- length(unique(data[,1]))
md <- rep(length(unique(data[,2])), D)
sigma2edt <- data[,ncol(data)]


### Calculating H3 and REML variance estimates
sigma.0 <- H3area(X, ydt, D, md, sigma2edt)

### Model with independent time effects
### Arguments: (X, ydt, D, md, sigma2edt, sigma.0, MAXITER = 500)
fit0 <- REMLarea.indep(X, ydt, D, md, sigma2edt, sigma.0=sigma.0)
sigmau.hat <- fit0[[1]]
if(sigmau.hat<0) {
    write.table(data.frame(sigmau.hat),
    file="VAR.NEGATIVE.indep.txt", append=TRUE)
    sigmau.hat <- 0
}
### Arguments: (X, ydt, D, md, sigma2edt, sigmau)
```

```
beta.u.hat <- BETA.U.area.indep(X, ydt, D, md, sigma2edt, sigmau.hat)
beta.hat0 <- beta.u.hat[1:ncol(X),]
Int0 <- Interval.indep(fit0, 0.90)
                beta.hat0-Int0[[2]]
                beta.hat0+Int0[[2]]
(pvalue0 <- pvalue(beta.hat0, fit0))
                pvalue0>0.1
udt.hat0 <- beta.u.hat[-(1:ncol(X)),]

### EBLUP of the population parameter
mudt.hat.0 <- as.vector(X%*%beta.hat0 + udt.hat0)
sqrt.mse.0 <- sqrt(mse.area.indep(X, D,md,sigma2edt,sigmau.hat,
fit0[[2]]))
residuals.0 <- ydt-mudt.hat.0
Henderson3 <- sigma.0

#########################################################################

### Reading data
data <- read.table(file = "dataExample.txt", header = T)

X <- as.matrix(data[,3:(ncol(data)-2)])
ydt <- data[,ncol(data)-1]
D <- length(unique(data[,1]))
md <- rep(length(unique(data[,2])), D)
sigma2edt <- data[,ncol(data)]


### Calculating H3 and REML variance estimates
sigma.0 <- H3area(X, ydt, D, md, sigma2edt)


### Model with time correlated effects
### Arguments: (X, ydt, D, md, sigma2edt, sigma.0, MAXITER = 500)
fit1 <- REMLarea.autocorr(X, ydt, D, md, sigma2edt, sigma.0=sigma.0)
sigmau.hat <- fit1[[1]][1]
rho.hat <- fit1[[1]][2]
if(sigmau.hat<0) {
    write.table(data.frame(sigmau.hat),
    file="VAR.NEGATIVE.autocorr.txt", append=TRUE, col.names=FALSE)
```

```
    sigmau.hat <- 0
}
if(rho.hat < -1 || rho.hat > 1 ) {
    write.table(data.frame(rho.hat),
    file="COEFFICIENT OF CORRELATION OUT OF RANGE.txt",
    append=TRUE, col.names=FALSE)
    if(rho.hat < -1) {
        rho.hat <- -0.8
    }
    else rho.hat <- 0.8
}

### Arguments: (X, ydt, D, md, sigma2edt, sigmau)
beta.u.hat <- BETA.U.area.autocorr(X, ydt, D, md, sigma2edt,
 sigmau.hat, rho.hat)
beta.hat1 <- beta.u.hat[1:ncol(X),]
Int1 <- Interval.autocorr(fit1, 0.90)
                beta.hat1-Int1[[3]]
                beta.hat1+Int1[[3]]
(pvalue1 <- pvalue(beta.hat1, fit1))
                pvalue1>0.1
udt.hat1 <- beta.u.hat[-(1:ncol(X)),]

### EBLUP of the population parameter
mudt.hat.1 <- as.vector(X%*%beta.hat1 + udt.hat1)
sqrt.mse.1 <- sqrt(mse.area.autocorr(X, D, md, sigma2edt, sigmau.hat,
 rho.hat, fit1[[2]]))
residuals.1 <- ydt-mudt.hat.1



### Create .txt files in the folder that contains for
### the resulting output
write.table(data.frame(data[,1:2], Direct=ydt, EBLUP.0=mudt.hat.0,
EBLUP.1 = mudt.hat.1, sqrt.mse.direct=sqrt(sigma2edt),sqrt.mse.0,
sqrt.mse.1),
file="EBLUP_Example.txt",
row.names=FALSE, sep="\t")

write.table(data.frame(names(data)[3:(ncol(data)-2)], beta.hat0,
```

```
Std.error.beta.hat0=Int0[[2]], beta.hat1,
Std.error.beta.hat1=Int1[[3]]),
file="beta_Example.txt",
row.names=FALSE, sep="\t")

write.table(data.frame(data[,1:2], udt.hat0, udt.hat1),
file="u_Example.txt",
row.names=FALSE, sep="\t")

write.table(data.frame(data[,1:2], residuals.0, residuals.1),
file="res_Example.txt",
row.names=FALSE, sep="\t")

write.table(data.frame(names(data)[3:(ncol(data)-2)],
beta.hat0, pvalue0,
beta.hat1, pvalue1),
file="pvalue_Example.txt",
row.names=FALSE, sep="\t")

write.table(data.frame(Henderson3),
file="H3_Example.txt",
row.names=FALSE, sep="\t")

rm(list=ls(all=TRUE))
```

3.3.3 Outputs

Outputs of model 3.1 are labeled with "0" and outputs of model 3.3 with "1". The resulting outputs appear in the files *EBLUP Example.txt, beta Example.txt, u Example.txt, res Example.txt* and *H3 Example.txt* they are:

```
"EBLUP Example.txt" output:
"Domain" "Time"     "Direct"      "EBLUP.0"          "EBLUP.1"
11        1   0.07099622 0.0810870361440193 0.0885760701808574
11        2   0.072288837 0.0714371458327979 0.0762137925944318
11        3   0.082660682 0.052765034270926 0.0538737359735373
12        1   0.146031284 0.109117506760301 0.102837121921340
12        2   0.082499785 0.0971554778459163 0.106764441585287
12        3   0.078653447 0.0843765271001174 0.0885295192441997
21         1   0.301060532 0.274500665705542 0.266447830656762
21        2   0.241819175 0.251428302623275 0.255877074238007
```

```
21        3   0.237411987 0.258537796783850 0.260107687775899
22        1   0.305820376 0.283098865615287 0.28319090899208
22        2   0.263558025 0.257285589812057 0.265416656672475
22         3   0.285240927 0.276171257611288 0.278843232596284
31         1   0.179595403 0.184281701245608 0.178946303365822
31        2   0.171986758 0.178743608216205 0.170751458760549
31        3   0.160000387 0.176197813417279 0.176946498998694
32        1   0.179514009 0.179403322361097 0.175353458075361
32        2   0.183061881 0.18819267495888 0.187602783590969
32        3   0.188520565 0.196045641946532 0.195173080779298
41        1   0.353368205 0.328880532968824 0.313891050763258
41        2   0.249208975 0.287780626070448 0.304526684502343
41        3   0.317849199 0.308344519206265 0.297811346284671
42        1   0.335358187 0.317365511683804 0.327043967075657
42        2   0.322904936 0.297114134229847 0.305165260878979
42        3   0.353882106 0.333682212237881 0.343986387001928
51        1   0.208500731 0.202743936123168 0.217921552265458
51        2   0.329334406 0.272296339000462 0.283812864935672
51        3   0.335023497 0.209031824984797 0.218745541751390
52        1   0.204206531 0.260791713818310 0.26913502405639
52        2   0.389675949 0.348727254435842 0.342316109985649
52        3   0.452922541 0.311715605292666 0.304711309262514

 "sqrt.mse.direct"     "sqrt.mse.0"     "sqrt.mse.1"
0.0259817820789876 0.0222492368527262  0.02529723350447
0.0274049812990266 0.0233016417373966  0.0252815935554673
0.0339824366401234 0.0276158146322022  0.0291731520035163
0.0395438364350249 0.0281434204926076  0.0291793759628180
0.0304960817155254 0.0243856072652821  0.0261038788228251
0.0323074604387284 0.0257724969473051  0.028042744686898
0.0328642967367324 0.0248947518737948  0.0270197642584938
0.0309823982286717 0.0241802664409422  0.0257100094090509
0.0353964122475711 0.0259738102143437  0.0276420314540668
0.0339976322704979 0.0257435562716122  0.0273014024355766
0.0323380271507091 0.0251764189235333  0.0260644239979016
0.0372770707003649 0.0275971919369438  0.0287455620155905
0.0167115528901416 0.0155811019725347  0.0218917272635606
0.016373698421554 0.0153266667694242  0.0222328564459353
0.0169894673253754 0.0158295168167019  0.0219648135137690
0.0157542057876619 0.0149352292491623  0.0217904387722267
```

```
0.0165109357699677 0.0154673378691320  0.0222683939673919
0.0180734888718255 0.0166336789434665  0.0221892735173991
0.0370142134861731 0.0278643356937977  0.0285147125903474
0.0325541548807522 0.0263980146010989  0.0268870621864523
0.0353399632144687 0.0273638999999272  0.0282672691197796
0.0336586541620428 0.0260350821153883  0.0277529991368032
0.0347997270104235 0.0258673070510146  0.0262767739803480
0.0373891294362412 0.0277340978267927  0.0290881039736661
0.0501121442367017 0.0301782165013877  0.0317869641656460
0.0637050233498113 0.0314629079368649  0.0317167152903283
0.0694504859594229 0.0314451804811292  0.0329894046030183
0.04939436202645 0.0328545769793037  0.0344369451171986
0.0660453253455534 0.0342088762681202  0.0338925210978683
0.0739934726850957 0.032842871228608   0.0336642969125901
```

"beta Example.txt" output:

```
"names.data..3..ncol.data....2.."
          "beta.hat0"   "Std.error.beta.hat0"
"ones" 1.10551525071017 0.289971666456405
"X1" -0.669666738329566 0.595923529114958
"X2" -3.87902568140871 0.666469527559167

         "beta.hat1"     "Std.error.beta.hat1"
"ones"  0.996423126478698 0.344512091727891
"X1"    -0.488073608903706 0.698080746499988
"X2"    -3.63831553531472 0.843460606017086
```

"u Example.txt" output:

```
"Domain" "Time"     "udt.hat0"             "udt.hat1"
11        1 -0.0151417833538597    -0.0192488034769234
11        2 0.00114871351187552    -0.0066681353538353
11        3 0.0262233120003018    0.0092309435331532
12         1 0.0239121976242283    0.00504305226140131
12        2 -0.0159627192542945    -0.0147500568838680
12        3 -0.00555408840689845   -0.0156781552255283
21        1 0.0249095455482918    0.0141319291469034
21        2 -0.0101401062182701    -0.00535766612788023
21        3 -0.0170798275905686    -0.0138959417024322
```

```
22        1 0.0199126224457154    0.0228644281268613
22        2 0.00607571993117639    0.0169270187938618
22        3 0.00661144680352257    0.0144974596439298
31        1 -0.0169974947363216    -0.0261441556468742
31        2 -0.0255293245181901    -0.0389541223494046
31        3 -0.0568427521514347    -0.0602471372373828
32        1 0.000451742320188551   -0.00563446795630265
32        2 -0.0190646925190417    -0.0212933730780450
32        3 -0.0233354166433626    -0.0263804646846378
41        1 0.0181050219383043    4.90153569384333e-06
41        2 -0.0368675251357491    -0.0228466003977432
41        3 0.00770892659885395    -0.00713044634013867
42        1 0.0160875850499344    0.0234862376843940
42        2 0.0215725644741984    0.0275172167976236
42        3 0.0146368285366828    0.0233409064848382
51        1 0.00232211321187295    0.0178768815245654
51        2 0.0142365922610651    0.0283510210380592
51        3 0.0264593924546214    0.0334430237828832
52        1 -0.0234928966839586    -0.00613231011438557
52        2 0.00950921134206837    0.0124023572829343
52        3 0.0261250911590495    0.0238630203744320

"res Example.txt" output:
"Domain" "Time"     "residuals.0"          "residuals.1"
11        1 -0.0100908161440193   -0.0175798501808574
11        2 0.000851691167202129  -0.00392495559443180
11        3 0.029895647729074   0.0287869460264627
12        1 0.0369137772396989   0.0431941620786601
12        2 -0.0146556928459163   -0.0242646565852869
12        3 -0.0057230801001174   -0.0098760722441997
21        1 0.0265598662944575   0.0346127013432377
21        2 -0.00960912762327454  -0.0140578992380065
21        3 -0.0211258097838496   -0.0226957007758992
22        1 0.0227215103847131   0.0226294670079202
22        2 0.00627243518794296   -0.00185863167247524
22        3 0.0090696693887124   0.00639769440371557
31        1 -0.00468629824560846  0.000649099634178196
31        2 -0.00675685021620515  0.00123529923945057
31        3 -0.0161974264172793   -0.0169461119986940
32        1 0.000110686638902596  0.00416055092463880
```

```
32        2 -0.00513079395887975  -0.00454090259096904
32        3 -0.00752507694653232  -0.00665251577929787
41        1 0.0244876720311756   0.0394771542367417
41        2 -0.0385716510704477   -0.0553177095023425
41        3 0.00950467979373487   0.0200378527153294
42        1 0.0179926753161957   0.00831421992434317
42        2 0.0257908017701525   0.0177396751210214
42        3 0.020199893762119   0.00989571899807246
51        1 0.00575679487683178   -0.00942082126545776
51        2 0.0570380669995378   0.0455215410643282
51        3 0.125991672015203   0.116277955248610
52        1 -0.0565851828183103   -0.06492849305639
52        2 0.0409486945641577   0.0473598390143509
52        3 0.141206935707334   0.148211231737486


"pvalue Example.txt" output:
"names.data..3..ncol.data....2.."
          "beta.hat0"           "pvalue0"
"ones" 1.10551525071017 3.58748940379529e-10
"X1" -0.669666738329566 0.0645448111231833
"X2" -3.87902568140871 1.03371377940706e-21

          "beta.hat1"           "pvalor1"
"ones" 0.996423126478698 1.96135137530811e-06
"X1" -0.488073608903706 0.250133966757249
"X2" -3.63831553531472 1.29192448422908e-12

"H3 Example.txt" output:
0.00101519275290970
```

Chapter 4

M-quantile small area estimators of the mean

4.1 Methodology

A recently proposed approach to small area estimation is based on the use of M-quantile models (Chambers & Tzavidis, 2006). M-quantile regression provides a "quantile-like" generalization of regression based on influence functions (Breckling & Chambers, 1998). M-quantile models do not depend on strong distributional assumptions nor on a predefined hierarchical structure, and outlier robust inference is automatically performed when these models are fitted. The M-quantile of order q for the conditional density of y given $\mathbf{X}$ is defined as the solution $Q_q(x;\psi)$ of the estimating equation $\int \psi_q(y-Q)f(y|\mathbf{X})dy = 0$, where ψ denotes an influence function associated with the M-quantile. In a linear M-quantile regression model the q-th M-quantile $Q_q(x,\psi)$ of the conditional distribution of y given $\mathbf{X}$ is such that

$$Q_q(x;\psi) = \mathbf{X}\beta_\psi(q) \tag{4.1}$$

where $\psi_q(r_{iq\psi}) = 2\psi\{s^{-1}r_{iq\psi}\}\ \{qI(r_{jq\psi} > 0) + (1-q)I(r_{jq\psi} \leq 0)\}$ and s is a suitable robust estimate of scale, e.g. the MAD estimate $s = median\left|r_{jq\psi}\right|/0.6745$. A popular choice for the influence function is the Huber Proposal 2, $\psi(u) = uI(-c \leq u \leq c) + c\,\mathtt{sgn}(u)$. However, other influence functions are also possible. For specified q and continuous ψ, an estimate $\hat{\beta}_\psi(q)$ of $\beta_\psi(q)$ is obtained via iterative weighted least squares. Note that there is a different set of regression parameters for each q.

Let $\Omega_d = \{1,\ldots,N_d\}$ be the population of area d. Let $\mathbf{y}_d = (y_1,\ldots,y_{N_d})'$ denote the variable values for the N_d small area population elements. We consider a sample $s_d \subset \Omega_d$, of $n_d \leq N_d$ units, and we denote with $r_d = \Omega_d - s_d$ the set of non sampled units. For each population unit j, let $\mathbf{x}_j = (x_{1j},\ldots,x_{pj})$ denote a vector of p known auxiliary variables. The small area specific empirical distribution function of y for area d is

$$F_d = N_d^{-1}\Big[\sum_{j\in s_d} \mathrm{I}(y_j \leqslant t) + \sum_{j\in r_d} \mathrm{I}(y_j \leqslant t)\Big]. \tag{4.2}$$

The problem of estimating $F_d(t)$ given the sample data essentially reduces to predicting the values y_j for the non-sampled units in small area d. One straightforward way of achieving this is to simply

replace the unknown non-sample values of y (4.2) by their predicted values $\hat{y}_j$ under an appropriate model, leading to a plug-in estimator of (4.2) of the form

$$\hat{F}_d = N_d^{-1}\Big[\sum_{j\in s_d} \mathrm{I}(y_j \leqslant t) + \sum_{j\in r_d} \mathrm{I}(\hat{y}_j \leqslant t)\Big]. \tag{4.3}$$

An estimator of the mean $\overline{Y}_d$ of y in area d is then defined by the value of the mean functional defined by (4.3). This leads to the usual plug-in estimator of the mean,

$$\hat{\overline{Y}}_d = \int_{-\infty}^{\infty} t d\hat{F}_d(t) = N_d^{-1}\left(\sum_{j\in s_d} y_j + \sum_{j\in r_d} \hat{y}_j\right).$$

The predicted value of a non-sample unit j in area d corresponds to an estimate $\hat{\mu}_j$ of its expected value given that it is located in area d.

When the conditional M-quantiles are assumed to follow a linear model, with $\beta_\psi(q)$ a sufficiently smooth function of q, this suggests an estimator of the distribution function

$$\hat{F}_d^{MQ}(t) = N_d^{-1}\left\{\sum_{j\in s_d} I(y_j \leq t) + \sum_{j\in r_d} I(\mathbf{x}_j\hat{\beta}_\psi(\hat{\theta}_d) \leq t)\right\} \tag{4.4}$$

where $\mathbf{x}_j\beta_\psi(\theta_d)$ is used to predict the unobserved value y_j for population unit $j \in r_d$. When there are no sampled observations in area d then $\hat{\theta}_d = 0.5$.

Using the empirical distribution function and the linear M-quantile small area models one can defined an estimators of the small area mean

$$\hat{\overline{Y}}_d^{MQ}(t) = \int_{-\infty}^{\infty} t d\hat{F}_d^{MQ}(t) = N_d^{-1}\left\{\sum_{j\in s_d} y_j + \sum_{j\in r_d} \mathbf{x}_j\hat{\beta}_\psi(\hat{\theta}_d)\right\}. \tag{4.5}$$

Chambers & Tzavidis (2006) observed that the naive M-quantile mean estimator (4.5) can be biased. The distribution function estimator (4.3) underlying (4.4) is not consistent in general. Thus, when the non-sample predicted values in (4.3) are estimated expectations that converge in probability to the actual expected values, we see that

$$\sum_{j\in r_d} \mathrm{I}(\hat{y}_j \leqslant t) = \sum_{j\in r_d} \mathrm{I}(y_j - (y_j - \hat{y}_j) \leqslant t) = \sum_{j\in r_d} \mathrm{I}(y_j \leqslant t + \varepsilon_j) \neq \sum_{j\in r_d} \mathrm{I}(y_j \leqslant t),$$

where ε_j are the actual regression errors. If these errors are independently and identically distributed symmetrically about zero we expect that the summation on the left hand side above will closely approximate the summation on the right for values of t near the median of the non-sampled area d values of y but not anywhere else. More generally, for heteroskedastic and/or asymmetric errors this correspondence will typically occur elsewhere in the support of y, although one would expect that in most reasonable situations it will be "close" to the median of y. In other words, it is not advisable to use (4.3) to predict a quantile of the area d distribution of y other than the median.

By combining a smearing argument (Duan, 1983) with a model for the finite population distribution of y, Chambers & Dunstan (1986), hereafter referred to as CD, developed a model-consistent estimator

for a finite population distribution function. In the context of the small area distribution function (4.2), and assuming that the residuals are homoskedastic within the small area of interest, this is of the form

$$\hat{F}_d^{MQ/CD}(t) = N_d^{-1}\left\{\sum_{j\in s_d} I(y_j \le t) + \sum_{k\in r_d} n_d^{-1}\sum_{j\in s_d} I(\hat{y}_k + (y_j - \hat{y}_j) \le t)\right\}. \tag{4.6}$$

It can be shown that under the CD estimator of the small area distribution function the mean functional defined by (4.6) takes the value

$$\hat{\bar{Y}}_d^{MQ/CD} = \int_{-\infty}^{\infty} t d\hat{F}_d^{MQ/CD}(t) = N_d^{-1}\left\{\sum_{j\in s_d} y_j + \sum_{j\in r_d} \hat{y}_j + (f_d^{-1} - 1)\sum_{j\in s_d}(y_j - \hat{y}_j)\right\} \tag{4.7}$$

where $f_d = n_d N_d^{-1}$ is the sampling fraction in area d, $\hat{y}_j = \mathbf{x}_j\hat{\beta}_\psi(\hat{\theta}_d)$, where $\hat{y}_j$ can be obtained either under the linear or the nonparametric M-quantile small area models. We refer to (4.7) as the bias adjusted M-quantile mean predictor. Due to the bias correction in (4.7), this predictor will have higher variability than (4.5) and so it should only be used when (4.4) are expected to have substantial bias, e.g. when there are large outlying data points.

4.1.1 Estimation of the MSE

A robust mean squared error estimation method for the naive M-quantile estimator (4.5) was described in Chambers & Tzavidis (2006). To start, we note that since an iteratively reweighted least squares algorithm is used to calculate the M-quantile regression fit at $\hat{\theta}_d$, we have

$$\hat{\beta}_\psi(\hat{\theta}_d) = (\mathbf{X}_s'\mathbf{W}_{s_d}\mathbf{X}_s)^{-1}\mathbf{X}_s'\mathbf{W}_{s_d}\mathbf{y}_s$$

where $\mathbf{X}_s$ and $\mathbf{y}_s$ denote the matrix of sample x values and the vector of sample y values respectively, and $\mathbf{W}_{s_d}$ denotes the diagonal weight matrix of order n that defines the estimator of the M-quantile regression coefficient with $q = \hat{\theta}_d$. It immediately follows that (4.5) can be written

$$\hat{\bar{Y}}_d^{MQ} = \mathbf{w}_{s_d}'\mathbf{y}_s, \tag{4.8}$$

with weights

$$\mathbf{w}_{s_d}^{MQ} = N_d^{-1}\left[\Delta_n^{(d)} + \mathbf{W}_s(\hat{\theta}_d)\mathbf{X}_s\{\mathbf{X}_s'\mathbf{W}_s(\hat{\theta}_d)\mathbf{X}_s\}^{-1}\mathbf{X}_r\Delta_{N-n}^{(d)}\right]. \tag{4.9}$$

Here $\mathbf{W}_s(\hat{\theta}_d)$ is the diagonal matrix of final weights used in the IRLS algorithm. It also follows that (4.7) can be written

$$\hat{\bar{Y}}_d^{MQ/CD} = \mathbf{w}_{s_d}'\mathbf{y}_s$$

with weights

$$\mathbf{w}_{s_d} = (w_{jd}) = n_d^{-1}\Delta_{s_d} + (1 - N_d^{-1}n_d)\mathbf{W}_d\mathbf{X}_s(\mathbf{X}_s'\mathbf{W}_d\mathbf{X}_s)^{-1}\{\bar{\mathbf{x}}_{r_d} - \bar{\mathbf{x}}_{s_d}\}. \tag{4.10}$$

with Δ_{s_d} denoting the n-vector that "picks out" the sample units from area d. Given the linear representation of the M-quantile predictors, methods of robust mean squared error estimation for linear predictors

of population quantities (Royall & Cumberland, 1978) can be used. In particular, the prediction variance of $\hat{\bar{Y}}_d^{MQ/CD}$ is estimated by

$$v(\hat{\bar{Y}}_d^{MQ/CD}) = \frac{1}{N_d^2}\sum_{g=1}^{G}\sum_{j\in s_g}\lambda_{jdg}\left\{y_j - \mathbf{x}_j\hat{\beta}_\psi(\hat{\theta}_g)\right\}^2, \tag{4.11}$$

where $\lambda_{jdg} = \left\{(w_{jd}-1)^2 + (n_d-1)^{-1}(N_d-n_d)\right\}I(g=d) + w_{jg}^2 I(g\neq d)$. This prediction variance estimator implicitly assumes a model where the regression of y on x varies between areas, and that this variation is consistently estimated by the fit of the M-quantile regression model in each area.

4.2 The Software: description of R functions

In this document we present the function mq.sae that is designed for producing small area mean estimates under the M-quantile small area model proposed by Chambers & Tzavidis (2006) and Tzavidis, Marchetti, & Chambers (2010). The computation of small area estimates and of the corresponding Mean Squared Error (MSE) using the R software is illustrated with a simulated data set. Full R codes are included in Appendix 4.

4.2.1 Required R packages

Before using mq.sae install the following packages

- MASS

4.2.2 mq.sae

```
>mq.sae(y,x,regioncode.s,m,p,x.outs,regioncode.r,tol.value,
 maxit.value,k.value)
```

- *x*: a $n \times p$ matrix of auxiliary variables which also has include a vector of ones for the intercept term
- *y*: the (numeric) response vector for sampled units
- *regioncode.s*: area code for sampled units
- *m*: the number of small areas
- *p*: size of x +1 (including the intercept)
- *x*.outs: covariate information for out of sample units
- *regioncode.r*: area code for out of sample units
- *tol.value*: Convergence tolerance limit for the M-quantile model. Default to 0.0001

- *maxit.value*: maximum number of iterations for the iterative weighted least squares. Default to 100
- *k.value*: tuning constant used with the Huber proposal 2 scale estimation. Default to 1.345

The function returns small area estimates of the mean under the M-quantile model as well as the corresponding MSE estimates.

- *mq.cd*: Estimates of small area means using the M-quantile Chambers-Dunstan estimator (Tzavidis, Marchetti, & Chambers, 2010)
- *mq.naive*: Estimates of small area means using the M-quantile naive estimator (Chambers & Tzavidis, 2006)
- *mse.cd*: MSE estimates for the M-quantile CD small area means (Tzavidis, Marchetti, & Chambers, 2010)
- *mse.naive*: MSE estimates for the M-quantile naive small area means (Chambers & Tzavidis, 2006)
- *code.area*: the codes of the small areas

4.3 Examples of usage of R functions

4.3.1 Data generation

For illustrating the use of the mq.sae function, data are generated under the following location-shift model

$$y_{ij} = 100 + 2x_{ij} + g_j + e_{ij}, i = 1,...,5, j = 1,...,40,$$

where the values of $x \sim \ LogNormal(\mathrm{N}, log(4.5) - 0.5, 0.5)$ and the error terms are generated from $g_j \sim N(0,9)$ and $e_{ij} \sim N(0,36)$.

```
> # MQ-EBLUP
> source("c:\\MQ_sae.R")
> library(pps)
> sigmasq.u=3
> sigmasq=6
> NoSim<-1
> m=40
> ni=rep(5,m)
> Ni=rep(100,m)
> N=sum(Ni)
> n=sum(ni)
```

```
> set.seed(1973)
> u=rnorm(m,0,sqrt(sigmasq.u))
> u=rep(u,each=100)
> e <- rnorm(N, 0, sqrt(sigmasq))
> gr=rep(1:40,each=100)
> ar=unique(gr)
> uno=matrix(c(rlnorm(N,log(4.5)-0.5,0.5)),nrow=N,ncol=1
> y=100+5*uno+u+e
> pop.matrix<-cbind(y,uno,gr)
> pop<-as.data.frame(pop.matrix)
> names(pop)<-c("y","x","area")
> # Drawing a sample
> s<-stratsrs(pop$area,ni)
> x.lme=pop[s,]$x
> y.lme=pop[s,]$y
> regioncode.lme=pop[s,]$area
> pop.r<-pop[-s,]
```

4.3.2 Example of R code for running function mq.sae

```
tmp<-mq.sae(y=y.lme,x=x.lme,regioncode.s=regioncode.lme,m=40,
p=2,x.outs=pop.r[,2], regioncode.r=pop.r[,3],tol.value=0.0001,
maxit.value=100,k.value=1.345)
```

4.3.3 Output of function mq.sae

```
> tmp

mq.cd
[1] 115.7275 117.9384 115.3374 115.5339 116.3331 ...

mq.naive
 [1] 115.9003 117.6583 115.0192 115.6947 116.0871 ...

mse.cd
 [1] 0.55237498 0.80473242 1.54140859 0.75538562 2.13604316 ...

mse.naive
[1] 0.09710564 0.02790977 0.16425263 0.05226719 0.12559878...

code.area
```

```
[1]  1  2  3  4  5...
```

Chapter 5

M-quantile Geographically Weighted Regression

5.1 Methodology

Typically, random effects models assume independence of the random area effects. This independence assumption is also implicit in M-quantile small area models. In economic applications, however, observations that are spatially close may be more related than observations that are further apart. This spatial correlation can be accounted for by extending the random effects model to allow for spatially correlated area effects using, for example, a Simultaneous Autoregressive (SAR) model (Petrucci & Salvati, 2006; Pratesi & Salvati, 2008, 2009). An alternative approach to incorporate the spatial information in the regression model is by assuming that the regression coefficients vary spatially across the geography of interest. Geographically Weighted Regression (GWR) (Brunsdon, Fotheringham & Charlton, 1996) extends the traditional regression model by allowing local rather than global parameters to be estimated. In a recent paper Salvati, Tzavidis, Pratesi & Chambers (2008) proposed an M-quantile GWR small area model. The authors proposed an extension to the GWR model, the M-quantile GWR model, i.e. a locally robust model for the M-quantiles of the conditional distribution of the outcome variable given the covariates. Here we report a brief description of the M-quantile GWR model.

The GWR model is a model for the conditional expectation of $\mathbf{y}$ given $\mathbf{X}$ at location u. This is easily generalised to a model for the M-quantile of order q of the conditional distribution of $\mathbf{y}$ given $\mathbf{X}$ at u. That is, we write

$$Q_q(\mathbf{X};\psi,u) = \mathbf{X}\beta_\psi(u;q) \tag{5.1}$$

where $\beta_\psi(u;q)$ varies with u as well as with q. That is, model (5.1) allows the entire conditional distribution (not just the mean) of y given $\mathbf{X}$ to vary from location to location. The parameter $\beta_\psi(u;q)$ in (5.1) can be estimated by solving normal equations by an iteratively re-weighted least squares algorithm that combines the iteratively re-weighted least squares algorithm used to fit a "spatially stationary" M-quantile model and the weighted least squares algorithm used to fit a GWR model.

The model (5.1) was then used to define a predictor of the small area characteristic of interest that

accounts for spatial structure of the data. The M-quantile GWR small area model integrates the concepts of robust small area estimation and borrowing strength over space within a unified modeling framework. Extending further the M-quantile GWR for poverty measures will enable the comparison of alternative robust models for borrowing strength over space in small area estimation and will significantly improve the collection of small area estimation tools.

An estimator of a first order approximation to the mean squared error of the M-quantile GWR predictor can be computed following methods of robust mean squared error estimation for linear predictors of population quantities (Royall & Cumberland, 1978). More details are available in Salvati, Tzavidis, Pratesi & Chambers (2008).

5.2 The Software: description of R functions

In this document we explain how to produce Small Area estimates using MQGWR and we present an example. The model and the estimator are described in Salvati, Tzavidis, Pratesi & Chambers (2008) and the computation of small area estimates using the R software is illustrated with a simulated data set. Full R codes are included in Appendix 5.

5.2.1 Required R packages

Before using mqgwr.sae install the following packages

- MASS
- nlme
- sp
- spqwr

5.2.2 mqgwr.sae

```
> mqgwr.sae(x,y,m,area,lon,lat,x.r,area.r,lon.r,lat.r,method,
  k.value=1.345,mqgwrweight)
```

- x: a $n \times p$ matrix of auxiliary variables with the intercept term for sampled units
- y: the (numeric) response vector for sampled units
- m: the number of small areas
- *area*: a vector of small area codes for sampled units
- *lon* is a vector of longitude of points representing the spatial positions of the sampled observations

- *lat*: a vector of latitude of points representing the spatial positions of the sampled observations
- *x.r*: a $(N-n) \times p$ matrix of auxiliary variables with the intercept term for out of sample units
- *area.r*: a $N-n$ vector of small area codes for out of sample units
- *lon.r*: a $N-n$ vector of longitude of points representing the spatial positions of the out of sample observations
- *lat.r*: a $N-n$ vector of latitude of points representing the spatial positions of the out of sample observations
- *k.value*: tuning constant used for Huber proposal 2 scale estimation. Default to 1.345
- *method*: a character string. If 'mqgwr' the M-quantile GWR model is used to fit the M-quantile surface. If 'mqgwr-li' the MQGWR-LI (Local Intercepts) is used. Defaults to 'mqgwr'
- *mqgwrweight*: geographical weighting function: gwr.gauss() if it is TRUE or gwr.bisquare() if it is FALSE. Defaults to TRUE

Return the small area estimates of the mean and of its MSE:

- *Area.code.in*: the codes of the sampled areas
- *Area.code.out*: the codes of the out of sample areas
- *Est.Mean.in*: the estimates of the small area mean for sampled areas
- *Est.Mean.out*: the estimates of the small area mean for out of sample areas
- *Est.mse.in*: the estimates of the MSE for sampled areas

5.3 Examples of usage of R functions

5.3.1 Example data

The data used in this example is a part of the synthetic population generated by Salvati et al. (2008). This pseudo-population was obtained by using the data from the U.S. Environmental Protection Agency's Environmental Monitoring and Assessment Program (EMAP) Northeast lakes survey. Between 1991 and 1995, researchers from the U.S. Environmental Protection Agency (EPA) conducted an environmental health study of the lakes in the north-eastern states of the U.S.A. For this study, a sample of 334 lakes (or more accurately, lake locations) was selected from the population of $21,026$ lakes in these states using a random systematic design. The lakes making up this population are grouped into 113 8-digit Hydrologic Unit Codes (HUCs). We defined HUCs as the small areas of interest, with lakes grouped within HUCs. The variable of interest was Acid Neutralizing Capacity (ANC), an indicator of the acidification risk of water bodies. In addition to ANC values and associated survey weights for the sampled locations, the

EMAP data set also contained the elevation and geographical coordinates of the centroid of each lake in the target area. Given the 21,026 lake locations, a synthetic population of ANC individual values were non parametrically simulated using a nearest-neighbour imputation algorithm that retained the spatial structure of the observed ANC values in the EMAP sample data. Details on the data generation are in Salvati, Tzavidis, Pratesi & Chambers (2008). In this example we consider the population for 10 HUCs, in which are grouped 1737 lakes. Moreover we don't draw any units for the last HUC. In other words, we have one out of sample area.

```
>source("mqgwr.R")
>library(sampling)
>data.lake=read.table("PopSynthIn.txt",header=TRUE,dec=",")
>s=strata(data.lake,"HUC",size=c(rep(5,9),1))
>s=s[-46,]
>sample=getdata(data.lake,s)
>lake.r=data.lake[-(s$ID_unit),]
```

A total of 45 lakes are drawn from the population. The data frame *sample* contains the observed lakes, whereas the data frame *lake.r* represents the out of sample units.

5.3.2 Example of R code for running function mqgwr.sae

```
>SaeEst=mqgwr.sae(x=sample$x,y=sample$y,m=10,area=sample$HUC,
lon=sample$lon,
lat=sample$lat,x.r=lake.r$x,area.r=lake.r$HUC,lon.r=lake.r$lon,
lat.r=lake.r$lat,k.value=1.345, method="mqgwr", mqgwrweight=TRUE)
```

5.3.3 Output of function mqgwr.sae

```
mqgwr-SAE estimates

data:  EMAP
SaeEst
$Area.code.in
[1] 1010001 1010002 1010003 1010004 1010005 1020001 1020002 1020003
1020004

$Area.code.out
[1] 1020005

$Est.Mean.in
```

```
[1] 345.1899 509.2659 387.1552 398.6101 548.1446 253.3449 433.2629
278.2129
356.8936

$Est.Mean.out
[1] 280.9551

$Est.mse.in
[1]  5856.302  2546.586  3523.715  3389.980  3953.433  6490.745
3697.981 9095.215 36428.013
```

Chapter 6

M-quantile CD estimators of the CDF

6.1 Methodology

Estimating the quantiles of a distribution function in addition to conventional outliers sensitive measures of central tendency, such as averages, provides a more complete picture of the study variable. This is the case particularly when handling highly skewed variables, such as income or consumption where we expect the median to be different from the mean. In this paper we focus on estimating the quantiles of the small area distribution function using the M-quantile model proposed by Chambers & Tzavidis (2006) and the estimator of the finite population distribution function proposed by Chambers & Dunstan (1986).

In what follows we assume that a vector of p auxiliary variable $\mathbf{x}_i$ is known for each population unit i in small area j and that values of the variable of interest y are available from a random sample, s, that include units from all the small areas of interest. We denote the population size, sample size, sampled part of the population and non sampled part of the population in area j respectively by N_j, n_j, s_j and r_j. We assume that the sum over the areas of N_j and n_j is equal to N and n respectively.

Under the Chambers & Tzavidis (2006) approach the M-quantile small area model for unit i in area j is defined as follows

$$y_{ij} = \mathbf{x}'_{ij}\beta_\psi(\theta_j) + \varepsilon_{ij}, \tag{6.1}$$

where ε_{ij} denotes the regression error for the unit i in area j,and $\hat{\theta}_j$ denotes the area specific M-quantile coefficients that describe the intra-area variation. Using the Chambers-Dunstan estimator of the distribution function and the linear M-quantile small area model, an estimator of the small area distribution function is

$$\hat{F}_j^{CD}(t) = N_j^{-1}\left[\sum_{i\in s_j} I(y_{ij} \le t) + n_j^{-1}\sum_{k\in r_j}\sum_{i\in s_j} I(\mathbf{x}'_{kj}\hat{\beta}_\psi(\hat{\theta}_j) + \hat{e}_{ij} \le t)\right], \tag{6.2}$$

where $\hat{\beta}_\psi$ and $\hat{\theta}_j$ are obtained following Chambers & Tzavidis (2006) and $\hat{e}_{ij}$ are the residuals under model (6.1). A point estimate of the small area quantile τ is then obtained by numerically solving

$$\int_{-\infty}^{\hat{q}(j;\tau)} d\hat{F}_j^{CD}(t) = \tau. \tag{6.3}$$

6.1.1 Bootstrap MSE Estimation for Small Area Quantiles

Analytic estimation of the MSE of the estimates of small area quantiles is complex. In this section we describe a semi-parametric bootstrap approach for estimating the MSE of small area quantiles. Our approach is based on an extension of the bootstrap procedure proposed by Lombardia, M., Gonzalez-Manteiga, W., and Prada-Sanchez, J. (2003).

Having estimated model (6.1), we compute the estimated M-quantile small area model residuals,

$$\hat{e}_{ij} = y_{ij} - \mathbf{x}'_{ij}\hat{\boldsymbol{\beta}}_{\psi}(\hat{\theta}_j).$$

A bootstrap population $\Omega^* = \{y^*_{ij}, \mathbf{x}_{ij}\}$, $i \in \Omega$, $j = 1, \ldots, d$ can be generated with

$$y^*_{ij} = \mathbf{x}'_{ij}\hat{\boldsymbol{\beta}}_{\psi}(\hat{\theta}_j) + e^*_{ij},$$

where the bootstrap residuals e^*_{ij} are obtained by sampling from an estimator of the distribution function $\hat{G}(t)$ of the model residuals $\hat{e}_{ij}$. For defining $\hat{G}(t)$ we consider two approaches: (1) sampling from the empirical distribution function of the model residuals and (2) sampling from a smoothed distribution function of these residuals. For each abovementioned case sampling of the residuals can be done in two ways, (1) by sampling from the distribution of all residuals without conditioning on the small area. We refer to this as the unconditional approach; and (2) by sampling from the conditional distribution of residuals within small area j. We refer to this as the conditional approach. Hence, in total there are four possible ways of defining e^*_{ij}.

The steps of our bootstrap procedure are as follows. Starting from sample s, selected from a finite population Ω without replacement, we generate B bootstrap populations, Ω^{*b}, using one of the four above mentioned methods for estimating the distribution of the residuals. From each bootstrap population, Ω^{*b}, we select L samples using simple random sampling within the small areas and without replacement in a way such that $n^*_j = n_j$. Bootstrap estimators of the bias and variance of our predictor of the distribution function in area j are defined respectively by

$$\widehat{\text{Bias}}_j = B^{-1}L^{-1}\sum_{b=1}^{B}\sum_{l=1}^{L}\left(\hat{F}_j^{*bl,CD}(t) - F_{N,j}^{*b}(t)\right),$$

$$\widehat{\text{Var}}_j = B^{-1}L^{-1}\sum_{b=1}^{B}\sum_{l=1}^{L}\left(\hat{F}_j^{*bl,CD}(t) - \bar{\hat{F}}_j^{*bl,CD}(t)\right)^2,$$

where $F_{N,j}^{*b}(t)$ is the distribution function of the bth bootstrap population, $\hat{F}_j^{*bl,CD}(t)$ is the Chambers-Dunstan estimator of $F_{N,j}^{*b}(t)$ computed from the lth sample of the bth bootstrap population and $\bar{\hat{F}}_j^{*bl,CD}(t) = L^{-1}\sum_{l=1}^{L}\hat{F}_j^{*bl,CD}(t)$. The bootstrap Mean Squared Error estimator of the estimated small area quantile is then defined as

$$\widehat{MSE}\left(\hat{F}_j^{CD}(t)\right) = \widehat{\text{Var}}_j + \widehat{\text{Bias}}_j^2.$$

Finally, using a normal approximation we can further compute approximate confidence intervals for the estimated small area quantiles.

More details are given in Tzavidis, Marchetti, & Chambers (2010).

6.2 The Software: description of R functions

In this document we present the function mq.sae.quant that is designed for producing estimates of the quantiles of the small area distribution function using the M-quantile small area model and the Chambers-Dunstan estimator of the cumulative distribution function. The function produces point estimates of small area quantiles as well as nonparametric bootstrap estimates of Mean Squared Error (MSE) using the methodology in Tzavidis, Marchetti, & Chambers (2010). The computation of the small area estimates and of the corresponding MSEs using the R software is illustrated with a simulated data set. Full R codes are included in Appendix 6.

6.2.1 Required R packages

Before using mq.sae.quant install the following packages

- MASS
- np

6.2.2 mq.sae.quant

```
>mq.sae.quant(qgrid,y,x.design,X.pop,regioncode,regioncodepop,adjseed,
myMSE,B=,R=,method="")
```

- *qgrid*: a vector of quantiles to be estimated for each small area
- *y*: the response variable for sampled units
- *x.design*: the sample design matrix to be used for fitting the M-quantile model
- $X.pop$: the design matrix for the entire population
- *regioncode*: vector of area codes
- *regioncodepop*: vector of area codes for the population data
- *adjseed*: parameter for the starting point for the numerical solution of the quantile integral. Default to $\max(0.15, mean(y)/500)$
- *myMSE*: If TRUE the function computes bootstrap MSE of estimates of small area quantiles. If FALSE only point estimates of small area quantiles are provided.

- *B*: number of bootstrap populations to be generated. Default to 1
- *R*: number of bootstrap samples selected from each bootstrap population. Default to 400
- *method*: method defines the type of residuals used for generating the bootstrap population: 'su' (smooth unconditional),'eu' (emprirical unconditional),'sc' (smooth conditional),'ec' (empirical uncoditional). Default is set to 'eu'

The function returns small area estimates of the quantiles of interest as well as the corresponding estimates of the root MSE.

- *quantiles*: Estimates of small area quantiles for each small area. The rows represent the quantiles and the columns the small areas
- *rmse*: Estimates of root mean squared error for each small area quantile estimate in each small area. The rows represent the quantiles and the columns the small areas

6.3 Examples of usage of R functions

6.3.1 Data generation

```
> source("mq.sae.quant.txt")

> #GENERATE DATA
> areas<-10
> n.area<-30
> n.area.pop<-300
> X.pop<-rnorm(n.area.pop*areas,10,2)
> X.pop<-cbind(rep(1,n.area.pop*areas),X.pop)
> x<-rnorm(n.area*areas,10,2)
> x.design<-cbind(rep(1,n.area*areas),x)
> y<-5+2*x+rep(rnorm(areas,1,2),n.area)+rnorm(n.area*areas,0,4)

> ####VARIABLES DESCRIPTION
> #areas <- the number of small area
> #n.area <- the sample size of the areas
> #y <- the target variable
> #x <- auxiliary variable
> #X.pop <- auxiliary variable for the population

> ####SET VARIABLES FOR THE ESTIMATION PROCEDURE
> ####(and for optional simulation)
```

```
> qgrid<-c(0.1,0.25,0.50,0.75,0.9) #Set of quantiles to be estimated
> regioncode<-rep(1:10,each=30)
> regioncodepop<-rep(1:10,each=300)
```

6.3.2 Example of R code for running function mq.sae.quant

```
>cdf.cd.est<-mq.sae.quant(qgrid,y,x.design,X.pop,regioncode,
regioncodepop,myMSE=TRUE,B=1,R=40,method="eu")
```

6.3.3 Output of function mq.sae.quant

```
> cdf.cd.est

$quantiles
       [,1]    [,2]    [,3]    [,4]    [,5]    [,6]    [,7]    [,8]
[1,] 18.0234 19.2805 18.1535 18.2715 16.9831 18.8019 19.2367 16.6958...
[2,] 21.6293 21.9110 21.8436 21.3836 19.8507 21.7745 21.5290 21.4206...
[3,] 25.8823 25.7067 25.0682 25.3097 24.5161 25.5556 26.0643 26.5182...
[4,] 32.7068 29.8444 28.6127 29.5115 27.2132 27.6156 30.6451 29.4411...
[5,] 35.1152 33.0718 31.7438 33.0362 30.0770 32.8009 34.2575 33.4954...

$rmse
        [,1]      [,2]      [,3]      [,4]      [,5]
[1,] 1.1595381 1.3712362 1.1664052 1.2772312 1.0793872...
[2,] 1.0568906 0.8799666 1.0041534 1.1280502 1.0285473...
[3,] 0.8476319 0.6893870 1.2079049 1.0794124 0.8962513...
[4,] 1.1068735 0.8294108 0.7490893 0.7651702 1.0633727...
[5,] 0.9891367 1.0894626 0.9488074 1.1451405 1.1892031...
```

Chapter 7

Appendix 1: R code for Fay-Herriot model

7.1 R code of fitFH

The R code of the function `fitFH` is listed bellow.

```
#################################################################
### ### This function fits a Fay-Herriot model ### Fitting method
can be chosen between REML and FH methods ### ### Work for
European project SAMPLE ### ### Author: Isabel Molina Peralta ###
File name: Fitting_FHModel.R ### Updated: Example ###
#################################################################

fitFH<-function(X,y,Dvec,method="REML",MAXITER=500) {

  m<-length(y) # Sample size or number of areas
  p<-dim(X)[2] # Num. of X columns of num. of auxiliary variables
  Xt<-t(X)

  # Fisher-scoring algorithm for REML estimator of variance A starts

  if (method=="REML") {

    # Initial value of variance A is fixed to the median of Dvec
    Aest.REML<-0
    Aest.REML[1]<-median(Dvec)

    k<-0
    diff<-1
    while ((diff>0.0001)&(k<MAXITER))
```

```
{
  k<-k+1
  Vi<-1/(Aest.REML[k]+Dvec)
  XtVi<-t(Vi*X)
  Q<-solve(XtVi%*%X)
  P<-diag(Vi)-t(XtVi)%*%Q%*%XtVi
  Py<-P%*%y
  # Score function obtained from restricted log-likelihood
  s<-(-0.5)*sum(diag(P))+0.5*(t(Py)%*%Py)
  # Fisher information obtained from restricted log-likelihood
  F<-0.5*sum(diag(P%*%P))
  # Updating equation
  Aest.REML[k+1]<-Aest.REML[k]+s/F
  # Relative difference of estimators in 2 iterations
  # for stopping condition
  diff<-abs((Aest.REML[k+1]-Aest.REML[k])/Aest.REML[k])
} # End of while

# Final estimator of variance A
A.REML<-max(Aest.REML[k+1],0)
print(Aest.REML)

# Indicator of convergence

if(k<MAXITER) {conv<-TRUE} else {conv<-FALSE}

# Computation of the coefficients'estimator beta

Vi<-1/(A.REML+Dvec)
XtVi<-t(Vi*X)
Q<-solve(XtVi%*%X)
beta.REML<-Q%*%XtVi%*%y

# Significance of the regression coefficients

varA<-1/F

std.errorbeta<-sqrt(diag(Q))
tvalue<-beta.REML/std.errorbeta
pvalue<-2*pnorm(abs(tvalue),lower.tail=FALSE)
```

```
  # Goodness of fit measures: loglikelihood, AIC, BIC

  Xbeta.REML<-X%*%beta.REML
  resid<-y-Xbeta.REML

  loglike<-(-0.5)*(sum(log(2*pi*(A.REML+Dvec))
          +(resid^2)/(A.REML+Dvec)))
  AIC<-(-2)*loglike+2*(p+1)
  BIC<-(-2)*loglike+(p+1)*log(m)

  goodness<-c(loglike=loglike,AIC=AIC,BIC=BIC)

  # Computation of the empirical best (EB) predictor

  thetaEB.REML<-Xbeta.REML+A.REML*Vi*resid
  coef<-data.frame(beta.REML,std.errorbeta,tvalue,pvalue)

  return(list(convergence=conv,modelcoefficients=coef,
  variance=A.REML,goodnessoffit=goodness,EBpredictor=thetaEB.REML))

  # Fisher-scoring algorithm for REML estimator of variance A starts

} else if (method=="FH") {

   # Initial value of variance A is fixed to the median of Dvec
   Aest.FH<-NULL
   Aest.FH[1]<-median(Dvec)

   k<-0
   diff<-1
   while ((diff>0.0001)&(k<MAXITER)){
     k<-k+1
     Vi<-1/(Aest.FH[k]+Dvec)
     XtVi<-t(Vi*X)
     Q<-solve(XtVi%*%X)
     betaaux<-Q%*%XtVi%*%y
     resaux<-y-X%*%betaaux
     # Left-hand side of equation for FH estimator
     s<-sum((resaux^2)*Vi)-(m-p)
```

```
  # Expectation of negative derivative of s
  F<-sum(Vi)
  # Updating equation
  Aest.FH[k+1]<-Aest.FH[k]+s/F
  # Relative difference of estimators in 2 iterations
  # for stopping condition
  diff<-abs((Aest.FH[k+1]-Aest.FH[k])/Aest.FH[k])

} # End of while

A.FH<-max(Aest.FH[k+1],0)
print(Aest.FH)

# Indicator of convergence

if(k<MAXITER) {conv<-TRUE} else {conv<-FALSE}

# Computation of the coefficients'estimator beta

Vi<-1/(A.FH+Dvec)
XtVi<-t(Vi*X)
Q<-solve(XtVi%*%X)
beta.FH<-Q%*%XtVi%*%y

# Significance of the regression coefficients

varA<-1/F
varbeta<-diag(Q)
std.errorbeta<-sqrt(varbeta)
zvalue<-beta.FH/std.errorbeta
pvalue<-2*pnorm(abs(zvalue),lower.tail=FALSE)

# Goodness of fit measures: loglikelihood, AIC, BIC

Xbeta.FH<-X%*%beta.FH
resid<-y-Xbeta.FH

loglike<-(-0.5)*(sum(log(2*pi*(A.FH+Dvec))+(resid^2)/(A.FH+Dvec)))
AIC<-(-2)*loglike+2*(p+1)
BIC<-(-2)*loglike+(p+1)*log(m)
```

```
    goodness<-c(loglike=loglike,AIC=AIC,BIC=BIC)

    # Computation of the empirical best (EB) predictor

    thetaEB.FH<-Xbeta.FH+A.FH*Vi*resid
    coef<-data.frame(beta.FH,std.errorbeta,zvalue,pvalue)

    return(list(convergence=conv,modelcoefficients=coef,variance=A.FH,
        goodnessoffit=goodness,EBpredictor=thetaEB.FH))

    # Error printing when method is different from REML of FH.
  }  else { print("Error: Unknown method") }

}
```

7.2 R code of MSE.FHmodel

The R code of the function `MSE.FHmodel` is listed bellow.

```
#####################################################################
### ### This function gives the MSE estimator of the EB estimator
under a FH model ### The EB estimator is obtained either by REML
or by FH fitting methods ### ### Work for European project SAMPLE
### ### Author: Isabel Molina Peralta ### File name: MSE_FHModel.R
### Updated: March 15th, 2010 ###
#####################################################################

MSE.FHmodel<-function(X,Dvec,A,method="REML"){

  m<-dim(X)[1] # Sample size or number of areas
  p<-dim(X)[2] # Num. of X columns of num. of auxiliary variables

  # Initialize vectors containing the values of g1-g3 and mse
  # for each area
  g1d<-rep(0,m)
  g2d<-rep(0,m)
  g3d<-rep(0,m)
  mse2d<-rep(0,m)
```

```
# Elements of the inverse covariance matrix in a vector
Vi<-1/(A+Dvec)
# Auxiliary calculations
Bd<-Dvec/(A+Dvec)
SumAD2<-sum(Vi^2)
XtVi<-t(Vi*X)
Q<-solve(XtVi%*%X)

# Calculation of g1-g3 and final MSE when fitting method is REML

if (method=="REML"){

  # Asymptotic variance of REML estimator of variance A
  VarA<-2/SumAD2

  for (d in 1:m){
    g1d[d]<-Dvec[d]*(1-Bd[d])
    xd<-matrix(X[d,],nr=1,nc=p)
    g2d[d]<-(Bd[d]^2)*xd%*%Q%*%t(xd)
    g3d[d]<-(Bd[d]^2)*VarA/(A+Dvec[d])
    mse2d[d]<-g1d[d]+g2d[d]+2*g3d[d]
  }

return(mse=mse2d)

# Calculation of g1-g3 and final MSE when fitting method is FH

} else if (method=="FH") {

  SumAD<-sum(Vi)
  # Asymptotic variance of FH estimator of variance A
  VarA<-2*m/(SumAD^2)

  # Asymptotic bias of FH estimator of A
  b<-2*(m*SumAD2-SumAD^2)/(SumAD^3)

  for (d in 1:m){
    g1d[d]<-Dvec[d]*(1-Bd[d])
    xd<-matrix(X[d,],nr=1,nc=p)
    g2d[d]<-(Bd[d]^2)*xd%*%Q%*%t(xd)
```

```
      g3d[d]<-(Bd[d]^2)*VarA/(A+Dvec[d])
      mse2d[d]<-g1d[d]+g2d[d]+2*g3d[d]-b*(Bd[d]^2)
    }

  return(mse=mse2d)

  # Error printing when fitting method is different from REML of FH.
  }  else { print("Error: Unknown method") }

}
```

Chapter 8

Appendix 2: R code for Spatial Fay-Herriot model

8.1 R code of fitSpatialFH

The R code of the function `fitSpatialFH` is listed bellow.

```
#####################################################################
### ### This function fits a spatial Fay-Herriot model, ### in
which random effects follow a Simultaneously Autorregressive (SAR)
process ### Fitting method can be chosen between REML and ML
methods ### ### Work for European project SAMPLE ### ### Author:
Isabel Molina Peralta ### File name: Fitting_SpatialFHModel.R ###
Updated: March 15th, 2010 ###
#####################################################################


fitSpatialFH<-function(X,y,Dvec,W,method="REML",MAXITER=500) {

  m<-length(y)  # Sample size or number of areas
  p<-dim(X)[2]  # Num. of X columns of num. of auxiliary variables
  Xt<-t(X)
  yt<-t(y)
  Wt<-t(W)
  I<-diag(1,m)

  # Initialize vectors containing estimators of variance and
  # spatial correlation
  par.stim<-matrix(0,2,1)
```

```
stime.fin<-matrix(0,2,1)

# Initialize scores vector and Fisher information matrix
s<-matrix(0,2,1)
Idev<-matrix(0,2,2)

# Initial value of variance set to the mean of sampling variances Dvec
# Initial value of spatial correlation set to 0.5
sigma2.u.stim.S<-0
rho.stim.S<-0

sigma2.u.stim.S[1]<-median(Dvec)
rho.stim.S[1]<-0.5

# Fisher-scoring algorithm for REML estimators start

if (method=="REML"){

  k<-0
  diff.S<-1

  while ((diff.S>0.0001)&(k<MAXITER)){

    k<-k+1
    # Derivative of covariance matrix V with respect to variance
    derSigma<-solve((I-rho.stim.S[k]*Wt)%*%(I-rho.stim.S[k]*W))
    # Derivative of covariance matrix V with respect to
    # spatial autocorrelation
    derRho<-2*rho.stim.S[k]*Wt%*%W-W-Wt
    derVRho<-(-1)*sigma2.u.stim.S[k]*(derSigma%*%derRho%*%derSigma)
    # Covariance matrix and inverse covariance matrix
    V<-sigma2.u.stim.S[k]*derSigma+I*Dvec
    Vi<-solve(V)
    # Matrix P and coefficients'estimator beta
    XtVi<-Xt%*%Vi
    Q<-solve(XtVi%*%X)
    P<-Vi-t(XtVi)%*%Q%*%XtVi
    b.s<-Q%*%XtVi%*%y
    # Terms involved in scores vector and Fisher information matrix
    PD<-P%*%derSigma
```

```
    PR<-P%*%derVRho
    Pdir<-P%*%y
    # Scores vector
    s[1,1]<-(-0.5)*sum(diag(PD))+(0.5)*(yt%*%PD%*%Pdir)
    s[2,1]<-(-0.5)*sum(diag(PR))+(0.5)*(yt%*%PR%*%Pdir)
    # Fisher information matrix
    Idev[1,1]<-(0.5)*sum(diag(PD%*%PD))
    Idev[1,2]<-(0.5)*sum(diag(PD%*%PR))
    Idev[2,1]<-Idev[1,2]
    Idev[2,2]<-(0.5)*sum(diag(PR%*%PR))

    # Updating equation
    par.stim[1,1]<-sigma2.u.stim.S[k]
    par.stim[2,1]<-rho.stim.S[k]

    stime.fin<-par.stim+solve(Idev)%*%s
    print(stime.fin)

    # Restrict spatial correlation to (-0.999,0.999) and variance
    # to (0.0001,infty)
    if ((stime.fin[2,1]<=0.999)&(stime.fin[2,1]>=-0.999)&
      (stime.fin[1,1]>0.0001)){
      sigma2.u.stim.S[k+1]<-stime.fin[1,1]
      rho.stim.S[k+1]<-stime.fin[2,1]
      diff.S<-max(abs(stime.fin-par.stim)/par.stim)
    }else{ diff.S<-0.00001 }

  } # End of while

# Fisher-scoring algorithm for REML estimators start

}else if (method=="ML"){

  k<-0
  diff.S<-1

  while ((diff.S>0.0001)&(k<MAXITER)){

    k<-k+1
    # Derivative of covariance matrix V with respect to variance
```

```
derSigma<-solve((I-rho.stim.S[k]*Wt)%*%(I-rho.stim.S[k]*W))
# Derivative of covariance matrix V with respect to
# spatial autocorrelation
derRho<-2*rho.stim.S[k]*Wt%*%W-W-Wt
derVRho<-(-1)*sigma2.u.stim.S[k]*(derSigma%*%derRho%*%derSigma)
# Covariance matrix and inverse covariance matrix
V<-sigma2.u.stim.S[k]*derSigma+I*Dvec
Vi<-solve(V)
# Coefficients'estimator beta and matrix P
XtVi<-Xt%*%Vi
Q<-solve(XtVi%*%X)
P<-Vi-t(XtVi)%*%Q%*%XtVi
b.s<-Q%*%XtVi%*%y
# Terms involved in scores vector and Fisher information matrix
PD<-P%*%derSigma
PR<-P%*%derVRho
Pdir<-P%*%y
ViD<-Vi%*%derSigma
ViR<-Vi%*%derVRho
# Scores vector
s[1,1]<-(-0.5)*sum(diag(ViD))+(0.5)*(yt%*%PD%*%Pdir)
s[2,1]<-(-0.5)*sum(diag(ViR))+(0.5)*(yt%*%PR%*%Pdir)
# Fisher information matrix
Idev[1,1]<-(0.5)*sum(diag(ViD%*%ViD))
Idev[1,2]<-(0.5)*sum(diag(ViD%*%ViR))
Idev[2,1]<-Idev[1,2]
Idev[2,2]<-(0.5)*sum(diag(ViR%*%ViR))

# Updating equation
par.stim[1,1]<-sigma2.u.stim.S[k]
par.stim[2,1]<-rho.stim.S[k]

stime.fin<-par.stim+solve(Idev)%*%s

# Restrict spatial correlation to (-0.999,0.999) and variance
# to (0.0001,infty)
if ((stime.fin[2,1]<=0.999)&(stime.fin[2,1]>=-0.999)&
  (stime.fin[1,1]>0.0001)){
  sigma2.u.stim.S[k+1]<-stime.fin[1,1]
  rho.stim.S[k+1]<-stime.fin[2,1]
```

```
      diff.S<-max(abs(stime.fin-par.stim)/par.stim)
    }else{ diff.S<-0.00001 }

  } # End of while
# Error message if method is different from REML or ML
} else { print("Error: Unknown method") }

# Final values of estimators
rho<-rho.stim.S[k+1]
sigma2u<-max(sigma2.u.stim.S[k+1],0)
#print(c(sigma2u,rho))

# Indicator of convergence

if(k<MAXITER) {conv<-TRUE} else {conv<-FALSE}

# Computation of the coefficients'estimator (Bstim)

A<-solve((I-rho*Wt)%*%(I-rho*W))
G<-sigma2u*A
V<-G+I*Dvec
Vi<-solve(V)
XtVi<-Xt%*%Vi
Q<-solve(XtVi%*%X)
Bstim<-Q%*%XtVi%*%y

# Significance of the regression coefficients

std.errorbeta<-sqrt(diag(Q))
tvalue<-Bstim/std.errorbeta
pvalue<-2*pnorm(abs(tvalue),lower.tail=FALSE)

coef<-data.frame(beta=Bstim,std.errorbeta,tvalue,pvalue)

# Goodness of fit measures: loglikelihood, AIC, BIC

Xbeta<-X%*%Bstim
resid<-y-Xbeta

loglike<-(-0.5)*(m*log(2*pi)+determinant(V,logarithm=T)$modulus+
```

```
    t(resid)%*%Vi%*%resid)

  AIC<-(-2)*loglike+2*(p+2)
  BIC<-(-2)*loglike+(p+2)*log(m)

  goodness<-c(loglike=loglike,AIC=AIC,BIC=BIC)

  # Computation of the Spatial EBLUP

  res<-y-X%*%Bstim
  thetaSpat<-X%*%Bstim+G%*%Vi%*%res

  return(list(convergence=conv,modelcoefficients=coef,variance=sigma2u,
    spatialcorr=rho,goodnessoffit=goodness,EBpredictor=thetaSpat))

}
```

8.2 R code of MSE.SpatialFH

The R code of the function `MSE.SpatialFH` is listed bellow.

```
#################################################################
### ### This function gives the analytical MSE estimator of the EB
estimator ### under a Spatial FH model, in which random effects
follow a ### Simultaneously Autorregressive (SAR) process. ### The
EB estimator is obtained by REML fitting method ### ### Work for
European project SAMPLE ### ### Author: Isabel Molina Peralta ###
File name: MSE_SpatialFHModel.R ### Updated: March 15th, 2010 ###
#################################################################

MSE.SpatialFHmodel<-function(X,Dvec,A,rho,W,method="REML"){

  m<-dim(X)[1]
  p<-dim(X)[2]

  g1d<-rep(0,m)
  g2d<-rep(0,m)
  g3d<-rep(0,m)
  g5d<-rep(0,m)
  mse2d.aux<-rep(0,m)
```

```
mse2d<-rep(0,m)

I<-diag(1,m)
Xt<-t(X)
Wt<-t(W)
Ci<-solve((I-rho*Wt)%*%(I-rho*W))
G<-A*Ci
V<-G+I*Dvec
Vi<-solve(V)
XtVi<-Xt%*%Vi
Q<-solve(XtVi%*%X)

Ga<-G-G%*%Vi%*%G

# g1 contains the diagonal elements of Ga
for (i in 1:m) {g1d[i]<-Ga[i,i]}

Gb<-G%*%Vi%*%X
Xa<-matrix(0,1,p)

for (i in 1:m) {
  Xa[1,]<-X[i,]-Gb[i,]
  g2d[i]<-Xa%*%Q%*%t(Xa)
}

derRho<-2*rho*Wt%*%W-W-Wt
Amat<-(-1)*A*(Ci%*%derRho%*%Ci)
P<-Vi-t(XtVi)%*%Q%*%XtVi
PCi<-P%*%Ci
PAmat<-P%*%Amat

Idev<-matrix(0,2,2)
Idev[1,1]<-(0.5)*sum(diag((PCi%*%PCi)))
Idev[1,2]<-(0.5)*sum(diag((PCi%*%PAmat)))
Idev[2,1]<-Idev[1,2]
Idev[2,2]<-(0.5)*sum(diag((PAmat%*%PAmat)))
Idevi<-solve(Idev)

ViCi<-Vi%*%Ci
ViAmat<-Vi%*%Amat
```

```
  l1<-ViCi-A*ViCi%*%ViCi
  l1t<-t(l1)
  l2<-ViAmat-A*ViAmat%*%ViCi
  l2t<-t(l2)
  L<-matrix(0,2,m)
  for (i in 1:m)
  {
    L[1,]<-l1t[i,]
    L[2,]<-l2t[i,]
    g3d[i]<-sum(diag(L%*%V%*%t(L)%*%Idevi))
  }

  mse2d.aux<-g1d+g2d+2*g3d

  # Bias correction of Singh et al

  psi<-diag(Dvec,m)
  D12aux<-(-1)*(Ci%*%derRho%*%Ci)
  D22aux<-2*A*Ci%*%derRho%*%Ci%*%derRho%*%Ci-2*A*Ci%*%Wt%*%W%*%Ci
  D<-psi%*%Vi%*%D12aux%*%Vi%*%psi*Idevi[1,2]
    +psi%*%Vi%*%D12aux%*%Vi%*%psi*Idevi[2,1]+
    psi%*%Vi%*%D22aux%*%Vi%*%psi*Idevi[2,2]

  for (i in 1:m) {g5d[i]<-(0.5)*D[i,i]}

  # Computation of analytical estimated of Singh et al

  mse2d<-mse2d.aux-g5d

  return(mse=mse2d)

}
```

8.3 R code of PBMSE.SpatialFH

The R code of the function `PBMSE.SpatialFH` is listed bellow.

```
#################################################################
### ### This function gives a parametric bootstrap MSE estimator
```

```
of the EB estimator ### under a Spatial FH model, in which random
effects follow a ### Simultaneously Autorregressive (SAR) process.
### The EB estimator is obtained either by REML or by FH fitting
methods ### ### Work for European project SAMPLE ### ### Author:
Isabel Molina Peralta ### File name: PBMSE_SpatialFHModel.R ###
Updated: March 15th, 2010 ###
#####################################################################

source("Fitting_SpatialFHModel.R")

PBMSE.SpatialFHmodel<-function(X,Dvec,beta,A,rho,W,n.boot,
method="REML"){

  m<-dim(X)[1]  # Sample size or number of areas
  p<-dim(X)[2]  # Num. of X columns of num. of auxiliary variables

  # Initial estimators of model coefficients, variance and spatial
  # correlation actong as true values in the bootstrap procedure.

  Bstim.boot<-beta
  rho.boot<-rho
  sigma2.boot<-A

  I<-diag(1,m)
  Xt<-t(X)
  Wt<-t(W)

  # Analytical estimators of g1 and g2, used for
  # the   bias-corrected PB MSE estimator

  g1sp<-rep(0,m)
  g2sp<-rep(0,m)

  Amat.sblup<-solve((I-rho.boot*Wt)%*%(I-rho.boot*W))
  G.sblup<-sigma2.boot*Amat.sblup
  V.sblup<-G.sblup+I*Dvec
  V.sblupi<-solve(V.sblup)
  XtV.sblupi<-Xt%*%V.sblupi
  Q.sblup<-solve(XtV.sblupi%*%X)
```

```
# Calculate g1

Ga.sblup<-G.sblup-G.sblup%*%V.sblupi%*%G.sblup
for (i in 1:m) {g1sp[i]<-Ga.sblup[i,i]}

# Calculate g2

Gb.sblup<-G.sblup%*%t(XtV.sblupi)
Xa.sblup<-matrix(0,1,p)

for (i in 1:m){
  Xa.sblup[1,]<-X[i,]-Gb.sblup[i,]
  g2sp[i]<-Xa.sblup%*%Q.sblup%*%t(Xa.sblup)
}

# Initialize vectors adding g1, g2, g3 and naive PB MSE estimators
summse.pb<-rep(0,m)
sumg1sp.pb<-rep(0,m)
sumg2sp.pb<-rep(0,m)
sumg3sp.pb<-rep(0,m)
g1sp.aux<-rep(0,m)
g2sp.aux<-rep(0,m)

# Bootstrap cycle starts

for (boot in 1:n.boot) {

  cat(date(),"Bootstrap iteration",boot,"\n",fill=T)

  # Generate a bootstrap sample
  u.boot<-rnorm(m,0,sqrt(sigma2.boot))
  v.boot<-solve(I-rho.boot*W)%*%u.boot
  theta.boot<-X%*%Bstim.boot+v.boot
  e.boot<-rnorm(m,0,sqrt(Dvec))
  direct.boot<-theta.boot+e.boot

  # Fit of the model to bootstrap data

  resultsSp<-fitSpatialFH(X,direct.boot,Dvec,W,method="REML")
```

```
# While the estimators are not satisfactory
# we generate a new sample
conv<-resultsSp$convergence
v<-resultsSp$variance
r<-resultsSp$spatialcorr
print(conv)
print(c(v,r))

if (conv==FALSE) {v<-0}
if (is.na(v)==TRUE) {v<-0}
if (is.na(r)==TRUE) {r<-1}

while ((v<0.0001)|(r<(-0.999))|(r>0.999)){
  print("New sample")
  u.boot<-rnorm(m,0,sqrt(sigma2.boot))
  v.boot<-solve(I-rho.boot*W)%*%u.boot
  theta.boot<-X%*%Bstim.boot+v.boot
  e.boot<-rnorm(m,0,sqrt(Dvec))
  direct.boot<-theta.boot+e.boot
  resultsSp<-fitSpatialFH(X,direct.boot,Dvec,W,method="REML")
  conv<-resultsSp$convergence
  v<-resultsSp$variance
  r<-resultsSp$spatialcorr
  if (conv==FALSE) {v<-0}
  if (is.na(v)==TRUE) {v<-0}
  if (is.na(r)==TRUE) {r<-1}
}

# Fit of the model to bootstrap data

sigma2.simula.ML<-resultsSp$variance
rho.simula.ML<-resultsSp$spatialcorr
beta.ML<-resultsSp$modelcoefficients$beta

# Calculation of the bootstrap Spatial EBLUP

Amat<-solve((I-rho.simula.ML*Wt)%*%(I-rho.simula.ML*W))
G<-sigma2.simula.ML*Amat
V<-G+I*Dvec
Vi<-solve(V)
```

```
Xbeta<-X%*%beta.ML
thetaEBLUPSpat.boot<-Xbeta+G%*%Vi%*%(direct.boot-Xbeta)

# Naive parametric bootstrap MSE

summse.pb<-summse.pb+(thetaEBLUPSpat.boot-theta.boot)^2

# Bias-corrected parametric bootstrap:
# For de bias of g1 and g2, calculate g1sp and g2sp for
# each bootstrap sample

XtVi<-Xt%*%Vi
Q<-solve(XtVi%*%X)

Ga<-G-G%*%Vi%*%G
for (i in 1:m) {g1sp.aux[i]<-Ga[i,i]}
# g1sp contains the diagonal elements of Ga

Gb<-G%*%Vi%*%X
Xa<-matrix(0,1,p)

for (i in 1:m){
  Xa[1,]<-X[i,]-Gb[i,]
  g2sp.aux[i]<-Xa%*%Q%*%t(Xa)
}

# Bootstrap spatial BLUP
Bstim.sblup<-solve(XtV.sblupi%*%X)%*%XtV.sblupi%*%direct.boot
Xbeta.sblup<-X%*%Bstim.sblup
thetaEBLUPSpat.sblup.boot<-Xbeta.sblup
+G.sblup%*%V.sblupi%*%(direct.boot-Xbeta.sblup)

# Parametric bootstrap estimator of g3
sumg3sp.pb<-sumg3sp.pb
+(thetaEBLUPSpat.boot-thetaEBLUPSpat.sblup.boot)^2

# Expectation of estimated g1 and g2
sumg1sp.pb<-sumg1sp.pb+g1sp.aux
sumg2sp.pb<-sumg2sp.pb+g2sp.aux
```

```
  } # End of bootstrap cycle

  # Final naive parametric bootstrap MSE estimator
  mse.pb<-summse.pb/n.boot
  # Final bias-corrected bootstrap MSE estimator
  g1sp.pb<-sumg1sp.pb/n.boot
  g2sp.pb<-sumg2sp.pb/n.boot
  g3sp.pb<-sumg3sp.pb/n.boot
  mse.pb2<-2*(g1sp+g2sp)-g1sp.pb-g2sp.pb+g3sp.pb

  # Return naive and bias-corrected parametric bootstrap
  return(data.frame(PBmse=mse.pb,bcPBmse=mse.pb2))

}
```

Chapter 9

Appendix 3.1: R code for area-level models with independent time effects

9.1 R code of H3area

The R code of the function **H3area** is listed bellow.

```
#####################################################################
###
###      Area level model with independent time effects
###                  SAMPLE project
###
### Author: Agustin Perez Martin
### File name: H3.R
### Updated: November 25th, 2009
###
#####################################################################

H3area <- function(X, ydt, D, md, sigma2edt) {

    p <- ncol(X)
    a <- list(1:md[1])
    mdcum <- cumsum(md)
    M <- sum(md)

    for(d in 2:D)
        a[[d]] <- (mdcum[d-1]+1):mdcum[d]

    yd <- Xd <- list()
```

```
    for(d in 1:D) {
        yd[[d]] <- ydt[a[[d]]]
        Xd[[d]] <- X[a[[d]],]
    }

    Vd.inv <- VinvXd <- list()
    Q2.inv <- XV2X <- matrix(0, nrow=p, ncol=p)
    yVX <- 0
    for(d in 1:D) {

        ### Elements of the variance matrix
        vd <- sigma2edt[a[[d]]]

        ### Inverse matrix of the variance and submatrices
        Vd.inv[[d]] <- diag(1/vd)

        ### Product between V^-1_ed  and  X_d for all d submatrices
        VinvXd[[d]] <- Vd.inv[[d]]%*%Xd[[d]]

        ### Inverse of Q2. Next we calculate Q2
        Q2.inv <- Q2.inv + t(Xd[[d]])%*%VinvXd[[d]]

        ### Sum in d of the product with  y^t_d  and  V^-1_ed  and  X_d
        yVX <- yVX + yd[[d]]%*%VinvXd[[d]]
    }
    Q2 <- solve(Q2.inv)

    tr.XV2XQ2 <- 0
    for(d in 1:D)
        tr.XV2XQ2 <- tr.XV2XQ2
                     + sum(diag( t(VinvXd[[d]])%*%VinvXd[[d]]%*%Q2))

    tr.P2 <- sum(1/sigma2edt) - tr.XV2XQ2
    yP2y <- sum(ydt^2/sigma2edt) - yVX%*%Q2%*%t(yVX)

    sigma.u <- (yP2y - (M-p))/tr.P2

    return(as.vector(sigma.u))
}
```

9.2 R code of REMLarea.indep

The R code of the function **REMLarea.indep** is listed bellow.

```
#########################################################################
###
###      Area level model with independent time effects
###                  SAMPLE project
###
### Author: Agustin Perez Martin
### File name: REMLindep.R
### Updated: November 25th, 2009
###
#########################################################################

REMLarea.indep <- function(X,ydt,D,md,sigma2edt,sigma.0,MAXITER=500){

    sigma.f <- sigma.0
    p <- ncol(X)
    a <- list(1:md[1])
    mdcum <- cumsum(md)
    for(d in 2:D)
        a[[d]] <- (mdcum[d-1]+1):mdcum[d]


    yd <- Xd <- list()
    for(d in 1:D) {
        yd[[d]] <- ydt[a[[d]]]
        Xd[[d]] <- X[a[[d]],]
    }

    for(ITER in 1:MAXITER){
        Vd.inv <- Vinvyd <- VinvXd <- list()
        Q.inv <- XV2X <- XV3X <- matrix(0, nrow=p, ncol=p)
        tr.V.inv <- tr.V2.inv <- yV2y <- yVX <- XV2y <- 0
        for(d in 1:D) {

          ### Elements of the variance matrix
          vd <- (sigma.f + sigma2edt)[a[[d]]]

          ### Inverse matrix of the variance and submatrices
```

```
  Vd.inv[[d]] <- diag(1/vd)

  ### Product between V^-1_ed  and  y_d for all d submatrices
  Vinvyd[[d]] <- Vd.inv[[d]]%*%yd[[d]]

  ### Product between V^-1_ed  and  X_d for all d submatrices
  VinvXd[[d]] <- Vd.inv[[d]]%*%Xd[[d]]

  ### Inverse of Q. Next we calculate Q
  Q.inv <- Q.inv + t(Xd[[d]])%*%VinvXd[[d]]

  ### Sum traces of V^-1_d
  tr.V.inv <- tr.V.inv + sum(1/vd)

  ### Sum traces of V^-2_d
  tr.V2.inv <- tr.V2.inv + sum(1/vd^2)

  ### Sum on d of the product between  X^t_d, V^-2_d and X_d
  XV2X <- XV2X + t(VinvXd[[d]])%*%VinvXd[[d]]

  ### Sum on d of the product between  X^t_d, V^-3_d and X_d
  XV3X <- XV3X + t(VinvXd[[d]])%*%Vd.inv[[d]]%*%VinvXd[[d]]

  ### Sum on d of the product between  y^t_d, V^-2_d and y_d
  yV2y <- yV2y + t(Vinvyd[[d]])%*%Vinvyd[[d]]

  ### Sum on d of the product between  y^t_d, V^-1_d and X_d
  yVX <- yVX + yd[[d]]%*%VinvXd[[d]]

  ### Sum on d of the product between  X^t_d, V^-2_d and y_d
  XV2y <- XV2y + t(VinvXd[[d]])%*%Vinvyd[[d]]
}
Q <- solve(Q.inv)

tr.XV2XQ <- 0
for(d in 1:D)
    tr.XV2XQ <- tr.XV2XQ
                + sum(diag( t(VinvXd[[d]])%*%VinvXd[[d]]%*%Q))

tr.P <- tr.V.inv - tr.XV2XQ
```

```
        tr.P2 <- tr.V2.inv - 2*sum(diag(XV3X%*%Q))
                 + sum(diag(XV2X%*%Q%*%XV2X%*%Q))
        yP2y <- yV2y - 2*yVX%*%Q%*%XV2y + yVX%*%Q%*%XV2X%*%Q%*%t(yVX)

        ### Scores and Fisher information matrix
        Ssig <- -0.5*tr.P + 0.5*yP2y
        Fsig <- 0.5*tr.P2

        ### Fisher-Scoring Algorithm
        dif <- Ssig/Fsig
        sigma.f <- sigma.f + dif

        ### Stopping criterion
        if(abs(dif)<0.000001)
            break
    }

    return(list(as.vector(sigma.f), Fsig, Q))
}
```

9.3 R code of BETA.U.area.indep

The R code of the function **BETA.U.area.indep** is listed bellow.

```
#####################################################################
###
###      Area level model with independent time effects
###                SAMPLE project
###
### Author: Agustin Perez Martin
### File name: EstimationBETAindep.R
### Updated: November 25th, 2009
###
#####################################################################

BETA.U.area.indep <- function(X, ydt, D, md, sigma2edt, sigmau) {

    p <- ncol(X)
    a <- list(1:md[1])
    mdcum <- cumsum(md)
```

```
    for(d in 2:D)
        a[[d]] <- (mdcum[d-1]+1):mdcum[d]

    yd <- Xd <- Vd <- Vd.inv <- list()
    Q.inv <- matrix(0, nrow=p, ncol=p)
    XVy <- 0
    for(d in 1:D) {
      yd[[d]] <- ydt[a[[d]]]
      Xd[[d]] <- X[a[[d]],]

      ### Elements of the variance matrix
      vd <- (sigmau + sigma2edt)[a[[d]]]

      ### Inverse matrix of the variance and d submatrices
      Vd.inv[[d]] <- diag(1/vd)

      ### Inverse of Q. Next we calculate Q
      Q.inv <- Q.inv + t(Xd[[d]])%*%Vd.inv[[d]]%*%Xd[[d]]

      ### Product between X^t_d,  V^-1_d and y_d for all d submatrices
      XVy <- XVy + t(Xd[[d]])%*%Vd.inv[[d]]%*%yd[[d]]
    }
    Q <- solve(Q.inv)

    beta <- Q%*%XVy

    u <- list()
    for(d in 1:D)
        u[[d]] <- sigmau*Vd.inv[[d]]%*%(yd[[d]]-Xd[[d]]%*%beta)
    u <- as.matrix(unlist(u))

    return(rbind(beta,u))
}
```

9.4 R code of mse.area.indep

The R code of the function **mse.area.indep** is listed bellow.

```
#####################################################################
###
###    Area level model with independent time effects
###                SAMPLE project
###
### Author: Agustin Perez Martin
### File name: EstimationMSEindep.R
### Updated: November 25th, 2009
###
#####################################################################

mse.area.indep <- function(X, D, md, sigma2edt, sigmau, Fsig) {

p <- ncol(X)
a <- list(1:md[1])
mdcum <- cumsum(md)
for(d in 2:D)
a[[d]] <- (mdcum[d-1]+1):mdcum[d]

Xd <- Vd.inv <- Sed.inv <- SinvXd <- VinvSinvXd <- g2.a <- list()
Q.inv <- matrix(0, nrow=p, ncol=p)
XVy <- 0
for(d in 1:D) {
  Xd[[d]] <- X[a[[d]],]

  ### Elements of the variance matrix
  vd <- (sigmau + sigma2edt)[a[[d]]]

  ### Inverse matrix of the variance and d submatrices
  Vd.inv[[d]] <- diag(1/vd)

  ### Elements of the variance matrix Sigma_e
  sd <- sigma2edt[a[[d]]]

  ### Inverse matrix of Sigma_ed in all d submatrices
  Sed.inv[[d]] <- diag(1/sd)
```

```
  ### Product between Sigma^-1_ed and X_d for all d submatrices
  SinvXd[[d]] <- Sed.inv[[d]]%*%Xd[[d]]

  ### Product between V^-1_d, Sigma^-1_ed and X_d for all d submatrices
  VinvSinvXd[[d]] <- Vd.inv[[d]]%*%SinvXd[[d]]

  ### First part of g2 (the second is its transpose)
  g2.a[[d]] <- Xd[[d]] - sigmau*SinvXd[[d]] + sigmau^2*VinvSinvXd[[d]]

  ### Inverse of Q. Next we calculate Q
  Q.inv <- Q.inv + t(Xd[[d]])%*%Vd.inv[[d]]%*%Xd[[d]]
 }
    Q <- solve(Q.inv)

    ### Elements of the variance matrix
    vd <- sigmau + sigma2edt
    q <- 1/vd - 2*(sigmau/vd^2) + (sigmau^2/vd^3)

    ### Calculation of g
    g1 <- (sigmau*sigma2edt)/vd
    g2 <- list()
    for(d in 1:D)
        g2[[d]] <- diag(g2.a[[d]]%*%Q%*%t(g2.a[[d]]))
    g2 <- unlist(g2)
    g3 <- q/Fsig

    return(g1+g2+2*g3)
}
```

9.5 R code of Interval.indep

The R code of the function **Interval.indep** is listed bellow.

```
#####################################################################
###
###    Area level model with independent time effects
###                  SAMPLE project
###
### Author: Agustin Perez Martin
### File name: ICindep.R
### Updated: November 25th, 2009
###
#####################################################################

Interval.indep <- function(fit, conf=0.95) {
    alfa <- 1-conf
    k <- 1-alfa/2
    z <- qnorm(k)

    Finv <- solve(fit[[2]])

    sigma.std.err <- z*sqrt(Finv[1,1])
    beta.std.err <- z*sqrt(as.vector(diag(fit[[3]])))

    return( list(sigma.std.err, beta.std.err) )
}
```

9.6 R code of pvalue

The R code of the function **pvalue** is listed bellow.

```
#####################################################################
####
###    Area level model with independent time effects
###        and with time correlated effects
###                 SAMPLE project
###
### Author: Agustin Perez Martin
### File name: pvalue.R
### Updated: November 25th, 2009
###
#####################################################################

pvalue <- function(beta0, fit) {

    z <- abs(beta0)/sqrt(as.vector(diag(fit[[3]])))
    pval <- 2*pnorm(z, lower.tail=F)

    return( pval )
}
```

Chapter 10

Appendix 3.2: R code for area-level models with correlated time effects

10.1 R code of REMLarea.autocorr

The R code of the function **REMLarea.autocorr** is listed bellow.

```
#########################################################################
###
###        Area level model with time correlated effects
###                        SAMPLE project
###
### Author: Maria Dolores Esteban Lefler
### File name: REMLautocorr.R
### Updated: November 25th, 2009
###
#########################################################################

REMLarea.autocorr<-function(X,ydt,D,md,sigma2edt,sigma.0,MAXITER=500){

rho.f <- 0
sigma.f <- sigma.0
theta.f  <- c(sigma.f,rho.f)
p <- ncol(X)
a <- list(1:md[1])
mdcum <- cumsum(md)
for(d in 2:D)
   a[[d]] <- (mdcum[d-1]+1):mdcum[d]
```

```
   yd <- Xd <- list()
    for(d in 1:D) {
       yd[[d]] <- ydt[a[[d]]]
       Xd[[d]] <- X[a[[d]],]
    }

for(ITER in 1:MAXITER){
    Vd.inv <- Vad  <- Vbd <- VinvVad <- VinvVbd  <-
    Vinvyd <- VinvXd <- XtVinvVadVinvX  <- XtVinvVbdVinvX <-
    VinvVadVinvVad <-  VinvVadVinvVad <- VinvVadVinvVad <-
    XtVinvVadVinvVadVinvX <- VinvVbdVinvVbd <- VinvVadVinvVbd <-
    XtVinvVbdVinvVbdVinvX <- XtVinvVadVinvVbdVinvX <- list()

    Q.inv <- matrix(0, nrow=p, ncol=p)

    tr.VinvVad  <- tr.VinvVbd   <- tr.VinvVadVinvVad  <-
    tr.VinvVbdVinvVbd <- tr.VinvVadVinvVbd  <- ytVinvX <-
    ytVinvVadVinvy  <- SumXtVinvVadVinvX <- ytVinvVadVinvX <-
    ytVinvVbdVinvy  <- ytVinvVbdVinvX <- SumXtVinvVbdVinvX  <- 0

### Matrix Omegad and its derivative

for(d in 1:D) {
    Omegad<-matrix(0,nrow=md[d],ncol=md[d])
    Omegad[lower.tri(Omegad)]<-rho.f^sequence((md[d]-1):1)
    Omegad<-Omegad+t(Omegad)
    diag(Omegad)<-1
    Omegad <- (1/(1-rho.f^2))*Omegad
    Vad[[d]] <- Omegad

### Derivative

    OmegadFirst<-matrix(0,nrow=md[d],ncol=md[d])
    OmegadFirst[lower.tri(OmegadFirst)]<-sequence((md[d]-1):1)*
                                        rho.f^(sequence((md[d]-1):1)-1)
    OmegadFirst<-OmegadFirst+t(OmegadFirst)
    OmegadFirst<- (1/(1-rho.f^2))*OmegadFirst
    OmegadFirst <- OmegadFirst + (2*rho.f/(1-rho.f^2))*Omegad
    Vbd[[d]] <- sigma.f*OmegadFirst
```

```
### Matrix Calculation for Scores and F

### Elements of the variance matrix
    Vd <- (sigma.f * Omegad + diag(sigma2edt[a[[d]]]))

### Inverse matrix of the variance and submatrices
    Vd.inv[[d]] <- solve(Vd)

### Product between V^-1_ed  and  y_d for all d submatrices
    Vinvyd[[d]] <- Vd.inv[[d]]%*%yd[[d]]

### Product between V^-1_ed  and  X_d for all d submatrices
    VinvXd[[d]] <- Vd.inv[[d]]%*%Xd[[d]]

### Inverse of Q. Next we calculate Q
    Q.inv <- Q.inv + t(Xd[[d]])%*%VinvXd[[d]]

### Sa
    VinvVad[[d]] <- Vd.inv[[d]]%*%Vad[[d]]
    tr.VinvVad <-  tr.VinvVad + sum(diag(VinvVad[[d]]))

    XtVinvVadVinvX[[d]]<- t(VinvXd[[d]])%*%Vad[[d]]%*%VinvXd[[d]]

    ytVinvX <- ytVinvX + t(yd[[d]])%*%VinvXd[[d]]
    ytVinvVadVinvy <- ytVinvVadVinvy
                      + t(Vinvyd[[d]])%*%Vad[[d]]%*%Vinvyd[[d]]
    ytVinvVadVinvX <- ytVinvVadVinvX
                      + t(Vinvyd[[d]])%*%Vad[[d]]%*%VinvXd[[d]]
    SumXtVinvVadVinvX <- SumXtVinvVadVinvX + XtVinvVadVinvX[[d]]

### Sb
    VinvVbd[[d]] <- Vd.inv[[d]]%*%Vbd[[d]]
    tr.VinvVbd <-  tr.VinvVbd + sum(diag(VinvVbd[[d]]))

    XtVinvVbdVinvX[[d]] <- t(VinvXd[[d]])%*%Vbd[[d]]%*%VinvXd[[d]]

    ytVinvVbdVinvy <- ytVinvVbdVinvy
                      + t(Vinvyd[[d]])%*%Vbd[[d]]%*%Vinvyd[[d]]
    ytVinvVbdVinvX <- ytVinvVbdVinvX
                      + t(Vinvyd[[d]])%*%Vbd[[d]]%*%VinvXd[[d]]
```

```
    SumXtVinvVbdVinvX <- SumXtVinvVbdVinvX + XtVinvVbdVinvX[[d]]

### Faa
    VinvVadVinvVad[[d]] <- Vd.inv[[d]]%*%Vad[[d]]%*%
                           Vd.inv[[d]]%*%Vad[[d]]
    tr.VinvVadVinvVad <- tr.VinvVadVinvVad
                         + sum(diag(VinvVadVinvVad[[d]]))

    XtVinvVadVinvVadVinvX[[d]] <- t(VinvXd[[d]])%*%Vad[[d]]%*%
                           Vd.inv[[d]]%*%Vad[[d]]%*%VinvXd[[d]]

### Fbb
    VinvVbdVinvVbd[[d]] <- Vd.inv[[d]]%*%Vbd[[d]]%*%
                           Vd.inv[[d]]%*%Vbd[[d]]
    tr.VinvVbdVinvVbd <- tr.VinvVbdVinvVbd
                         + sum(diag(VinvVbdVinvVbd[[d]]))

    XtVinvVbdVinvVbdVinvX[[d]] <- t(VinvXd[[d]])%*%Vbd[[d]]%*%
                          Vd.inv[[d]]%*%Vbd[[d]]%*%VinvXd[[d]]

### Fab
    VinvVadVinvVbd[[d]]  <- Vd.inv[[d]]%*%
                            Vad[[d]]%*%Vd.inv[[d]]%*%Vbd[[d]]
    tr.VinvVadVinvVbd <- tr.VinvVadVinvVbd
                         + sum(diag(VinvVadVinvVbd[[d]]))

    XtVinvVadVinvVbdVinvX[[d]] <- t(VinvXd[[d]])%*%Vad[[d]]%*%
                          Vd.inv[[d]]%*%Vbd[[d]]%*%VinvXd[[d]]
        }

Q <- solve(Q.inv)


tr.XtVinvVadVinvXQ <- tr.XtVinvVbdVinvXQ  <-
tr.XtVinvVadVinvVadVinvXQ <- tr.XtVinvVbdVinvVbdVinvXQ <-
tr.XtVinvVadVinvVbdVinvXQ <- tr.XtVinvVbdVinvVbdVinvXQ <-
XtVinvVadVinvXQ <- XtVinvVbdVinvXQ <- 0


for(d in 1:D){
```

```
tr.XtVinvVadVinvXQ <- tr.XtVinvVadVinvXQ
                      + sum(diag(XtVinvVadVinvX[[d]]%*%Q))
tr.XtVinvVbdVinvXQ <- tr.XtVinvVbdVinvXQ
                      + sum(diag(XtVinvVbdVinvX[[d]]%*%Q))
tr.XtVinvVadVinvVadVinvXQ <- tr.XtVinvVadVinvVadVinvXQ
                      + sum(diag(XtVinvVadVinvVadVinvX[[d]]%*%Q))
XtVinvVadVinvXQ  <- XtVinvVadVinvXQ
                     + XtVinvVadVinvX[[d]]%*%Q
 tr.XtVinvVbdVinvVbdVinvXQ <- tr.XtVinvVbdVinvVbdVinvXQ
                       + sum(diag(XtVinvVbdVinvVbdVinvX[[d]]%*%Q))
 XtVinvVbdVinvXQ  <- XtVinvVbdVinvXQ + XtVinvVbdVinvX[[d]]%*%Q
 tr.XtVinvVadVinvVbdVinvXQ <- tr.XtVinvVadVinvVbdVinvXQ
                       + sum(diag(XtVinvVadVinvVbdVinvX[[d]]%*%Q))
    }

 tr.XtVinvVadVinvXQXtVinvVadVinvXQ <- sum(diag(XtVinvVadVinvXQ%*%
                                      XtVinvVadVinvXQ))
 tr.XtVinvVbdVinvXQXtVinvVbdVinvXQ <- sum(diag(XtVinvVbdVinvXQ%*%
                                      XtVinvVbdVinvXQ))
 tr.XtVinvVadVinvXQXtVinvVbdVinvXQ <- sum(diag(XtVinvVadVinvXQ%*%
                                      XtVinvVbdVinvXQ))

 tr.PVa <- tr.VinvVad - tr.XtVinvVadVinvXQ
 tr.PVb <- tr.VinvVbd - tr.XtVinvVbdVinvXQ
 tr.PVaPVa  <- tr.VinvVadVinvVad - 2*tr.XtVinvVadVinvVadVinvXQ
               + tr.XtVinvVadVinvXQXtVinvVadVinvXQ
 tr.PVbPVb  <- tr.VinvVbdVinvVbd - 2*tr.XtVinvVbdVinvVbdVinvXQ
               + tr.XtVinvVbdVinvXQXtVinvVbdVinvXQ
 tr.PVaPVb  <- tr.VinvVadVinvVbd - 2*tr.XtVinvVadVinvVbdVinvXQ
               + tr.XtVinvVadVinvXQXtVinvVbdVinvXQ

 ytPVaPy  <- ytVinvVadVinvy - ytVinvVadVinvX%*%Q%*%t(ytVinvX)
             - ytVinvX%*%Q%*%t(ytVinvVadVinvX) + ytVinvX%*%Q%*%
             SumXtVinvVadVinvX%*%Q%*%t(ytVinvX)

 ytPVbPy  <- ytVinvVbdVinvy - ytVinvVbdVinvX%*%Q%*%t(ytVinvX)
             - ytVinvX%*%Q%*%t(ytVinvVbdVinvX) + ytVinvX%*%Q%*%
             SumXtVinvVbdVinvX%*%Q%*%t(ytVinvX)
```

```
   ### Scores and Fisher information matrix

   Sa <- -0.5*tr.PVa + 0.5*ytPVaPy
   Sb <- -0.5*tr.PVb + 0.5*ytPVbPy
   Faa  <- 0.5*tr.PVaPVa
   Fbb  <- 0.5*tr.PVbPVb
   Fab  <- 0.5*tr.PVaPVb

   Ssig <- c(Sa,Sb)
   Fsig <- matrix(c(Faa,Fab,Fab,Fbb),ncol=2)

   ### Fisher-Scoring Algorithm

   Fsig.inv <- solve(Fsig)
   dif <- Fsig.inv%*%Ssig

   theta.f <- theta.f + dif

   ###print(rho.f)
   rho.f <- theta.f[2,1]

   ###print(sigma.f)
   sigma.f <- theta.f[1,1]


   ### output3 <- data.frame(ITER, Sa, Sb, Faa, Fbb, Fab)
   ### output3 <- as.data.frame(t(output3))
   ### write.table(output3, file="Output3.txt", sep="\t")

   ### Stopping criterion
   if(abs(dif[1,1])<0.000001 && abs(dif[2,1])<0.000001)
      break
    }

   return(list(as.vector(theta.f), Fsig, Q))
}
```

10.2 R code of BETA.U.area.autocorr

The R code of the function **BETA.U.area.autocorr** is listed bellow.

```
#########################################################################
###
###          Area level model with time correlated effects
###                          SAMPLE project
###
### Author: Maria Dolores Esteban Lefler
### File name: EstimationBETAautocorr.R
### Updated: November 25th, 2009
###
#########################################################################

BETA.U.area.autocorr <- function(X,ydt,D,md,sigma2edt,sigmau,rho) {

    p <- ncol(X)
    a <- list(1:md[1])
    mdcum <- cumsum(md)
    for(d in 2:D)
        a[[d]] <- (mdcum[d-1]+1):mdcum[d]

    yd <- Xd <- Vd <- Vd.inv <- list()
    Q.inv <- matrix(0, nrow=p, ncol=p)
    XVy <- 0
    for(d in 1:D) {
        yd[[d]] <- ydt[a[[d]]]
        Xd[[d]] <- X[a[[d]],]
        Omegad<-matrix(0,nrow=md[d],ncol=md[d])
              Omegad[lower.tri(Omegad)]<-rho^sequence((md[d]-1):1)
              Omegad<-Omegad+t(Omegad)
              diag(Omegad)<-1
              Omegad<- (1/(1-rho^2))*Omegad

      ### Elements of the variance matrix
        Vd <- (sigmau * Omegad + diag(sigma2edt[a[[d]]]))

      ### Inverse matrix of the variance and d submatrices
        Vd.inv[[d]] <- solve(Vd)

      ### Inverse of Q. Next we calculate Q
        Q.inv <- Q.inv + t(Xd[[d]])%*%Vd.inv[[d]]%*%Xd[[d]]
```

```
      ### Product between X^t_d, V^-1_d and y_d for all d submatrices
        XVy <- XVy + t(Xd[[d]])%*%Vd.inv[[d]]%*%yd[[d]]
    }
    Q <- solve(Q.inv)

    beta <- Q%*%XVy

    u <- list()
    for(d in 1:D)
        u[[d]] <- sigmau*(Omegad%*%Vd.inv[[d]]%*%(yd[[d]]
                  - Xd[[d]]%*%beta))
    u <- as.matrix(unlist(u))

    return(rbind(beta,u))
}
```

10.3 R code of mse.area.autocorr

The R code of the function **mse.area.autocorr** is listed bellow.

```
#####################################################################
###
###       Area level model with time correlated effects
###                     SAMPLE project
###
### Author: Maria Dolores Esteban Lefler
### File name: EstimationMSEautocorr.R
### Updated: November 25th, 2009
###
#####################################################################


mse.area.autocorr <- function(X,D,md,sigma2edt,sigmau,rho,Fsig) {

    p <- ncol(X)
    a <- list(1:md[1])
    mdcum <- cumsum(md)
    for(d in 2:D)
        a[[d]] <- (mdcum[d-1]+1):mdcum[d]
```

```
    Xd <- Vd.inv <- Sed.inv <- Omegad  <- OmegadFirst <- VinvOmega <-
    OmegaVinvOmega <- VinvOmegaFirst  <- OmegaVinvOmegaFirst <-
    OmegaFirstVinvOmegaFirst <- SinvXd <- OmegaVinvOmegadSinvXd <-
    g1.a  <- g2.a  <- q11  <- q12 <- q22 <- list()

    Q.inv <- matrix(0, nrow=p, ncol=p)

    for(d in 1:D) {

### Matrix Omegad and its derivative

    Omegad[[d]]<-matrix(0,nrow=md[d],ncol=md[d])
    Omegad[[d]][lower.tri(Omegad[[d]])]<-rho^sequence((md[d]-1):1)
    Omegad[[d]]<-Omegad[[d]]+t(Omegad[[d]])
    diag(Omegad[[d]])<-1
    Omegad[[d]] <- (1/(1-rho^2))*Omegad[[d]]

### Derivative

    OmegadFirst[[d]]<-matrix(0,nrow=md[d],ncol=md[d])
    OmegadFirst[[d]][lower.tri(OmegadFirst[[d]])] <-
    sequence((md[d]-1):1)*rho^(sequence((md[d]-1):1)-1)
    OmegadFirst[[d]]<-OmegadFirst[[d]]+t(OmegadFirst[[d]])
    OmegadFirst[[d]]<- (1/(1-rho^2))*OmegadFirst[[d]]
    OmegadFirst[[d]] <- OmegadFirst[[d]]
                        + (2*rho/(1-rho^2))*Omegad[[d]]

    Xd[[d]] <- X[a[[d]],]

### Matrix Sigma_e
    Sed <- diag(sigma2edt[a[[d]]])

### Matrix of variance
     Vd <- (sigmau * Omegad[[d]] + Sed)

### Inverse matrix of the variance and d submatrices
    Vd.inv[[d]] <- solve(Vd)

### Inverse matrix of Sigma_ed in all d submatrices
    Sed.inv[[d]] <- solve(Sed)
```

```
### Product between V^-1_d and Omega
    VinvOmega[[d]] <-  Vd.inv[[d]]%*%  Omegad[[d]]
    VinvOmegaFirst[[d]] <-  Vd.inv[[d]]%*% OmegadFirst[[d]]

### Product between Omega, V^-1_d and Omega
       OmegaVinvOmega[[d]] <- t(VinvOmega[[d]])%*%Omegad[[d]]

### Product between Omega, V^-1_d and OmegadFirst
    OmegaVinvOmegaFirst[[d]] <- Omegad[[d]] %*% Vd.inv[[d]] %*%
                                OmegadFirst[[d]]

### Product between OmegadFirst, V^-1_d and OmegadFirst
    OmegaFirstVinvOmegaFirst[[d]] <- OmegadFirst[[d]]%*%
                                     Vd.inv[[d]]%*%OmegadFirst[[d]]

### Product between Sigma^-1_ed and X_d for all d submatrices
     SinvXd[[d]] <- Sed.inv[[d]]%*%Xd[[d]]

### Product between Omegad, V^-1_d, Omegad, Sigma^-1_ed
### and X_d for d submatrices
    OmegaVinvOmegadSinvXd[[d]] <- OmegaVinvOmega[[d]]%*%SinvXd[[d]]

### First part of g1 (the second is its transpose)
    g1.a[[d]] <- sigmau * Omegad[[d]] - sigmau^2 *OmegaVinvOmega[[d]]

### First part of g2 (the second is its transpose)
g2.a[[d]] <- Xd[[d]] - sigmau*Omegad[[d]]%*%SinvXd[[d]]
           + sigmau^2* OmegaVinvOmegadSinvXd[[d]]

q11[[d]] <- OmegaVinvOmega[[d]] - 2*sigmau*OmegaVinvOmega[[d]]%*%
          VinvOmega[[d]] + sigmau^2*OmegaVinvOmega[[d]] %*%
          Vd.inv[[d]]%*%OmegaVinvOmega[[d]]

q12[[d]] <- sigmau*OmegaVinvOmegaFirst[[d]] - sigmau^2*
          OmegaVinvOmegaFirst[[d]]%*%VinvOmega[[d]]
          - sigmau^2*OmegaVinvOmega[[d]]%*%VinvOmegaFirst[[d]]
          + sigmau^3*OmegaVinvOmega[[d]]%*%VinvOmegaFirst[[d]]%*%
          VinvOmega[[d]]
q22[[d]] <- sigmau^2*OmegaFirstVinvOmegaFirst[[d]]
```

```
          - 2*sigmau^3*OmegaVinvOmegaFirst[[d]]%*%
           VinvOmegaFirst[[d]] + sigmau^4* OmegaVinvOmegaFirst[[d]]%*%
           Vd.inv[[d]]%*%t(OmegaVinvOmegaFirst[[d]])

### Inverse of Q. Next we calculate Q
 Q.inv <- Q.inv + t(Xd[[d]])%*%Vd.inv[[d]]%*%Xd[[d]]
 }

   Q <- solve(Q.inv)

   ### Calculation of g

   g1 <- g2 <- g3 <- list()
   for(d in 1:D){
       g1[[d]] <- diag(g1.a[[d]])
       g2[[d]] <- diag(g2.a[[d]]%*%Q%*%t(g2.a[[d]]))
       q11[[d]] <- diag(q11[[d]])
       q12[[d]] <- diag(q12[[d]])
       q22[[d]] <- diag(q22[[d]])
       }

   for(d in 1:D){
       g3[[d]] <- vector()
          for(t in 1:md[d]){
              g3[[d]][t] <- sum(diag(matrix(c(q11[[d]][t],
                            rep(q12[[d]][t],2),q22[[d]][t]),
                            nrow=2)%*%solve(Fsig)))
                }
    }
   g1 <- unlist(g1)
   g2 <- unlist(g2)
   g3 <- unlist(g3)

   return(g1+g2+2*g3)
}
```

10.4 R code of Interval.autocorr

The R code of the function **Interval.autocorr** is listed bellow.

```
#################################################################
###
###       Area level model with time correlated effects
###                     SAMPLE project
###
### Author: Maria Dolores Esteban Lefler
### File name: ICautocorr.R
### Updated: November 25th, 2009
###
#################################################################

Interval.autocorr <- function(fit, conf=0.95) {
    alfa <- 1-conf
    k <- 1-alfa/2
    z <- qnorm(k)

    Finv <- solve(fit[[2]])

    sigma.std.err <- z*sqrt(Finv[1,1])
    rho.std.err <- z*sqrt(Finv[2,2])
    beta.std.err <- z*sqrt(as.vector(diag(fit[[3]])))

    return( list(sigma.std.err, rho.std.err, beta.std.err) )
}
```

Chapter 11

Appendix 4: R code for M-quantile small area estimators of the mean

11.1 R code of mq.sae

The R code of the function **mq.sae** is listed bellow.

```
#####################################################################
###
###             M-quantile estimators for the mean
###                               SAMPLE project
###
### Authors: N. Salvati, N.Tzavidis, C. Giusti,
###          S. Marchetti and M. Pratesi
### File name: mq.sae.R
### Updated: March 17th, 2010
###
#####################################################################

library(MASS)

#M-quantile function
QRLM <- function (x, y, case.weights = rep(1, nrow(x)),
var.weights = rep(1, nrow(x)), w = rep(1, nrow(x)), init = "ls",
psi = psi.huber, scale.est = c("MAD", "Huber", "proposal 2"),
k2 = 1.345, method = c("M", "MM"), maxit = 20, acc = 1e-04,
test.vec = "resid", q = 0.5)
{
```

```
irls.delta <- function(old,new)
              sqrt(sum((old-new)^2)/max(1e-20,sum(old^2)))
irls.rrxwr <- function(x, w, r) {
w <- sqrt(w)
max(abs((matrix(r*w,1,length(r))%*% x)/sqrt(matrix(w,1,length(r)) %*%
%*% (x^2))))/sqrt(sum(w*r^2))
}
method <- match.arg(method)
nmx <- deparse(substitute(x))
if (is.null(dim(x))) {
x <- as.matrix(x)
colnames(x) <- nmx
}
else x <- as.matrix(x)
if (is.null(colnames(x)))
colnames(x) <- paste("X", seq(ncol(x)), sep = "")
if (qr(x)$rank < ncol(x))
stop("x is singular: singular fits are not implemented in rlm")
if (!(any(test.vec == c("resid", "coef", "w", "NULL")) ||
is.null(test.vec)))
stop("invalid testvec")
if (length(var.weights) != nrow(x))
stop("Length of var.weights must equal number of observations")
if (any(var.weights < 0))
stop("Negative var.weights value")
if (length(case.weights) != nrow(x))
stop("Length of case.weights must equal number of observations")
w <- (w * case.weights)/var.weights
if (method == "M") {
scale.est <- match.arg(scale.est)
if (!is.function(psi))
psi <- get(psi, mode = "function")
arguments <- list(...)
if (length(arguments)) {
pm <- pmatch(names(arguments), names(formals(psi)), nomatch = 0)
if (any(pm == 0))
warning(paste("some of ... do not match"))
pm <- names(arguments)[pm > 0]
formals(psi)[pm] <- unlist(arguments[pm])
}
```

```
if (is.character(init)) {
if (init == "ls")
temp <- lm.wfit(x, y, w, method = "qr")
else if (init == "lts")
temp <- lqs.default(x, y, intercept = FALSE, nsamp = 200)
else stop("init method is unknown")
coef <- temp$coef
resid <- temp$resid
}
else {
if (is.list(init))
coef <- init$coef
else coef <- init
resid <- y - x %*% coef
}
}
else if (method == "MM") {
scale.est <- "MM"
temp <- lqs.default(x, y, intercept=FALSE, method="S", k0=1.548)
coef <- temp$coef
resid <- temp$resid
psi <- psi.bisquare
if (length(arguments <- list(...)))
if (match("c", names(arguments), nomatch = FALSE)) {
c0 <- arguments$c
if (c0 > 1.548) {
psi$c <- c0
}
else warning("c must be at least 1.548 and has been ignored")
}
scale <- temp$scale
}
else stop("method is unknown")
done <- FALSE
conv <- NULL
n1 <- nrow(x) - ncol(x)
if (scale.est != "MM")
scale <- mad(resid/sqrt(var.weights), 0)
theta <- 2 * pnorm(k2) - 1
gamma <- theta + k2^2 * (1 - theta) - 2 * k2 * dnorm(k2)
```

```
qest <- matrix(0, nrow = ncol(x), ncol = length(q))
qwt <- matrix(0, nrow = nrow(x), ncol = length(q))
qfit <- matrix(0, nrow = nrow(x), ncol = length(q))
qres <- matrix(0, nrow = nrow(x), ncol = length(q))
for(i in 1:length(q)) {
for (iiter in 1:maxit) {
if (!is.null(test.vec))
testpv <- get(test.vec)
if (scale.est != "MM") {
if (scale.est == "MAD")
scale <- median(abs(resid/sqrt(var.weights)))/0.6745
else scale <- sqrt(sum(pmin(resid^2/var.weights,
              (k2*scale)^2))/(n1*gamma))
if (scale == 0) {
done <- TRUE
break
}
}
w <- psi(resid/(scale * sqrt(var.weights))) * case.weights
ww <- 2 * (1 - q[i]) * w
ww[resid > 0] <- 2 * q[i] * w[resid > 0]
w <- ww
temp <- lm.wfit(x, y, w, method = "qr")
coef <- temp$coef
resid <- temp$residuals
if (!is.null(test.vec))
convi <- irls.delta(testpv, get(test.vec))
else convi <- irls.rrxwr(x, wmod, resid)
conv <- c(conv, convi)
done <- (convi <= acc)
if (done)
break
}
if (!done)
warning(paste("rlm failed to converge in",maxit,"steps at q=",q[i]))
qest[, i] <- coef
qwt[, i] <- w
qfit[, i] <- temp$fitted.values
qres[,i] <- resid
}
```

```
list(fitted.values = qfit, residuals = qres, q.values = q,
q.weights = qwt, coefficients = qest)
}

# COMPUTE THE QUANTILE ORDER

# COMPUTING OF THE QUANTILE-ORDERS
"zerovalinter"<-function(y, x)
{
        if(min(y) > 0) {
                xmin <- x[y == min(y)]
                if(length(xmin) > 0)
                        xmin <- xmin[length(xmin)]
                xzero <- xmin
        }

else {
                if(max(y) < 0) {
                        xmin <- x[y == max(y)]
                        if(length(xmin) > 0)
                                xmin <- xmin[1]
                        xzero <- xmin
                }
                else {
                        y1 <- min(y[y > 0])
                        if(length(y1) > 0)
                                y1 <- y1[length(y1)]
                        y2 <- max(y[y < 0])
                        if(length(y2) > 0)
                                y2 <- y2[1]
                        x1 <- x[y == y1]
                        if(length(x1) > 0)
                                x1 <- x1[length(x1)]
                        x2 <- x[y == y2]
                        if(length(x2) > 0)
                                x2 <- x2[1]
                        xzero <- (x2 * y1 - x1 * y2)/(y1 - y2)
                        xmin <- x1
                        if(abs(y2) < y1)
                                xmin <- x2
```

```
                }
        }
        resu <-  xzero
        resu
}

# Function for Finding the Quantile Orders by Linear Interpolation
# Assumes that "zerovalinter" function has been already loaded

"gridfitinter"<-function(y,expectile,Q)
# computing of the expectile-order of each observation of y by
# interpolation
{
nq<-length(Q)
  diff <- y %*% t(as.matrix(rep(1, nq))) - expectile
 vectordest <- apply(diff, 1, zerovalinter,Q)

#print(vectordest)
#qord<-list(ord=c(vectordest))
#qord
}


#y: study variable
#x: set of covariates without the intercept for sampled units
#regioncode.s: area code for sampled units
#x.r: set of covariates for out of sample units
#regioncode.r: area code for out of sample units
#p size of x +1 (intercept)


mq.sae=function(y,x,regioncode.s,m,p,x.outs,regioncode.r,
tol.value=0.0001,maxit.value=100,k.value=1.345)

{

MQE<-c(rep(0,m))
MQNAIVE<-c(rep(0,m))
```

```
datanew=cbind(y,x,regioncode.s)

ni=as.numeric(table(regioncode.s))

sample.sizer<-as.numeric(table(regioncode.r))

Ni=sample.sizer+ni


N<-sum(Ni)
n<-sum(ni)

x=matrix(x,n,p-1)
x.r=matrix(x.outs,(N-n),p-1)
x.t=rbind(x,x.r)
x.c<-rep(1,n)
x.design<-cbind(x.c,x)

p=ncol(x.design)

ob<-QRLM(x.design, y,q=sort(c(seq(0.006,0.99,0.045),0.5,0.994,
0.01,0.02,0.96,0.98)),k = k.value,maxit=maxit.value,acc=tol.value)

qo<-matrix(c(gridfitinter(y,ob$fitted.values,ob$q.values)),
           nrow=n,ncol=1)

qmat<-matrix(c(qo,regioncode.s),nrow=sum(ni),ncol=2)

mqo<-aggregate(qmat[,1],list(d2=qmat[,2]),mean)[,2]

uar<-sort(unique(regioncode.s))
saq<-matrix(c(mqo,uar),nrow=m,ncol=2)

saq<-rbind(saq,c(0.5,9999))

beta.stored=matrix(0,m,2)
res.s=NULL
tttmp1<-NULL
ci<-array(rep(0,n*m),dim=c(n,m,1))
ci1<-array(rep(0,n*m),dim=c(n,m,1))
```

```
prs<-NULL
prr<-NULL
wi<-matrix(0,n,m)

for(i in 1:m){

ob1<-QRLM(x.design,y,q=mqo[i],psi=psi.huber,k = k.value,
maxit=maxit.value,acc=tol.value)

wd<-diag(c(ob1$q.weights))

# Regional parameters from multiquantile model

coef<-matrix(c(t(ob1$coefficients)),nrow=1,ncol=p)
# need to be ordered by area

coef<-t(coef)

meat<-wd%*%x.design%*%solve(t(x.design)%*%wd%*%x.design)

x1<-c(rep(1,(Ni[i]-ni[i])))

ir<-rep(0,n)

ir[regioncode.s==uar[i]]<-1

rj1<-sample.sizer[i]

r=NULL

for (kk in 1:(p-1))
{
r<-c(r,sum(x.r[,kk][regioncode.r==uar[i]]))}

r=c(rj1,r)


sj1<-sum(rep(1,ni[i]))

tss=NULL
```

```
for (kk in 1:(p-1))
{
tss<-c(tss,sum(x[,kk][regioncode.s==uar[i]]))}

tss<-c(sj1,tss)

w.welsh<-((Ni[i])/(ni[i]))*ir+meat%*%(r-((Ni[i]-ni[i])/ni[i])*tss)

MQE[i]<-sum(w.welsh*y)/sum(w.welsh)
y.i<-y[regioncode.s==uar[i]]

y.pred.s<-cbind(1,x[regioncode.s==uar[i],])%*%coef
residual<-y.i-y.pred.s

tttmp1[i]<-(sample.sizer[i]/ni[i])*sum(residual)

prs<-c(prs,y.pred.s)
res.s<-c(res.s,residual)
y.pred<-cbind(1,x.r[regioncode.r==uar[i],])%*%coef
prr<-c(prr,y.pred)

data<-cbind(regioncode.s,w.welsh)

for (kk in 1:n){

if (data[kk,1]==uar[i]) ci[kk,i,1]<-data[kk,2]-1
else if (data[kk,1]!=uar[i]) ci[kk,i,1]<-data[kk,2]  }

MQNAIVE[i]<-(1/Ni[i])*as.real(sum(y.i)+sum(y.pred))

#f<-(sample.sizer[i])/ni[i]
ai<-r
bi<-ai%*%solve(t(x.design)%*%wd%*%x.design)%*%t(x.design)%*%wd
bi<-c(bi)
wi[,i]<-c(ir+bi)

datanaive<-cbind(regioncode.s,wi[,i])
for (kk in 1:n){
if (datanaive[kk,1]==uar[i]) ci1[kk,i,1]<-datanaive[kk,2]-1
```

```
else if (datanaive[kk,1]!=uar[i]) ci1[kk,i,1]<-datanaive[kk,2]  }

}

res.d=res.s^2

res.d<-cbind(res.d,regioncode.s,ci[,,1])

v<-NULL
for (oo in 1:m){
v[oo]<-1/Ni[oo]^2*(sum((res.d[,(oo+2)][res.d[,2]==uar[oo]]^2+
+(sample.sizer[oo])/ni[oo])*res.d[,1][res.d[,2]==uar[oo]])+
+sum(res.d[,(oo+2)][res.d[,2]!=uar[oo]]^2*res.d[,1][res.d[,2]!=uar[oo]]))

}

res.d1<-cbind(res.s^2,regioncode.s,ci1[,,1])
v1<-NULL
bias<-NULL
mse<-NULL
for (oo in 1:m){
v1[oo]<-1/Ni[oo]^2*(sum((res.d1[,(oo+2)][res.d1[,2]==uar[oo]]^2+
+(sample.sizer[oo])/n)*res.d1[,1][res.d1[,2]==uar[oo]])+
+sum(res.d1[,(oo+2)][res.d1[,2]!=uar[oo]]^2*res.d1[,1]
     [res.d1[,2]!=uar[oo]]))

bias[oo]<-(1/Ni[oo])*(sum(wi[,oo]*prs)-sum(c(prs[regioncode.s==uar[oo]],
prr[regioncode.r==uar[oo]])))

mse[oo]<-v1[oo]+(bias[oo])^2
}

list(mq.cd=MQE,mq.naive=MQNAIVE,mse.cd=v,mse.naive=mse,code.area=uar)
}
```

Chapter 12

Appendix 5: R code for M-quantile Geographically Weighted Regression

12.1 R code of mqgwr.sae

The R code of the function **mqgwr.sae** is listed bellow.

```
#################################################################
###
###             M-quantile GWR estimators for the mean
###                               SAMPLE project
###
### Authors: N. Salvati, N.Tzavidis, C. Giusti,
###          S. Marchetti and M. Pratesi
### File name: mqgwr.sae.R
### Updated: March 17th, 2010
###
#################################################################

library(MASS)
library(nlme)
library(sp)
library(spgwr)

#M-estimator with GWR weights

QRLM <- function (x, y, case.weights = rep(1, nrow(x)),
var.weights = rep(1, nrow(x)), ..., w = rep(1, nrow(x)),
```

```
init="ls", psi=psi.huber, scale.est=c("MAD", "Huber", "proposal 2"),
k2 = 1.345, method = c("M", "MM"), maxit = 20, acc = 1e-04,
test.vec = "resid", q = 0.5,w1)

{
irls.delta <- function(old,new)sqrt(sum((old-new)^2)/
                        max(1e-20,sum(old^2)))
irls.rrxwr <- function(x, w, r) {
w <- sqrt(w)
max(abs((matrix(r*w,1,length(r))%*% x)/sqrt(matrix(w,1,length(r)) %*%
%*% (x^2))))/sqrt(sum(w*r^2))
}
method <- match.arg(method)
nmx <- deparse(substitute(x))
if (is.null(dim(x))) {
x <- as.matrix(x)
colnames(x) <- nmx
}
else x <- as.matrix(x)
if (is.null(colnames(x)))
colnames(x) <- paste("X", seq(ncol(x)), sep = "")
if (qr(x)$rank < ncol(x))
stop("x is singular: singular fits are not implemented in rlm")
if (!(any(test.vec == c("resid", "coef", "w", "NULL")) ||
    is.null(test.vec)))
stop("invalid testvec")
if (length(var.weights) != nrow(x))
stop("Length of var.weights must equal number of observations")
if (any(var.weights < 0))
stop("Negative var.weights value")
if (length(case.weights) != nrow(x))
stop("Length of case.weights must equal number of observations")
w <- (w * case.weights)/var.weights
if (method == "M") {
scale.est <- match.arg(scale.est)
if (!is.function(psi))
psi <- get(psi, mode = "function")
arguments <- list(...)
if (length(arguments)) {
pm <- pmatch(names(arguments), names(formals(psi)), nomatch = 0)
```

```
if (any(pm == 0))
warning(paste("some of ... do not match"))
pm <- names(arguments)[pm > 0]
formals(psi)[pm] <- unlist(arguments[pm])
}
if (is.character(init)) {
if (init == "ls")
temp <- lm.wfit(x, y, w, method = "qr")
else if (init == "lts")
temp <- lqs.default(x, y, intercept = FALSE, nsamp = 200)
else stop("init method is unknown")
coef <- temp$coef
resid <- temp$resid
}
else {
if (is.list(init))
coef <- init$coef
else coef <- init
resid <- y - x %*% coef
}
}
else if (method == "MM") {
scale.est <- "MM"
temp <- lqs.default(x, y, intercept=FALSE, method="S", k0=1.548)
coef <- temp$coef
resid <- temp$resid
psi <- psi.bisquare
if (length(arguments <- list(...)))
if (match("c", names(arguments), nomatch = FALSE)) {
c0 <- arguments$c
if (c0 > 1.548) {
psi$c <- c0
}
else warning("c must be at least 1.548 and has been ignored")
}
scale <- temp$scale
}
else stop("method is unknown")
done <- FALSE
conv <- NULL
```

```
n1 <- nrow(x) - ncol(x)
if (scale.est != "MM")
scale <- mad(resid/sqrt(var.weights), 0)
theta <- 2 * pnorm(k2) - 1
gamma <- theta + k2^2 * (1 - theta) - 2 * k2 * dnorm(k2)
qest <- matrix(0, nrow = ncol(x), ncol = length(q))
qwt <- matrix(0, nrow = nrow(x), ncol = length(q))
qfit <- matrix(0, nrow = nrow(x), ncol = length(q))
qres <- matrix(0, nrow = nrow(x), ncol = length(q))
for(i in 1:length(q)) {
for (iiter in 1:maxit) {
if (!is.null(test.vec))
testpv <- get(test.vec)
if (scale.est != "MM") {
if (scale.est == "MAD")
scale <- median(abs(resid/sqrt(var.weights)))/0.6745
else scale <- sqrt(sum(pmin(resid^2/var.weights,(k2*scale)^2))/
              (n1*gamma))
if (scale == 0) {
done <- TRUE
break
}
}
w <- psi(resid/(scale * sqrt(var.weights))) * case.weights
ww <- 2 * (1 - q[i]) * w
ww[resid > 0] <- 2 * q[i] * w[resid > 0]
w <- ww*diag(w1)
temp <- lm.wfit(x, y, w, method = "qr")
coef <- temp$coef
resid <- temp$residuals
if (!is.null(test.vec))
convi <- irls.delta(testpv, get(test.vec))
else convi <- irls.rrxwr(x, wmod, resid)
conv <- c(conv, convi)
done <- (convi <= acc)
if (done)
break
}
if (!done)
warning(paste("rlm failed to converge in",maxit,"steps at q = ",q[i]))
```

```
qest[, i] <- coef
qwt[, i] <- w
qfit[, i] <- temp$fitted.values
qres[,i] <- resid
}
list(fitted.values = qfit, residuals = qres, q.values = q,
q.weights = qwt, coefficients = qest)
}

# COMPUTE THE QUANTILE ORDER
#
zerovalinter<-function(y, x)
{
       if(min(y) > 0) {
                xmin <- x[y == min(y)]
                if(length(xmin) > 0)
                        xmin <- xmin[length(xmin)]
                xzero <- xmin
       }
else {
                if(max(y) < 0) {
                        xmin <- x[y == max(y)]
                        if(length(xmin) > 0)
                                xmin <- xmin[1]
                        xzero <- xmin
                }
                else {
                        y1 <- min(y[y > 0])
                        if(length(y1) > 0)
                                y1 <- y1[length(y1)]
                        y2 <- max(y[y < 0])
                        if(length(y2) > 0)
                                y2 <- y2[1]
                        x1 <- x[y == y1]
                        if(length(x1) > 0)
                                x1 <- x1[length(x1)]
                        x2 <- x[y == y2]
                        if(length(x2) > 0)
                                x2 <- x2[1]
                        xzero <- (x2 * y1 - x1 * y2)/(y1 - y2)
```

```
                        xmin <- x1
                        if(abs(y2) < y1)
                                xmin <- x2
                }
        }
        resu <-  xzero
        resu
}
#
# Function for Finding the Quantile Orders by Linear Interpolation
#
gridfitinter<-function(y,expectile,Q)
# computing of the expectile-order of each observation of y by
# interpolation
{
nq<-length(Q)
 diff <- y %*% t(as.matrix(rep(1, nq))) - expectile
 vectordest <- apply(diff, 1, zerovalinter,Q)
}

##M-estimator original

QRLM1 <- function (x, y, case.weights = rep(1, nrow(x)),
var.weights = rep(1, nrow(x)), ..., w = rep(1, nrow(x)), init = "ls",
psi = psi.huber, scale.est = c("MAD", "Huber", "proposal 2"),
k2 = 1.345, method = c("M", "MM"), maxit = 20, acc = 1e-04,
test.vec = "resid", q = 0.5)
{
irls.delta <- function(old,new)sqrt(sum((old-new)^2)/
               max(1e-20,sum(old^2)))
irls.rrxwr <- function(x, w, r) {
w <- sqrt(w)
max(abs((matrix(r*w,1,length(r)) %*% x)/sqrt(matrix(w,1,length(r)) %*%
 %*% (x^2))))/sqrt(sum(w*r^2))
}
method <- match.arg(method)
nmx <- deparse(substitute(x))
if (is.null(dim(x))) {
x <- as.matrix(x)
colnames(x) <- nmx
```

```
}
else x <- as.matrix(x)
if (is.null(colnames(x)))
colnames(x) <- paste("X", seq(ncol(x)), sep = "")
if (qr(x)$rank < ncol(x))
stop("x is singular: singular fits are not implemented in rlm")
if (!(any(test.vec == c("resid", "coef", "w", "NULL")) ||
      is.null(test.vec)))
stop("invalid testvec")
if (length(var.weights) != nrow(x))
stop("Length of var.weights must equal number of observations")
if (any(var.weights < 0))
stop("Negative var.weights value")
if (length(case.weights) != nrow(x))
stop("Length of case.weights must equal number of observations")
w <- (w * case.weights)/var.weights
if (method == "M") {
scale.est <- match.arg(scale.est)
if (!is.function(psi))
psi <- get(psi, mode = "function")
arguments <- list(...)
if (length(arguments)) {
pm <- pmatch(names(arguments), names(formals(psi)), nomatch = 0)
if (any(pm == 0))
warning(paste("some of ... do not match"))
pm <- names(arguments)[pm > 0]
formals(psi)[pm] <- unlist(arguments[pm])
}
if (is.character(init)) {
if (init == "ls")
temp <- lm.wfit(x, y, w, method = "qr")
else if (init == "lts")
temp <- lqs.default(x, y, intercept = FALSE, nsamp = 200)
else stop("init method is unknown")
coef <- temp$coef
resid <- temp$resid
}
else {
if (is.list(init))
coef <- init$coef
```

```
else coef <- init
resid <- y - x %*% coef
}
}
else if (method == "MM") {
scale.est <- "MM"
temp <- lqs.default(x, y, intercept=FALSE, method="S", k0=1.548)
coef <- temp$coef
resid <- temp$resid
psi <- psi.bisquare
if (length(arguments <- list(...)))
if (match("c", names(arguments), nomatch = FALSE)) {
c0 <- arguments$c
if (c0 > 1.548) {
psi$c <- c0
}
else warning("c must be at least 1.548 and has been ignored")
}
scale <- temp$scale
}
else stop("method is unknown")
done <- FALSE
conv <- NULL
n1 <- nrow(x) - ncol(x)
if (scale.est != "MM")
scale <- mad(resid/sqrt(var.weights), 0)
theta <- 2 * pnorm(k2) - 1
gamma <- theta + k2^2 * (1 - theta) - 2 * k2 * dnorm(k2)
qest <- matrix(0, nrow = ncol(x), ncol = length(q))
qwt <- matrix(0, nrow = nrow(x), ncol = length(q))
qfit <- matrix(0, nrow = nrow(x), ncol = length(q))
qres <- matrix(0, nrow = nrow(x), ncol = length(q))
for(i in 1:length(q)) {
for (iiter in 1:maxit) {
if (!is.null(test.vec))
testpv <- get(test.vec)
if (scale.est != "MM") {
if (scale.est == "MAD")
scale <- median(abs(resid/sqrt(var.weights)))/0.6745
else scale <- sqrt(sum(pmin(resid^2/var.weights,(k2*scale)^2))/
```

```
              (n1*gamma))
if (scale == 0) {
done <- TRUE
break
}
}
w <- psi(resid/(scale * sqrt(var.weights))) * case.weights
ww <- 2 * (1 - q[i]) * w
ww[resid > 0] <- 2 * q[i] * w[resid > 0]
w <- ww
temp <- lm.wfit(x, y, w, method = "qr")
coef <- temp$coef
resid <- temp$residuals
if (!is.null(test.vec))
convi <- irls.delta(testpv, get(test.vec))
else convi <- irls.rrxwr(x, wmod, resid)
conv <- c(conv, convi)
done <- (convi <= acc)
if (done)
break
}
if (!done)
warning(paste("rlm failed to converge in",maxit,"steps at q = ",q[i]))
qest[, i] <- coef
qwt[, i] <- w
qfit[, i] <- temp$fitted.values
qres[,i] <- resid
}
list(fitted.values = qfit, residuals = qres, q.values = q,
q.weights = qwt, coefficients = qest)
}

mqgwr.sae=function(x,y,m,area,lon,lat,x.r,area.r,
lon.r,lat.r,method="mqgwr",k.value=1.345, mqgwrweight=TRUE)
{

id.area<- sort(unique(area))
m<-length(id.area)

id.area.r=sort(unique(area.r))
```

```
m.r<-length(id.area.r)

tmp.cont=rep(0,m.r)
for (i in 1:m.r)
{for (j in 1:m)
{
if (id.area.r[i]==id.area[j])tmp.cont[i]=1
}

}

tmp0=which(tmp.cont==0)
id.area.out=id.area.r[tmp0]
id.area.in=id.area

ni<- rep(0,m)
for(i in 1:m)ni[i]<- sum(area==id.area.in[i])
n<- sum(ni)

ri<- rep(0,m)
for(i in 1:m)ri[i]<- sum(area.r==id.area.in[i])
r<- sum(ri)

Ni=ri+ni

RI.MQGWR_CD.Mean=rep(0,m)
if (m.r>m)RI.MQGWR_CD.Mean.out=rep(0,(m.r-m))
if (m.r==m)RI.MQGWR_CD.Mean.out=NULL
mse=rep(0,m)

# Compute the Distance Matrix

eu_dist=as.matrix(dist(cbind(as.vector(lon),as.vector(lat))))
x.design=cbind(1,x)
p=ncol(x.design)
q.value=sort(c(seq(0.002,0.99,0.045),0.5,0.994,0.01,0.02,0.96,0.98))

if (method=="mqgwr")
{
```

```
ob.trad<-QRLM1(x.design, y,maxit=100,q=q.value,k=k.value)
qo.trad<-matrix(c(gridfitinter(y,ob.trad$fitted.values,q.value)),
nrow=n,ncol=1)

if (mqgwrweight==TRUE)band=gwr.sel(y ~ 1 + x, coords=cbind(lon, lat),
gweight=gwr.gauss)

if (mqgwrweight==FALSE)band=gwr.sel(y ~ 1 + x, coords=cbind(lon, lat),
gweight=gwr.bisquare)

if (mqgwrweight==TRUE)w.sp<-gwr.gauss((eu_dist^2),band)
if (mqgwrweight==FALSE)w.sp<-gwr.bisquare((eu_dist^2),band)
    q.new=0
    for(ii in 1:n){
    w.new<-diag(w.sp[,ii])
    ob<-QRLM(x.design, y,maxit=100,q=sort(c(seq(0.002,0.99,0.045),0.5,
    0.994,0.01,0.02,0.96,0.98)),w1=w.new,k=k.value)

    qo<-matrix(c(gridfitinter(y,ob$fitted.values,ob$q.values)),
    nrow=n,ncol=1)
    q.new[ii]=as.real(qo[ii,1])
    if (is.na(q.new[ii])) q.new[ii]=as.real(qo.trad[ii,1])
    }


    qmat1<-matrix(c(q.new,area),nrow=n,ncol=2)

    mqo1=tapply(qmat1[,1],qmat1[,2],mean)

    saq<-matrix(0,nrow=m,ncol=2)

    saq[,1]=mqo1

    saq[,2]=sort(unique(qmat1[,2]))

    ci=array(rep(0,n*m),dim=c(n,m,1))

    res.s=NULL
```

```
for (i in 1:m)
   {
   pred.medr=0
   x.r.area<-matrix(cbind(1,x.r)[area.r==id.area.in[i]],ri[i],p)
   lon.gwr=lon.r[area.r==id.area.in[i]]
   lat.gwr=lat.r[area.r==id.area.in[i]]

   tmp=matrix(0,n,1)
   tmp1=matrix(0,n,1)

   for (j in 1:ri[i]){
   dbase=as.matrix(rbind(cbind(lon.gwr[j],lat.gwr[j]),
   cbind(as.vector(lon),as.vector(lat))))

   dist.r=(as.matrix(dist(dbase))[-1,1])
   if (mqgwrweight==TRUE)w.new=gwr.gauss((dist.r)^2,band)
   if (mqgwrweight==FALSE)w.new=gwr.bisquare((dist.r)^2,band)
   w.new=diag(w.new)
   ob1=QRLM(x.design, y,maxit = 100,q=c(saq[i,1]),w1=w.new,k=k.value)
   coef<-matrix(c(t(ob1$coef)),nrow=1,ncol=p)
   # need to be ordered by area

   coef<-t(coef)
   W_star=diag(c(ob1$q.weight),n,n)
   S=W_star%*%x.design%*%solve(t(x.design)%*%W_star%*%x.design)
   xir=x.r.area[j,]
   tmp=tmp+S%*%xir
   pred.medr[j]<-(x.r.area[j,]%*%coef[,1])
   }

   pred.meds=0
   sj<-matrix(x.design[area==id.area.in[i]],ni[i],p)

   lon.gwr=(lon)[area==id.area.in[i]]
   lat.gwr=(lat)[area==id.area.in[i]]
   for (j in 1:ni[i]){
   dbase=as.matrix(rbind(cbind(lon.gwr[j],lat.gwr[j]),
   cbind(as.vector(lon),as.vector(lat))))

   dist.r=(as.matrix(dist(dbase))[-1,1])
```

```
    if (mqgwrweight==TRUE)w.new=gwr.gauss((dist.r)^2,band)
    if (mqgwrweight==FALSE)w.new=gwr.bisquare((dist.r)^2,band)
    w.new=diag(w.new)
    ob1=QRLM(x.design, y,maxit = 100,q=c(saq[i,1]),w1=w.new,k=k.value)
    coef<-matrix(c(t(ob1$coef)),nrow=1,ncol=p)
    # need to be ordered by area
    coef<-t(coef)
    W_star=diag(c(ob1$q.weight),n,n)
    S=W_star%*%x.design%*%solve(t(x.design)%*%W_star%*%x.design)
    xis=sj[j,]
    tmp1=tmp1+S%*%xis
    pred.meds[j]<-(sj[j,]%*%coef[,1])
    }


   f1<-y[area==id.area.in[i]]
   res.s<-c(res.s,(f1-pred.meds))

   ir=rep(0,n)
   ir[area==id.area.in[i]]<-1
   welsh.cd=ir+ir*((ri[i]+ni[i])/ni[i])+tmp-((ri[i]+ni[i])/ni[i])*tmp1
   data<-cbind(as.vector(area),welsh.cd)

   for (kk in 1:n)
   {
   if (data[kk,1]==id.area.in[i]) ci[kk,i,1]=data[kk,2]-1
   else if (data[kk,1]!=id.area.in[i]) ci[kk,i,1]=data[kk,2]}
   RI.MQGWR_CD.Mean[i]=as.real(1/(ri[i]+ni[i])*(t(welsh.cd)%*%
   %*%as.vector(y)))
   }
   res.s=res.s^2
   res.d=cbind(res.s,as.vector(area),ci[,,1])

 for (oo in 1:m)
 {mse[oo]=(1/(ni[oo]+ri[oo])^2)*(sum((res.d[,(oo+2)][res.d[,2]==oo]^2+
 +(ri[oo])/n)*res.d[,1][res.d[,2]==oo])+
 +sum(res.d[,(oo+2)][res.d[,2]!=oo]^2*res.d[,1][res.d[,2]!=oo]))}
}

  if (method=="mqgwr-li")
```

```
  {
   if (mqgwrweight==TRUE)band=gwr.sel(y ~ 1 + x,
coords=cbind(lon, lat),gweight=gwr.gauss)

if (mqgwrweight==FALSE)band=gwr.sel(y ~ 1 + x, coords=cbind(lon, lat),
gweight=gwr.bisquare)

if (mqgwrweight==TRUE)w.sp<-gwr.gauss((eu_dist^2),band)
if (mqgwrweight==FALSE)w.sp<-gwr.bisquare((eu_dist^2),band)
     q.new=0
     n.q=length(q.value)
     fitted=matrix(0,n,n.q)
     for (qj in 1:n.q)
     {
     ob.trad<-QRLM1(x.design, y,maxit=100,q=q.value[qj])
     for(ii in 1:n){
     w.new<-(w.sp[,ii])
     err<-sum(w.new*ob.trad$q.weights*ob.trad$residuals)/
     /sum(w.new*ob.trad$q.weights)

     fitted[ii,qj]=ob.trad$fitted.values[ii]+err}
     }
     q.new<-matrix(c(gridfitinter(y,fitted,q.value)),nrow=n,ncol=1)

     qmat1<-matrix(c(q.new,area),nrow=n,ncol=2)

     mqo1=tapply(qmat1[,1],qmat1[,2],mean)

     saq<-matrix(0,nrow=m,ncol=2)

     saq[,1]=mqo1

     saq[,2]=sort(unique(qmat1[,2]))

     ci=array(rep(0,n*m),dim=c(n,m,1))

     res.s=NULL

for (i in 1:m)
```

```
{
pred.medr=0
x.r.area<-matrix(cbind(1,x.r)[area.r==id.area.in[i]],ri[i],p)
lon.gwr=lon.r[area.r==id.area.in[i]]
lat.gwr=lat.r[area.r==id.area.in[i]]

tmp=matrix(0,1,n)
tmp1=matrix(0,1,n)

ob.trad<-QRLM1(x.design, y,maxit=100,q=c(saq[i,1]))
coef<-matrix(c(t(ob.trad$coef)),nrow=1,ncol=2)
# need to be ordered by area
coef<-t(coef)
wd<-diag(c(ob.trad$q.weights))
meat<-wd%*%x.design%*%solve(t(x.design)%*%wd%*%x.design)
meat1=(diag(1,n,n)-x.design%*%solve(t(x.design)%*%wd%*%x.design)%*%
%*%t(x.design)%*%wd)

for (j in 1:ri[i]){
dbase=as.matrix(rbind(cbind(lon.gwr[j],lat.gwr[j]),
cbind(as.vector(lon),as.vector(lat))))

dist.r=(as.matrix(dist(dbase))[-1,1])
if (mqgwrweight==TRUE)w.new=gwr.gauss((dist.r)^2,band)
if (mqgwrweight==FALSE)w.new=gwr.bisquare((dist.r)^2,band)

uno=matrix(1,n,1)
err1=((t(uno)%*%(diag(c(w.new),n,n)%*%wd%*%meat1))*
*as.real(solve(t(uno)%*%diag(c(w.new),n,n)%*%wd%*%uno)))
tmp=tmp+err1
pred.medr[j]<-(x.r.area[j,]%*%coef[,1]+as.real(err1%*%matrix(y,n,1)))
}

pred.meds=0
sj<-matrix(x.design[area==id.area.in[i]],ni[i],p)

lon.gwr=(lon)[area==id.area.in[i]]
lat.gwr=(lat)[area==id.area.in[i]]
for (j in 1:ni[i]){
dbase=as.matrix(rbind(cbind(lon.gwr[j],lat.gwr[j]),
```

```
 cbind(as.vector(lon),as.vector(lat))))

 dist.r=(as.matrix(dist(dbase))[-1,1])
 if (mqgwrweight==TRUE)w.new=gwr.gauss((dist.r)^2,band)
 if (mqgwrweight==FALSE)w.new=gwr.bisquare((dist.r)^2,band)

 err2=((t(uno)%*%(diag(c(w.new),n,n)%*%wd%*%meat1))*
 *as.real(solve(t(uno)%*%diag(c(w.new),n,n)%*%wd%*%uno)))
  tmp1=tmp1+err2
 pred.meds[j]<-(sj[j,]%*%coef[,1]+as.real(err2%*%matrix(y,n,1)))
 }


f1<-y[area==id.area.in[i]]
res.s<-c(res.s,(f1-pred.meds))

sj.tot<-apply(sj,2,sum)
rj.tot<-apply(x.r.area,2,sum)

ir=rep(0,n)
ir[area==id.area.in[i]]<-1


welsh.cd=ir*((ri[i]+ni[i])/ni[i])+meat%*%
%*%(rj.tot-(ri[i]/ni[i])*sj.tot)+t(tmp)-((ri[i])/ni[i])*t(tmp1)

data<-cbind(as.vector(area),welsh.cd)

for (kk in 1:n)
{
if (data[kk,1]==id.area.in[i]) ci[kk,i,1]=data[kk,2]-1
else if (data[kk,1]!=id.area.in[i]) ci[kk,i,1]=data[kk,2]}
RI.MQGWR_CD.Mean[i]=as.real(1/(ri[i]+ni[i])*(t(welsh.cd)%*%as.vector(y)))
 }
res.s=res.s^2
res.d=cbind(res.s,as.vector(area),ci[,,1])

for (oo in 1:m)
{mse[oo]=(1/(ni[oo]+ri[oo])^2)*(sum((res.d[,(oo+2)][res.d[,2]==oo]^2+
+(ri[oo])/n)*res.d[,1][res.d[,2]==oo])+
```

```
+sum(res.d[,(oo+2)][res.d[,2]!=oo]^2*res.d[,1][res.d[,2]!=oo]))}

 }
      #out of sample area
      if (m.r>m){
       rr.sample=m.r-m
       for (i in 1:rr.sample)
       {
      Ri=sum(area.r==id.area.out[i])
       x.r.area<-matrix(cbind(1,x.r)[area.r==id.area.out[i]],Ri,p)
       pred.medr=0
       lon.gwr=lon.r[area.r==id.area.out[i]]
       lat.gwr=lat.r[area.r==id.area.out[i]]
       for (j in 1:Ri){
       dbase=as.matrix(rbind(cbind(lon.gwr[j],lat.gwr[j]),
       cbind(as.vector(lon),as.vector(lat))))
       dist.r=(as.matrix(dist(dbase))[-1,1])
       if (mqgwrweight==TRUE)w.new=gwr.gauss((dist.r)^2,band)
       if (mqgwrweight==FALSE)w.new=gwr.bisquare((dist.r)^2,band)
       w.new=diag(w.new)
       ob1=QRLM(x.design, y,maxit = 100,q=0.5,w1=w.new)
       coef<-matrix(c(t(ob1$coef)),nrow=1,ncol=2)
       # need to be ordered by area
       coef<-t(coef)
       pred.medr[j]<-(x.r.area[j,]%*%coef[,1])
       }


        RI.MQGWR_CD.Mean.out[i]<-1/(Ri)*(sum(pred.medr))}
        }

 list(Area.code.in=id.area.in,Area.code.out=id.area.out,
 Est.Mean.in=RI.MQGWR_CD.Mean,
 Est.Mean.out=RI.MQGWR_CD.Mean.out,Est.mse.in=mse)

}
```

Chapter 13

Appendix 6: R code for M-quantile CD estimators of the CDF

13.1 R code of mq.sae.quant

The R code of the function **mq.sae.quant** is listed bellow.

```
############################################################
###
###             M-quantile CD estimators for the quantiles
###                              SAMPLE project
###
### Authors: N. Salvati, N.Tzavidis, C. Giusti,
###          S. Marchetti and M. Pratesi
### File name: mq.sae.quant.R
### Updated: March 17th, 2010
###
############################################################

library(MASS)
library(np)


QRLM <- function (x, y, case.weights = rep(1, nrow(x)),
var.weights = rep(1, nrow(x)), ..., w = rep(1, nrow(x)),
init="ls",psi=psi.huber, cale.est=c("MAD","Huber","proposal 2"),
```

```
k2=1.345,method= ("M","MM"),maxit=20, acc=1e-04,test.vec="resid",q=0.5)
{
irls.delta <- function(old,new)sqrt(sum((old-new)^2)/
               max(1e-20,sum(old^2)))
irls.rrxwr <- function(x, w, r) {
w <- sqrt(w)
max(abs((matrix(r*w,1,length(r)) %*% x)/sqrt(matrix(w,1,length(r)) %*%
%*%(x^2))))/sqrt(sum(w*r^2))
}
method <- match.arg(method)
nmx <- deparse(substitute(x))
if (is.null(dim(x))) {
x <- as.matrix(x)
colnames(x) <- nmx
}
else x <- as.matrix(x)
if (is.null(colnames(x)))
colnames(x) <- paste("X", seq(ncol(x)), sep = "")
if (qr(x)$rank < ncol(x))
stop("x is singular: singular fits are not implemented in rlm")
if (!(any(test.vec == c("resid", "coef", "w", "NULL")) ||
        is.null(test.vec)))
stop("invalid testvec")
if (length(var.weights) != nrow(x))
stop("Length of var.weights must equal number of observations")
if (any(var.weights < 0))
stop("Negative var.weights value")
if (length(case.weights) != nrow(x))
stop("Length of case.weights must equal number of observations")
w <- (w * case.weights)/var.weights
if (method == "M") {
scale.est <- match.arg(scale.est)
if (!is.function(psi))
psi <- get(psi, mode = "function")
arguments <- list(...)
if (length(arguments)) {
pm <- pmatch(names(arguments), names(formals(psi)), nomatch = 0)
if (any(pm == 0))
warning(paste("some of ... do not match"))
pm <- names(arguments)[pm > 0]
```

```
formals(psi)[pm] <- unlist(arguments[pm])
}
if (is.character(init)) {
if (init == "ls")
temp <- lm.wfit(x, y, w, method = "qr")
else if (init == "lts")
temp <- lqs.default(x, y, intercept = FALSE, nsamp = 200)
else stop("init method is unknown")
coef <- temp$coef
resid <- temp$resid
}
else {
if (is.list(init))
coef <- init$coef
else coef <- init
resid <- y - x %*% coef
}
}
else if (method == "MM") {
scale.est <- "MM"
temp <- lqs.default(x, y, intercept=FALSE, method="S", k0=1.548)
coef <- temp$coef
resid <- temp$resid
psi <- psi.bisquare
if (length(arguments <- list(...)))
if (match("c", names(arguments), nomatch = FALSE)) {
c0 <- arguments$c
if (c0 > 1.548) {
psi$c <- c0
}
else warning("c must be at least 1.548 and has been ignored")
}
scale <- temp$scale
}
else stop("method is unknown")
done <- FALSE
conv <- NULL
n1 <- nrow(x) - ncol(x)
if (scale.est != "MM")
scale <- mad(resid/sqrt(var.weights), 0)
```

```
theta <- 2 * pnorm(k2) - 1
gamma <- theta + k2^2 * (1 - theta) - 2 * k2 * dnorm(k2)
qest <- matrix(0, nrow = ncol(x), ncol = length(q))
qwt <- matrix(0, nrow = nrow(x), ncol = length(q))
qfit <- matrix(0, nrow = nrow(x), ncol = length(q))
qres <- matrix(0, nrow = nrow(x), ncol = length(q))
for(i in 1:length(q)) {
for (iiter in 1:maxit) {
if (!is.null(test.vec))
testpv <- get(test.vec)
if (scale.est != "MM") {
if (scale.est == "MAD")
scale <- median(abs(resid/sqrt(var.weights)))/0.6745
else scale <- sqrt(sum(pmin(resid^2/var.weights,(k2*scale)^2))/
              (n1*gamma))
if (scale == 0) {
done <- TRUE
break
}
}
w <- psi(resid/(scale * sqrt(var.weights))) * case.weights
ww <- 2 * (1 - q[i]) * w
ww[resid > 0] <- 2 * q[i] * w[resid > 0]
w <- ww
temp <- lm.wfit(x, y, w, method = "qr")
coef <- temp$coef
resid <- temp$residuals
if (!is.null(test.vec))
convi <- irls.delta(testpv, get(test.vec))
else convi <- irls.rrxwr(x, wmod, resid)
conv <- c(conv, convi)
done <- (convi <= acc)
if (done)
break
}
if (!done)
warning(paste("rlm failed to converge in", maxit, "steps at q = ", q[i]))
qest[, i] <- coef
qwt[, i] <- w
qfit[, i] <- temp$fitted.values
```

```
qres[,i] <- resid
}
list(fitted.values = qfit, residuals = qres, q.values = q,
q.weights = qwt, coefficients = qest)
}

# COMPUTE THE QUANTILE ORDER

# COMPUTING OF THE QUANTILE-ORDERS
"zerovalinter"<-function(y, x)
{
        if(min(y) > 0) {
                xmin <- x[y == min(y)]
                if(length(xmin) > 0)
                        xmin <- xmin[length(xmin)]
                xzero <- xmin
        }

else {
                if(max(y) < 0) {
                        xmin <- x[y == max(y)]
                        if(length(xmin) > 0)
                                xmin <- xmin[1]
                        xzero <- xmin
                }
                else {
                        y1 <- min(y[y > 0])
                        if(length(y1) > 0)
                                y1 <- y1[length(y1)]
                        y2 <- max(y[y < 0])
                        if(length(y2) > 0)
                                y2 <- y2[1]
                        x1 <- x[y == y1]
                        if(length(x1) > 0)
                                x1 <- x1[length(x1)]
                        x2 <- x[y == y2]
                        if(length(x2) > 0)
                                x2 <- x2[1]
                        xzero <- (x2 * y1 - x1 * y2)/(y1 - y2)
                        xmin <- x1
```

```
                        if(abs(y2) < y1)
                                xmin <- x2
                }
        }
        resu <-  xzero
        resu
}

# Function for Finding the Quantile Orders by Linear Interpolation
# Assumes that "zerovalinter" function has been already loaded

"gridfitinter"<-function(y,expectile,Q)
# computing of the expectile-order of each observation of y by
# interpolation
{
nq<-length(Q)
  diff <- y %*% t(as.matrix(rep(1, nq))) - expectile
 vectordest <- apply(diff, 1, zerovalinter,Q)

#print(vectordest)
#qord<-list(ord=c(vectordest))
#qord
}

mq.coef<-function(myx,myy,myregioncode,maxiter=100){

 #This function estimates the m-quantile regression coefficients
 #myx<- x sample matrix of auxiliary variables
 #myy<- y vector
 #mynumauxvar<- number of auxiliary variables (include constant)
 #myregioncode<- area code for y and x units, data must be ordered by
  #area code
#maxiter<- OPTIONAL, number of maximum iteration for ob algorithm

myar<-unique(myregioncode)
myareas<-length(myar)
mysamplesize<-sum(as.numeric(table(myregioncode)))
mynumauxvar<-dim(myx)[2]

ob<-QRLM(myx, myy, maxit=maxiter,q=sort(c(seq(0.006,0.99,0.045),
```

```
0.5,0.994,0.01,0.02,0.96,0.98)))

qo<-matrix(c(gridfitinter(myy,ob$fitted.values,ob$q.values)),
nrow=mysamplesize,ncol=1)

qmat<-matrix(c(qo,myregioncode),nrow=length(myregioncode),ncol=2)
mqo<-aggregate(qmat[,1],list(d2=qmat[,2]),mean)[,2]
saq<-matrix(c(mqo,myar),nrow=myareas,ncol=2)
saq<-rbind(saq,c(0.5,9999))
ob1<-QRLM(myx, myy,maxit = maxiter,q=c(mqo[1:myareas]))
mycoef<-matrix(c(t(ob1$coefficients)),nrow=myareas,ncol=mynumauxvar)
# need to be ordered by area
mycoef<-t(mycoef)
mycoef
}

intsolver<-function(myqest,myyboot,myX,myregioncodepop,
mypopsize,myar,myareas,adjseed,mysboot){

myres<-array(0,dim=c(myareas))

myy<-myyboot[mysboot]
myx<-myX[mysboot,]
myregioncode<-myregioncodepop[mysboot]
myregioncoder<-myregioncodepop[-mysboot]
mysamplesizer<-as.numeric(table(myregioncoder))
myX.r<-myX[-mysboot,]

# M-quantiles
coef.boot<-mq.coef(myx,myy,myregioncode)

# Quantile Estimation Using the Chambers Dunstan Estimator
for(i in 1:myareas){
f1<-myy[myregioncode==myar[i]]
X.aux.i<-as.matrix(myX.r[myregioncoder==myar[i],])
pred.medr<-(X.aux.i%*%coef.boot[,i])
x.design.i<-as.matrix(myx[myregioncode==myar[i],])
pred.meds<-(x.design.i%*%coef.boot[,i])
res.s<-f1-pred.meds
#z<-matrix(rep(res.s,sample.sizer[i]),nrow=sample.sizer[i],
```

```
#ncol=sample.size[i])
z<-sample(res.s,mysamplesizer[i],replace=TRUE)
z<-z+pred.medr
z<-pmax(0,z)
comb<-c(f1,pred.medr)
start0<-quantile(comb,prob=c(myqest))
sameside<-T

while (sameside){
ff2<-sum(c(z)<=start0)
ff1<-sum(c(f1)<=start0)
f0<-1/(mypopsize[i])*(ff1+ff2)
if (f0<=myqest) start1<-start0+adjseed
if (f0>myqest) start1<-start0-adjseed
ff2<-sum(c(z)<=start1)
ff1<-sum(c(f1)<=start1)
f.new<-1/(mypopsize[i])*(ff1+ff2)
start.bef<-start0
start.aft<-start1
if (f0<=myqest & f.new>=myqest) sameside<-F
if (f0>=myqest & f.new<=myqest) sameside<-F
start0<-start1
}

ff2<-sum(c(z)<=start.bef)
ff1<-sum(c(f1)<=start.bef)
f.bef<-1/(mypopsize[i])*(ff1+ff2)
ff2<-sum(c(z)<=start.aft)
ff1<-sum(c(f1)<=start.aft)
f.aft<-1/(mypopsize[i])*(ff1+ff2)
fdif<-abs(f.bef-f.aft)
if (fdif>=0.01) {
start.med<-(start.bef+start.aft)/2
eps<-0.001
tol<-50

while (abs(tol)>=0.5){
ff2<-sum(c(z)<=start.med)
ff1<-sum(c(f1)<=start.med)
fmed<-1/(mypopsize[i])*(ff1+ff2)
```

```
tol<-(fmed-myqest)
fmedl<-1/(mypopsize[i])*(ff1+ff2)-eps
fmedu<-1/(mypopsize[i])*(ff1+ff2)+eps
if (fmed<myqest & fmedl<fmedu) start.bef<-start.med
if (fmed<myqest & fmedl>fmedu) start.aft<-start.med
if (fmed>myqest & fmedl<fmedu) start.aft<-start.med
if (fmed>myqest & fmedl>fmedu) start.bef<-start.med
#if (fmedl<fmedu) start.bef<-start.med
#if (fmedl>fmedu) start.aft<-start.med
#if (fmedl<fmedu) start.aft<-start.med
#if (fmedl>fmedu) start.bef<-start.med
start.med<-(start.bef+start.aft)/2
#print(tol)
}
}

if (fdif<0.01) {
start.med<-(start.bef+start.aft)/2
eps<-0.001
tol<-50

while (abs(tol)>0.5){
ff2<-sum(c(z)<=start.med)
ff1<-sum(c(f1)<=start.med)
fmed<-1/(mypopsize[i])*(ff1+ff2)
tol<-(fmed-myqest)
fmedl<-1/(mypopsize[i])*(ff1+ff2)-eps
fmedu<-1/(mypopsize[i])*(ff1+ff2)+eps
if (fmed<myqest & fmedl<fmedu) start.bef<-start.med
if (fmed<myqest & fmedl>fmedu) start.aft<-start.med
if (fmed>myqest & fmedl<fmedu) start.aft<-start.med
if (fmed>myqest & fmedl>fmedu) start.bef<-start.med
#if (fmedl<fmedu) start.bef<-start.med
#if (fmedl>fmedu) start.aft<-start.med
#if (fmedl<fmedu) start.aft<-start.med
#if (fmedl>fmedu) start.bef<-start.med
start.med<-(start.bef+start.aft)/2
#print(tol)
}
}
```

```
myres[i]<-start.med
#print(i)
}#Iteration i in bootstrap ends here
myres
}


boot.CD.R<-function(myqest,myyboot,myX,myregioncodepop,mypopsize,
mysamplesize,myar,myareas,myadjseed,myR,myid){

myproc.a<-array(0,dim=c(myareas,myR))

#Sampling from bootstrap population
for (r in 1:myR){
mysboot<-NULL
s.boot.i<-NULL
for (i in 1:myareas){
s.boot.i<-sample(myid[myregioncodepop==myar[i]],mysamplesize[i])
mysboot<-c(mysboot,s.boot.i)
}

myproc.a[,r]<-intsolver(myqest,myyboot,myX,myregioncodepop,mypopsize,
myar,myareas,myadjseed,mysboot)
}#R ends here
myproc.a
}


MQ.SAE.quant<-function(myqgrid,myy,myx,myX,myregioncode,myregioncodepop,
adjseed=max(0.15,mean(myy)/500),myMSE=FALSE,B=1,R=400,method="su"){

#This function estimate quantiles via CD estimator when
# n/N -> p with p very small

#myqgrid<- quantiles order to be estimated (i.e. 0.25,0.50,0.75)
#myy<- y vector
#myx<- x sample matrix of auxiliary variables
#myX<- X population matrix of auxiliary variables
#myregioncode<- area code for y and x units, data must be ordered
```

```
#by area code
#myregioncodepop<- area code for X units (population), data must be
#ordered by area code

#adjseed<- OPTIONAL, tune the value used to find two good starting point
# to solve the integral

#myMSE<-TRUE compute the MSE of the CD quantiles estimate via bootstrap,
# FALSE does not compute any MSE

#B<- number of bootstrap population
#R<- number of bootstrap samples
#method<- which method to be used to estimate residuals distribution,
#choice= "su" (smooth unconditional),"eu" (emprirical unconditional),
#"sc" (smooth conditional),"ec" (empirical uncoditional)


myar<-unique(myregioncode)
myareas<-length(myar)
mypopsize<-as.numeric(table(myregioncodepop))
mysamplesize<-as.numeric(table(myregioncode))
myquantnum<-length(myqgrid)
id<-seq(1:sum(mypopsize))

myarea.q<-matrix(0,myquantnum,myareas)
myq.true.boot<-array(0,dim=c(myquantnum,myareas,B))
myarea.q.boot.r<-array(0,dim=c(myquantnum,myareas,B,R))
myarea.q.boot<-array(0,dim=c(myquantnum,myareas,B))
myq.true.boot.m<-array(0,dim=c(myquantnum,myareas))
myarea.q.boot.m<-array(0,dim=c(myquantnum,myareas))
BIAS.boot<-matrix(0,myquantnum,myareas)
VAR.boot<-matrix(0,myquantnum,myareas)
MSE.boot<-matrix(0,myquantnum,myareas)
kk<-0

if(R%%2 != 0) stop("R must be even")

mycoef<-mq.coef(myx,myy,myregioncode)

for(qq in myqgrid){
```

```
qest<-qq
kk<-kk+1
myres<-NULL

for(i in 1:myareas){
f1<-myy[myregioncode==myar[i]]
X.aux.i<-as.matrix(myX[myregioncodepop==myar[i],])
pred.medtot<-(X.aux.i%*%mycoef[,i])
x.design.i<-as.matrix(myx[myregioncode==myar[i],])
pred.meds<-(x.design.i%*%mycoef[,i])
res.s<-f1-pred.meds
myres[i]<-list(res.s)
# z<-matrix(rep(res.s,pop.size[i]),nrow=pop.size[i],ncol=sample.size[i])
z<-sample(res.s,mypopsize[i],replace=TRUE)
z<-z+pred.medtot
z<-pmax(0,z)
comb<-c(pred.medtot)
start0<-quantile(comb,prob=c(qest))
sameside<-T

while (sameside){
ff2<-sum(c(z)<=start0)
# ff1<-sum(c(f1)<=start0)
f0<-1/(mypopsize[i])*(ff2)
if (f0<=qest) start1<-start0+adjseed
if (f0>qest) start1<-start0-adjseed
ff2<-sum(c(z)<=start1)
# ff1<-sum(c(f1)<=start1)
f.new<-1/(mypopsize[i])*(ff2)
start.bef<-start0
start.aft<-start1
if (f0<=qest & f.new>=qest) sameside<-F
if (f0>=qest & f.new<=qest) sameside<-F
start0<-start1
}

ff2<-sum(c(z)<=start.bef)
# ff1<-sum(c(f1)<=start.bef)
f.bef<-1/(mypopsize[i])*(ff2)
ff2<-sum(c(z)<=start.aft)
```

```
# ff1<-sum(c(f1)<=start.aft)
f.aft<-1/(mypopsize[i])*(ff2)
fdif<-abs(f.bef-f.aft)
if (fdif>=0.01) {
start.med<-(start.bef+start.aft)/2
eps<-0.001
tol<-50

while (abs(tol)>=0.5){
ff2<-sum(c(z)<=start.med)
# ff1<-sum(c(f1)<=start.med)
fmed<-1/(mypopsize[i])*(ff2)
tol<-(fmed-qest)
fmedl<-1/(mypopsize[i])*(ff2)-eps
fmedu<-1/(mypopsize[i])*(ff2)+eps
if (fmed<qest & fmedl<fmedu) start.bef<-start.med
if (fmed<qest & fmedl>fmedu) start.aft<-start.med
if (fmed>qest & fmedl<fmedu) start.aft<-start.med
if (fmed>qest & fmedl>fmedu) start.bef<-start.med
#if (fmedl<fmedu) start.bef<-start.med
#if (fmedl>fmedu) start.aft<-start.med
#if (fmedl<fmedu) start.aft<-start.med
#if (fmedl>fmedu) start.bef<-start.med
start.med<-(start.bef+start.aft)/2
#print(tol)
}
}

if (fdif<0.01) {
start.med<-(start.bef+start.aft)/2
eps<-0.001
tol<-50

while (abs(tol)>0.5){
ff2<-sum(c(z)<=start.med)
# ff1<-sum(c(f1)<=start.med)
fmed<-1/(mypopsize[i])*(ff2)
tol<-(fmed-qest)
fmedl<-1/(mypopsize[i])*(ff2)-eps
fmedu<-1/(mypopsize[i])*(ff2)+eps
```

```
if (fmed<qest & fmedl<fmedu) start.bef<-start.med
if (fmed<qest & fmedl>fmedu) start.aft<-start.med
if (fmed>qest & fmedl<fmedu) start.aft<-start.med
if (fmed>qest & fmedl>fmedu) start.bef<-start.med
#if (fmedl<fmedu) start.bef<-start.med
#if (fmedl>fmedu) start.aft<-start.med
#if (fmedl<fmedu) start.aft<-start.med
#if (fmedl>fmedu) start.bef<-start.med
start.med<-(start.bef+start.aft)/2
#print(tol)
}
}
#print(i)
myarea.q[kk,i]<-start.med
}#Iteration i ends here

if(myMSE){

######Bootstrap######

#Generate B bootstrap Population (size N)

if(method=="sc"){
#Centering residuals in each areas
#(use this for area conditioned approach)
res.s.centered<-NULL
for (i in 1:myareas){
res.s.centered[i]<-list(myres[[i]]-mean(myres[[i]]))
}

#smoothed density of residuals areas conditioned
Fhat.ord<-NULL
res.ord<-NULL
for (i in 1:myareas){
bw<-npudensbw(~res.s.centered[[i]],ckertype="epanechnikov")
Fhat<-fitted(npudist(bws=bw))
res.ord[i]<-list(sort(res.s.centered[[i]]))
Fhat.ord[i]<-list(sort(Fhat))
}
}
```

```
if(method=="su"){

#Centering residuals for the whole sample
# (use this for area unconditioned approach)

res.s.centered<-NULL
for (i in 1:myareas){
res.s.centered<-c(res.s.centered,myres[[i]])
}
res.s.centered<-sort(res.s.centered-mean(res.s.centered))

#smoothed density of residuals areas unconditioned
Fhat.ord<-NULL
bw<-npudensbw(~res.s.centered,ckertype="epanechnikov")
Fhat<-fitted(npudist(bws=bw))
Fhat.ord<-sort(Fhat)
}

if(method=="ec"){

#Centering residuals in each areas
#(use this for area conditioned approach)

res.s.centered<-NULL
for (i in 1:myareas){
res.s.centered[i]<-list(myres[[i]]-mean(myres[[i]]))
}
}

if(method=="eu"){

#Centering residuals for the whole sample
# (use this for area unconditioned approach)

res.s.centered<-NULL
for (i in 1:myareas){
res.s.centered<-c(res.s.centered,myres[[i]])
}
res.s.centered<-sort(res.s.centered-mean(res.s.centered))
```

```
}

for (b in 1:B){

cat(date(),"Generating Bootstrap Population ",b,"\n")

if(method=="sc"){
#Sample from kernel density areas conditioned
samp.boot<-NULL
for (i in 1:myareas){
s.boot<-NULL
for (g in 1:mypopsize[i]){
s.boot[g]<-which(Fhat.ord[[i]]==quantile(Fhat.ord[[i]],
    prob=runif(1),type=3))
}
samp.boot[i]<-list(s.boot)
}
#Population smoothed density of residuals area conditioned
y.boot<-NULL
y.boot.i<-NULL
for (i in 1:myareas){
y.boot.i<-myX[myregioncodepop==myar[i],]%*%mycoef[,i]+
+res.ord[[i]][samp.boot[[i]]]

y.boot<-c(y.boot,y.boot.i)
}
}

if(method=="su"){
#Sample from kernel density areas unconditioned
samp.boot<-NULL
for (i in 1:myareas){
s.boot<-NULL
for (g in 1:mypopsize[i]){
s.boot[g]<-which(Fhat.ord==quantile(Fhat.ord,prob=runif(1),type=3))
}
samp.boot[i]<-list(s.boot)
}
#Population smoothed density of residuals areas unconditioned
y.boot<-NULL
```

```
y.boot.i<-NULL
for (i in 1:myareas){
y.boot.i<-myX[myregioncodepop==myar[i],]%*%mycoef[,i]+
+ res.s.centered[samp.boot[[i]]]

y.boot<-c(y.boot,y.boot.i)
}
}

if(method=="ec"){
#Population empirical density of residuals area conditioned
y.boot<-NULL
y.boot.i<-NULL
for (i in 1:myareas){
y.boot.i<-myX[myregioncodepop==myar[i],]%*%mycoef[,i]+
+sample(res.s.centered[[i]],mypopsize[i],replace=TRUE)

y.boot<-c(y.boot,y.boot.i)
}
}

if(method=="eu"){
#Population empirical density of residuals area unconditioned
y.boot<-NULL
y.boot.i<-NULL
for (i in 1:myareas){
y.boot.i<-myX[myregioncodepop==myar[i],]%*%mycoef[,i]+
+sample(res.s.centered,mypopsize[i],replace=TRUE)

y.boot<-c(y.boot,y.boot.i)
}
}

for (ii in 1:myareas){
   myq.true.boot[kk,ii,b]<-quantile(y.boot[myregioncodepop==myar[ii]],
    prob=c(qest))
}

myarea.q.boot.r[kk,,b,1:R]<-boot.CD.R(qest,y.boot,myX,myregioncodepop,
mypopsize,mysamplesize,myar,myareas,adjseed,R,id)
```

```
#print(myarea.q.boot.r[kk,,b,])
#myarea.q.boot.r[kk,,b,(R/2+1):R]<-mycollect[[2]]
#print(myarea.q.boot.r[kk,,b,])

for (i in 1:myareas){
myarea.q.boot[kk,i,b]<-mean(myarea.q.boot.r[kk,i,b,])
}

}#B ends here

for (i in 1:myareas){
myq.true.boot.m[kk,i]<-mean(myq.true.boot[kk,i,])
myarea.q.boot.m[kk,i]<-mean(myarea.q.boot[kk,i,])
}

for (i in 1:myareas){
BIAS.boot[kk,i]<-myarea.q.boot.m[kk,i]-myq.true.boot.m[kk,i]
aux<-matrix(0,B,1)
for (b in 1:B){
aux[b,1]<-(1/R)*sum((myarea.q.boot.r[kk,i,b,]-myarea.q.boot[kk,i,b])^2)
}
VAR.boot[kk,i]<-(1/B)*sum(aux[,1])
}

for (i in 1:myareas){
MSE.boot[kk,i]<-((BIAS.boot[kk,i])^2)+VAR.boot[kk,i]
}

}#end if myMSE
}#Iteration k ends here
MSE.CD<-sqrt(MSE.boot)
quantiles<-myarea.q
rest<-list(quantiles=myarea.q,rmse=MSE.CD)
}
```

Bibliography

ANSELIN, L. (1988).*Spatial Econometrics. Methods and Models*. Boston: Kluwer Academic Publishers.

ARORA, V. & LAHIRI, P. (1997). Bayesian method over the BLUP in small area estimation problems. *Statistica Sinica* **7**, 1053–1063.

BANERJEE, S., CARLIN, B. & GELFAND, A.(2004). *Hierarchical Modeling and Analysis for Spatial Data*. New York: Chapman and Hall.

BRECKLING, J. & CHAMBERS, R.L. (1998). M-quantiles. *Biometrika*, **75**, 761–771.

BRUNSDON, C., FOTHERINGHAM, A.S. & CHARLTON, M. (1996). Geographically weighted regression: a method for exploring spatial nonstationarity. *Geographical Analysis*, **28**, 281–298.

CHAMBERS, R. & DUNSTAN (1986). Estimating distribution function from survey data. *Biometrika*, **73**, 597–604.

CHAMBERS, R. & TZAVIDIS, N. (2006). M-quantile models for small area estimation. *Biometrika*, **93**, 255–68.

CHAMBERS, R.L., CHANDRA, H., & TZAVIDIS, N. (2009). On robust mean squared error estimation for linear predictors for domains. *Paper submitted for pubblication. A copy is available upon request*.

CRESSIE, N. (1993). *Statistics for Spatial Data.*New York: John Wiley & Sons.

DATTA, G. S. & LAHIRI, P. (2000). A unified measure of uncertainty of estimated best linear unbiased predictors in small area estimation problems. *Statistica Sinica* **10**, 613–627.

DUAN, N. (1983). Smearing estimate: A nonparametric retransformation method. *Journal of the American Statistical Association*, **78**, 605–610.

DATTA, G. S. & RAO, J. N. K. & SMITH D. D.(2005). On measuring the variability od small area estimators under a basic area. *Biometrika* **92**, 183–196.

FAY, R. & HERRIOT, R. (1979). Estimates of income for small places: an application of James–Stein procedures to census data. *Journal of the American Statistical Association* **74**, 269–277.

GONZÁLEZ-MANTEIGA, W., LOMBARDÍA, M., MOLINA, I., MORALES, D. & SANTAMARÍA, L. (2008). Analytic and bootstrap approximations of prediction errors under a multivariate Fay–Herriot model. *Computational Statistics and Data Analysis*, **52**, 5242–5252.

JIANG, J. (1996). REML estimation: asymptotic behavior and related topics. *Annals of Statistics* **24**, 255-286.

LOMBARDIA, M., GONZALEZ-MANTEIGA, W. & PRADA-SANCHEZ, J. (2003). Bootstrapping the Chambers-Dunstan estimate of finite population distribution function. *Journal of Statistical Planning and Inference*, **116**, 367–388.

PRASAD, N. & RAO, J. (1990). The estimation of the mean squared error of small-area estimators. *Journal of the American Statistical Association* **85**, 163–171.

PETRUCCI, A. & SALVATI, N. (2006). Small area estimation for spatial correlation in watershed erosion assessment. *Journal of Agricultural, Biological and Environmental Statistics* **11**, 169–182.

PFEFFERMANN, D. & TILLER, R. (2005). Bootstrap approximation to prediction MSE for state-space models with estimated parameters. *Journal of Time Series Analysis* **26**, 893–916.

PRASAD, N. & RAO, J. (1990). The estimation of the mean squared error of small-area estimators. *Journal of the American Statistical Association* **85**, 163–171.

PRATESI, M. & SALVATI, N. (2008). Small Area Estimation: the EBLUP estimator based on spatially correlated random area effects. *Statistical Methods and Applications* **17**, 113–141.

PRATESI, M. & SALVATI, N. (2009). Small Area Estimation in the presence of correlated random area effects. *Journal of Official Statistics* **25**, 37–53.

ROYALL, R.M. & CUMBERLAND, W.G. (1978) Variance estimation in finite population sampling. *Journal of the American Statistical Association* **73**, 351–358.

SALVATI, N., TZAVIDIS, N., PRATESI, M. & CHAMBERS, R. (2008) Small Area Estimation Via M-quantile Geographically Weighted Regression. *University of Manchester, CSSR*, paper submitted for publication, currently under review.

SINGH, B., SHUKLA, G. & KUNDU, D.(2005). Spatio-temporal models in small area estimation. *Survey Methodology* **31**, 183–195.

TZAVIDIS, N., MARCHETTI, S. & CHAMBERS, R. (2010). Robust estimation of small area means and quantiles. *To appear in the Australian and New Zealand Journal of Statistics*.

YOU, Y. & CHAPMAN, B. (1979). Small Area Estimation Using Area Level Models and Estimated Sampling Variances. *Survey Methodology* **32**, 97–103.